Semantic Trajectories: Computing and Understanding Mobility Data

THÈSE N^{O} 5144 (2011)

PRÉSENTÉE LE 16 AOÛT 2011

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE DE SYSTÈMES D'INFORMATION RÉPARTIS
PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Zhixian YAN

acceptée sur proposition du jury:

Prof. C. Petitpierre, président du jury
Prof. K. Aberer, Prof. S. Spaccapietra, directeurs de thèse
Prof. J.-P. Hubaux, rapporteur
Prof. Y. Theodoridis, rapporteur
Prof. R. Weibel, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2011

- 献给我最最挚爱的亲人们
  - 父亲（严汉雨）母亲（左云南）
  - 岳父（吴继友）岳母（庞桂芳）
  - 妻子（吴静）
  - 小妹（严慧俊）

2011.08.01 瑞士洛桑

- ***Dedicated to My Precious Family***

  - *Father (Hanyu Yan), Mother (Yunnan Zuo)*
  - *Father-in-law (Jiyou Wu), Mother-in-law (Guifang Pang)*
  - *Wife (Jing Wu)*
  - *Sister (Huijun Yan)*

Zhixian @ Lausanne, Switzerland

1st August, 2011

Acknowledgements

The journey of this four-year PhD study at EPFL has been proved to be a unique learning experience for me, both at the professional level and on the personal aspect. I am sincerely grateful to many people who accompanied with me during this challenging and exciting doctoral-student period.

Different from most doctoral students at EPFL with only one PhD supervisor, I am exceptionally fortunate to be co-advised by two admirable professors, i.e., Prof. Stefano Spaccapietra and Prof. Karl Aberer. First and foremost, I am sincerely obliged to them for their extremely proficient and ingenious supervision.

I am deeply indebted to Stefano for taking me as his last PhD student. He always offers me his kindness, support, guidance, and friendship over the last four years. He enlightens me about all aspects of doing research and being a scientist, e.g., clarifying research problems, providing precise definitions, writing good papers in terms of critical reading, correcting my English, as well as enjoying the scientific lifestyle. Even after his official retirement, he keeps providing fully support and guidance on my thesis work. It is almost well-known that Stefano is not only a great scientist but also an honorable artist. From him, I in person learned that science and art are two sides of one coin and cannot be separated. The four-year guidance from him is extremely beneficial, both scientifically and personally. Therefore, I have to say that I am very lucky and happy to be his last PhD student.

I genuinely appreciate that Karl accepted to supervise me together with Stefano. Actually, since the very early stage of my PhD at EPFL, Karl has been continuously providing me a lot of constructive feedback and insights into this "semantic trajectory" study. Every time after the discussion with him, I could be able to broaden and deepen my research topics, and gain better ideas to target higher research levels. Without his guidance and suggestions, it would be impossible for me to expand this thesis study into such promising and challenging domains of mobility and sensing, which are well beyond the initial scope of this "trajectory" work. I definitely want to continue improving my research ability under his guidance. Therefore, my greatest gratitude goes to Karl for all of his kindness and supports from all perspectives.

I want to thank all of the jury members of my thesis defense, including Prof. Claude Petitpierre, Prof. Jean-Pierre Hubaux, Prof. Robert Weibel, and Prof. Yannis Theodoridis. I am appreciative of their time and efforts in reading and assessing this thesis work. A special thank goes to Prof. Hubaux for his agreement on serving as a committee member to evaluate my thesis at a very late request.

This thesis would not be able to reach this level without the contribution from many excellent colleagues and my paper coauthors, both internally and externally: Prof. Christine Parent, Prof. Yannis Theodoridis, Prof. Archan Misra, Prof. José Antônio Fernandes de Macêdo, Prof. Ying Ding, Dr. Dipanjan Chakraborty, Dr. Nikos Pelekis, Dr. Hoyoung Jeung, Nikos Giatrakos, Vangelis Katsikaros, etc. My greatest appreciation goes to Christine for her never-ending encouragement and supports during my doctoral work, and I personally consider her as my non-official PhD supervisor. Furthermore, she helped me translate this thesis abstract into French. In addition, I would like to particularly show my grateful to Dipanjan as we have co-built many nice works together. I was fully convinced by his patience in listening, organizing meetings and thoroughly taking notes. The collaboration with him is always productive and enjoyable. Regarding my two-month visit in Athens, I had a great time with my Greek coauthors and friends. Many special thanks go to Nikos, Vagelis and Despina for their kind hospitality, which made my stay in Athens very comfortable, enjoyable, and fruitful.

I would like to thank all of colleagues and friends in my two labs (LBD and LSIR) at EPFL, for their generous supports and friendships. The LBD members are: Marlyse, Shijun (1st office mate), Christelle, Tonho, Jana (2nd office mate), Fabio, Lina, Laura, David, Charles, and Catalin (3rd office mate) etc. The LSIR members are: Chantal, Dipanjan, Hoyoung, Saket, Ram, Surender, Alex, Hung, Mehdi, Zoltan, Nicolas, Wojciech, Urs, Thnasis etc. In particular, I want to express my sincere gratitude to the two secretaries, Marlyse and Chantal; I bothered them very frequently, but they never lost patience with me.

Besides my own PhD study, it was a very challenging and unexceptional experience for me in designing and guiding student projects. I had a great time in working with many master and bachelor students: Simone, viet Hung, Shuang, Lazar, Samy, Giorgi, Hoan, le Hung, Ming, Younes, Mouna. Thanks for their trusts in me and I learned a lot in trying to be a qualified mentor and good leader.

During last four years, the PhD journey became more colorful with a lot of Chinese friends in Lausanne. They are Shijun, Beilu, Hai, Huan, Luoming, Junmei, Jingshi, Feng, Rong, Jie, Yu, Xiuwei, Le, Fei, Hu, Li, Jiaqing, Zichong, Weijia, Ji, Jingmin, Na, Duan, Wenqi, Yuxuan, Yuanfang, Xiao, Xiaolu, Xinchao, Runwei, Yu, many others and new friends just arrived. It is too long to list all of their names here, but I would like to take this opportunity to express my appreciation and thank them all. I will never forget uncountable joyful social parties together.

Last but not least, even most importantly, my heartfelt gratitude goes to my precious family for their unconditional love, endless patience, and steady support, particularly to my father Hanyu Yan, mother Yunnan Zuo, younger-sister Huijun Yan, my father-in-law Jiyou Wu, mother-in-law Guifang Pang, and of course my great beloved wife Jing Wu. This doctoral dissertation is dedicated to them.

Abstract

Thanks to the rapid development of mobile sensing technologies (like GPS, GSM, RFID, accelerometer, gyroscope, sound and other sensors in smartphones), the large-scale capture of evolving positioning data (called *mobility data* or *trajectories*) generated by moving objects with embedded sensors has become easily feasible, both technically and economically. We have already entered a world full of trajectories. The state-of-the-art on trajectory, either from the moving object database area or in the statistical analysis viewpoint, has built a bunch of sophisticated techniques for trajectory data ad-hoc storage, indexing, querying and mining etc. However, most of these existing methods mainly focus on a spatio-temporal viewpoint of mobility data, which means they analyze only the geometric movement of trajectories (e.g., the raw $\langle x, y, t \rangle$ sequential data) without enough consideration on the high-level semantics that can better understand the underlying meaningful movement behaviors.

Addressing this challenging issue for better understanding movement behaviors from the raw mobility data, this doctoral work aims at providing a high-level modeling and computing methodology for semantically abstracting the rapidly increasing mobility data. Therefore, we bring top-down semantic modeling and bottom-up data computing together and establish a new concept called "*semantic trajectories*" for mobility data representation and understanding. As the main novelty contribution, this thesis provides a rich, holistic, heterogeneous and application-independent methodology for computing semantic trajectories to better understand mobility data at different levels. In details, this methodology is composed of five main parts with dedicated contributions.

(1) ***Semantic Trajectory Modeling***. By investigating trajectory modeling requirements to better understand mobility data, this thesis first designs a *hybrid spatio-semantic trajectory model* that represents mobility with *rich* data abstraction at different levels, i.e., from the low-level *spatio-temporal trajectory* to the intermediate-level *structured trajectory*, and finally to the high-level *semantic trajectory*. In addition, a semantic based *ontological framework* has also been designed and applied for querying and reasoning on trajectories.

(2) ***Offline Trajectory Computing***. To utilize the hybrid model, the thesis complementarily designs a *holistic* trajectory computing platform with dedicated algorithms for reconstructing trajectories at different levels. The platform can preprocess collected mobility data (i.e., raw movement tracks like GPS feeds) in terms of data cleaning/compression etc., identify individual trajectories, and segment them into structurally meaningful trajectory episodes. Therefore, this

trajectory computing platform can construct *spatio-temporal trajectories* and *structured trajectories* from the raw mobility data. Such computing platform is initially designed as an offline solution which is supposed to analyze past trajectories via a batch procedure.

(3) ***Trajectory Semantic Annotation***. To achieve the final semantic level for better understanding mobility data, this thesis additionally designs a semantic annotation platform that can enrich trajectories with third party sources that are composed of geographic background information and application domain knowledge, to further infer more meaningful *semantic trajectories*. Such annotation platform is *application-independent* that can annotate various trajectories (e.g., mobility data of people, vehicle and animals) with *heterogeneous* data sources of semantic knowledge (e.g., third party sources in any kind of geometric shapes like point, line and region) that can help trajectory enrichment.

(4) ***Online Trajectory Computing***. In addition to the offline trajectory computing for analyzing past trajectories, this thesis also contributes to dealing with ongoing trajectories in terms of real-time trajectory computing from movement data streams. The online trajectory computing platform is capable of providing real-life trajectory data *cleaning*, *compression*, and *segmentation* over streaming movement data. In addition, the online platform explores the functionality of *online tagging* to achieve fully semantic-aware trajectories and further evaluate trajectory computing in a real-time setting.

(5) ***Mining Trajectories from Multi-Sensors***. Previously, the focus is on computing semantic trajectories using single-sensory data (i.e., GPS feeds), where most datasets are from moving objects with wearable GPS-embedded sensors (e.g., mobility data of animal, vehicle and people tracking). In addition, we explore the problem of mining people trajectories using multi-sensory feeds from smartphones (GPS, gyroscope, accelerometer etc). The research results reveal that the combination of two sensors (GPS+accelerometer) can significantly infer a complete life-cycle semantic trajectories of people's daily behaviors, both outdoor movement via GPS and indoor activities via accelerometer.

Keywords: *semantic trajectory*, *structured trajectory*, *spatio-temporal trajectory*, *hybrid trajectory model*, *trajectory ontologies*, *trajectory computing*, *offline computing*, *online computing*, *activity recognition*, *indoor activities*, *outdoor movement*, *semantic annotation*, *movement behavior*

Résumé

Le développement rapide des technologies associées aux capteurs de position, tels que les GPS, GSM, RFID, accéléromètre, gyroscope, ou les fonctions de localisation des Smartphones, a rendu économiquement rentable et très facile l'enregistrement et le stockage à grande échelle des traces des positions des objets mobiles, tels que les camions de livraison, les taxis, les animaux ou les humains pourvus d'un tel capteur. Ces traces sont couramment appelées trajectoires. Nous sommes entrés dans un monde parcouru en tous sens de trajectoires. L'état de l'art sur les trajectoires, que ce soit dans le domaine des bases de données pour objets mobiles ou dans celui de l'analyse statistique des mouvements, comprend un vaste ensemble de techniques dédiées au stockage, à l'indexation, à l'interrogation, à la fouille, etc. des trajectoires. Cependant la plupart de ces techniques se limitent à l'étude des caractéristiques spatio-temporelles des trajectoires, c'est-à-dire à la séquence des positions (x_i, y_i, t_i). Elles ne prennent pas (ou trop peu) en compte la sémantique qui peut tre associée à ces positions, et ainsi rendent difficiles, voire impossibles, l'analyse et la compréhension du mouvement et in fine celles du comportement de l'objet mobile. Cette thèse contribue à l'étude du comportement des objets mobiles en proposant une modélisation des trajectoires à plusieurs niveaux, du niveau spatio-temporel pur au niveau sémantique, et une méthodologie associée qui permet de transformer, en plusieurs étapes, les données de base recueillies par les capteurs en données sémantiques. Nous obtenons ainsi des trajectoires sémantiques qui sont bien adaptées aux traitements des applications.

La thèse apporte cinq contributions principales qui sont décrites dans cinq chapitres:

(1) ***Modélisation sémantique des trajectoires***. A partir de l'analyse des besoins pour la description des mouvements, nous proposons un modèle de trajectoire hybride qui décrit les aspects spatiaux et les aspects sémantiques à différents niveaux d'abstraction. Le premier niveau décrit les trajectoires spatio-temporelles, c'est-à-dire uniquement leurs caractéristiques spatiales et temporelles. Le second niveau décrit les trajectoires structurées qui structurent la séquence des positions de la trajectoire en épisodes. Le dernier niveau décrit les trajectoires sémantiques dont les positions sont transformées en références à des objets du monde réel.

(2) ***Calcul des trajectoires en différé***. Nous avons défini un premier jeu d'algorithmes qui permet de construire les deux premiers niveaux de trajectoires à partir des séquences de données brutes recueillies par les capteurs. Une première

étape consiste à nettoyer et comprimer si besoin est les données recueillies. La seconde identifie dans la séquence des positions les trajectoires qui sont significatives pour l'application. On obtient alors des trajectoires spatio-temporelles. L'étape suivante segmente ces trajectoires spatio-temporelles en épisodes, afin d'obtenir des trajectoires structurées. Ces algorithmes ont été conus pour tre exécutés en mode différé.

(3) ***Annotation sémantique des trajectoires***. Afin d'obtenir les trajectoires sémantiques, nous avons défini un second jeu d'algorithmes qui permet d'enrichir les trajectoires avec des annotations qui référencent des objets géo-localisés du contexte. Les algorithmes peuvent utiliser n'importe quelle source de données qui décrit le contexte, comme une base de données spatiales ou cartographiques. Le mécanisme d'annotation marche pour tout type d'objet spatial, que ce soit des objets représentés par des points, des lignes ou des surfaces, ainsi que pour des trajectoires de type véhicules ou humains. Le résultat de ce processus d'annotation est les trajectoires sémantiques.

(4) ***Calcul des trajectoires en temps réel***. Ce chapitre est le pendant des chapitres (2) et (3) pour le temps réel. Nous avons défini un jeu d'algorithmes pour construire des trajectoires en temps réel au fur et à mesure de l'acquisition des données par les capteurs. Les algorithmes permettent d'effectuer le nettoyage et la compression des trajectoires en continu, ainsi que leur annotation. Nous pouvons ainsi construire des trajectoires en temps réel.

(5) ***Fouille des trajectoires recueillies par plusieurs capteurs***. Les algorithmes des chapitres précédents ont été conus essentiellement pour des données de type GPS, pratiquement les données des GPS de véhicules ou de personnes. Ce chapitre traite des données qui sont recueillies à l'aide de plusieurs types de capteurs, tels qu'un GPS et un accéléromètre ou gyroscope, comme c'est le cas des données recueillies par les smartphones. Le résultat de nos recherches est que disposer des données de deux types de capteurs (en l'occurrence GPS et accéléromètre) permet effectivement d'inférer quelles sont les activités poursuivies par les possesseurs de smartphones, et ce tant à l'extérieur qu'à l'intérieur.

Mots clés:

trajectoire sémantique, trajectoire structurée, trajectoire spatio-temporelle, modèle de trajectoire, calcul de trajectoire, calcul en différé, calcul en temps réel, annotation sémantique, inférence des activités, analyse du comportement.

Contents

Abstract **v**

Résumé **vii**

Contents **ix**

List of Figures **xv**

List of Tables **xix**

List of Algorithms **xxi**

1 Introduction **1**

1.1 Background 1

1.2 Motivation 3

1.3 Core Issues 3

1.4 Contributions 4

1.4.1 Hybrid Spatio-Semantic Trajectory Models 4

1.4.2 Offline Trajectory Computing 5

1.4.3 Trajectory Semantic Annotation 6

1.4.4 Online Trajectory Computing 7

1.4.5 Trajectory Computing from Multi-Sensors in Smartphones 9

1.5 Thesis Organization 9

2 State of the Art **11**

2.1 Introduction 11

2.2 Data Management for Trajectory 11

2.2.1 Trajectory Data Models and Systems 12
2.2.2 Trajectory Indexing 15
2.2.3 Trajectory Querying 16
2.3 Trajectory Data Processing 18
2.3.1 Trajectory Cleaning 19
2.3.2 Map Matching on Trajectory 20
2.3.3 Trajectory Compression 21
2.3.4 Trajectory Segmentation 23
2.4 Trajectory Data Mining 24
2.4.1 Trajectory Sequential Mining 24
2.4.2 Trajectory Clustering 25
2.4.3 Trajectory Classification 26
2.5 Semantic Processing of Trajectories 26
2.5.1 Ontology based Spatio-temporal System 26
2.5.2 Conceptual Views on Spatio-temporal System 28
2.5.3 Conceptual Views on Trajectory 28
2.6 Activity Recognition from Sensor Data 29
2.6.1 Activity Recognition from GPS Trajectories 29
2.6.2 Activity Recognition from Accelerometer Data 31
2.6.3 Activity Recognition from Multiple Phone Sensors 31
2.7 Summary 32

3 Semantic Trajectory Modeling 33

3.1 Introduction 33
3.2 Modeling Requirements 33
3.2.1 Trajectory Scenarios 34
3.2.2 Spatio-Temporal Knowledge 36
3.2.3 Semantic Knowledge 37
3.3 Trajectory Ontologies 38
3.3.1 Geometric Trajectory Ontology 40
3.3.2 Geography Ontology 42
3.3.3 Application Domain Ontology 42
3.3.4 The Complete Modular Ontology 43
3.4 Case Study on Trajectory Ontologies 43

3.4.1 Building the Semantic Trajectory Ontologies . . . 46
3.4.2 Querying the Semantic Trajectory Ontologies . . . 47
3.5 A Hybrid Spatio-Semantic Trajectory Model . . . 49
3.5.1 Data Model . . . 50
3.5.2 Conceptual Model . . . 52
3.5.3 Semantic Model . . . 53
3.6 Summary . . . 54

4 Offline Trajectory Computing **57**
4.1 Introduction . . . 57
4.2 Trajectory Computing Framework . . . 58
4.3 Data Preprocessing Layer . . . 59
4.3.1 Data Transformation . . . 60
4.3.2 Cleaning via Filtering and Smoothing . . . 61
4.3.3 Cleaning via Map Matching . . . 62
4.3.4 Data Compression . . . 67
4.4 Trajectory Identification Layer . . . 70
4.4.1 Identification via Raw Gap . . . 70
4.4.2 Identification via Prior Knowledge . . . 71
4.4.3 Correlation-based Identification . . . 72
4.5 Trajectory Structure Layer - Segmentation . . . 72
4.5.1 Trajectory Segmentation as Stop Discovery . . . 73
4.5.2 Velocity based Trajectory Structure . . . 77
4.5.3 Density based Trajectory Structure . . . 80
4.5.4 Other Trajectory Structure Methods . . . 82
4.6 Experiment . . . 87
4.6.1 Trajectory Datasets . . . 87
4.6.2 Trajectories at Different Levels – Data Abstraction . . . 88
4.6.3 Trajectories at Different Levels – Visualization . . . 89
4.6.4 Sensitivity Analysis of Computing Parameters . . . 91
4.7 Summary . . . 93

5 Trajectory Semantic Annotation **95**
5.1 Introduction . . . 95

5.2 Semantic Annotation Approach 96
5.2.1 Annotation Challenges 97
5.2.2 Annotation Framework 98
5.3 Annotation with Semantic Regions 102
5.3.1 Annotation with Free-Style Regions 102
5.3.2 Annotation with Well-Divided Grids 103
5.4 Annotation with Semantic Lines 107
5.4.1 Map Matching based Annotation 107
5.4.2 Inferring Transportation Mode 108
5.5 Annotation with Semantic Points 109
5.5.1 HMM-based Stop Sequence Modeling 110
5.5.2 Inferring Stop Activities 114
5.6 Experiment 116
5.6.1 Implementation Setup 116
5.6.2 Semantic Place Data Sources 117
5.6.3 Trajectory Annotation with Landuse 118
5.6.4 Trajectory Annotation with Road Networks 119
5.6.5 Annotation with Points of Interest 120
5.6.6 Web Interface and Performance 122
5.7 Summary 124

6 Online Trajectory Computing **125**
6.1 Introduction 125
6.2 Online Computing Approach 126
6.2.1 Real-time Settings and Challenges 126
6.2.2 Window Specifications for Online Computing 128
6.2.3 SeTraStream Overview 130
6.3 Online Data Preparation 131
6.3.1 Online Cleaning 132
6.3.2 Online Compression 134
6.4 Online Trajectory Segmentation 136
6.4.1 Sliding Window Method 136
6.4.2 Time and Space Complexity 137
6.5 Trajectory Episode Tagging 139

6.6 Experiment . . . 139
6.6.1 Experimental Setup . . . 139
6.6.2 Online Data Preparation . . . 140
6.6.3 Online Segmentation . . . 141
6.7 Summary . . . 142

7 Semantic Trajectory from Multi-Sensors 143
7.1 Introduction . . . 143
7.2 Learning from Multi-Sensors . . . 144
7.2.1 Activities from GPS+Accelerometer . . . 144
7.2.2 Two-Tier Computing Framework . . . 145
7.2.3 Challenges and Contributions . . . 147
7.2.4 Preliminaries and Definitions . . . 149
7.3 Micro-Activity Inference from Accelerometer . . . 151
7.3.1 Frame based MA Inference . . . 151
7.3.2 Description of the Empirical User Study . . . 154
7.3.3 Segmentation based MA Inference . . . 156
7.4 High-level Semantic Activity Inference . . . 159
7.4.1 Description of Empirical User Study . . . 159
7.4.2 The Order-Oblivious (**OO**) Approach . . . 161
7.4.3 The Sequence-Aware (**SA**) Approach . . . 161
7.4.4 Baseline approach . . . 163
7.5 Experiment . . . 163
7.5.1 Accelerometer Segmentation . . . 163
7.5.2 Mining Micro-Activity . . . 165
7.5.3 Mining High-level Semantic Activity . . . 165
7.6 Summary . . . 167

8 Conclusion and Outlook 169
8.1 Conclusion . . . 169
8.2 Open Issues and Future Directions . . . 171
8.2.1 Efficient Real-time & Distributed Trajectory Computing . . . 171
8.2.2 Collaborative & Personalized Semantic Trajectory Computing . . . 171
8.2.3 Online Activity Learning from Multiple Sensors . . . 172

8.2.4 Smartphone-based Trajectory & LBS Applications 172

Bibliography **173**

Curriculum Vitae **193**

List of Figures

1.1 Different kinds of mobility data scenarios 2
1.2 A semantic trajectory example 6

2.1 DOMINO system [WSX+99] 12
2.2 SECONDO system [GdAA+05] 12
2.3 HERMES system architecture [PTVP06] 14
2.4 Survey of spatio-temporal (trajectory) index methods [NDAM10] 17
2.5 Trajectory before map matching 20
2.6 Trajectory after map matching 20
2.7 Douglas-Peucker algorithm 22
2.8 Synchronized Euclidian distance 22
2.9 MoveMine system architecture [LJL+10] 27

3.1 Ontological infrastructure for modeling trajectories 39
3.2 Example concept hierarchies in spatial and temporal ontologies 41
3.3 Semantic trajectory ontology for a traffic management application 44
3.4 A hybrid semantic trajectory model 51
3.5 Spatio-temporal trajectory (a sequence of spatio-temporal points) 52
3.6 Structured trajectory (a sequence of episodes) 53
3.7 Semantic trajectory (a sequence of semantic episodes) 54

4.1 Offline trajectory computing framework 58
4.2 Data transformation of coordinates – from $\langle longitude, latitude \rangle$ to $\langle x, y \rangle$ 60
4.3 A trajectory example with two outliers 61
4.4 Filtering the outliers in trajectory data 62

4.5 Smooth GPS (x) . . . 63
4.6 Smooth GPS (y) . . . 63
4.7 Original sequence and smoothed new sequence . . . 63
4.8 Network constrained trajectories . . . 64
4.9 Perpendicular distance: between point and edge . . . 65
4.10 Distance between point and edge segment . . . 65
4.11 A global map matching approach . . . 66
4.12 Data compression of DP extensions using SED . . . 69
4.13 Data compression using STTrace via direction & speed . . . 69
4.14 Trajectory identification (from cleaned movement data to $\mathcal{T}_{spa}$) . . . 70
4.15 Trajectory structure (from $\mathcal{T}_{spa}$ to $\mathcal{T}_{str}$) . . . 74
4.16 Shared stops vs. personalized stops . . . 75
4.17 Building tree-based hierarchal *stops* . . . 76
4.18 Velocity-based stop identification . . . 77
4.19 Density-based stop identification . . . 80
4.20 The main notions in DBSCAN . . . 81
4.21 Speed time series ACF/PACF (original vs. differenced) . . . 84
4.22 Diagnose checking plots of TrajARIMA . . . 86
4.23 Trajectory speed forecasting . . . 87
4.24 Vehicle trajectory computing - data abstraction by $log_2(\frac{\#GPS}{\#dataComputed})$ . . . 89
4.25 People trajectories computing (results distribution) . . . 90
4.26 People trajectories computing (6 sample users) . . . 90
4.27 Visualization - from GPS feed to semantic trajectories . . . 90
4.28 Episodes in Milano car trajectories . . . 91
4.29 Δ_{speed} w.r.t. total stop number . . . 92
4.30 Δ_{speed} w.r.t. total stop time . . . 92
4.31 Example of map matching data with ground truth . . . 92
4.32 Sensitivity of map matching accuracy w.r.t. R/σ . . . 92

5.1 Logical view of trajectory annotation . . . 96
5.2 System architecture including annotation layers . . . 101
5.3 EPFL area free-style regions in Openstreetmap . . . 102
5.4 EPFL area free-style regions extracted and Google Earth visualization . . . 102
5.5 A region annotation example (visualized by Google Map) . . . 103

5.6 A region annotation example (visualized by trajectory Web interface) 103
5.7 Landuse ontology 104
5.8 The detailed landuse ontology: categories and aggregations (from Swisstopo) . . 105
5.9 Real landuse in Lausanne downtown area 106
5.10 Synthetic landuse with Gaussian distribution 106
5.11 A trajectory on landuse grids 107
5.12 Annotation trajectory with grids 107
5.13 Annotation trajectory with regions that merged by grids 107
5.14 Schmid's semantic trajectory compression via map matching 108
5.15 The procedure for inferring transportation mode 109
5.16 HMM formalism for inferring POI category 112
5.17 POI examples 113
5.18 POI densities 113
5.19 POI discretization and neighboring 114
5.20 POI category annotation result 116
5.21 Landuse distribution for annotating taxi data 118
5.22 Landuse distribution for taxi data on SeMiTri's Web Interface 119
5.23 Landuse category distribution and top-5 categories in people trajectories 120
5.24 Move annotation - a home-office example (via Metro) 121
5.25 Home-office (via Bike and via Bus) 121
5.26 Transport modes of *Home-office* moves (phone user 4) 122
5.27 Semantic stops/trajectories w.r.t. POIs by point annotation 122
5.28 Web Interface of SeMiTri 123
5.29 Trajectory computing and annotation latency measurement 124

6.1 Real-life distributed setting for online trajectory computing 126
6.2 From streaming movement data to semantic trajectory 128
6.3 Data and semantic information model 130
6.4 Online trajectory computing framework 131
6.5 Online data cleaning (outlier removal and smoothing) 140
6.6 Data compression rate w.r.t. different thresholds 141
6.7 $RV(B_l, B_r)$ in the two neighboring batches (without div_{thres}) 141
6.8 Episode identification varying batch size, for $\sigma = 0.6$ 141
6.9 Sensitivity of RV w.r.t. different σ at $\tau = 8s$ 141

6.10 Segmentation latency with different τ sizes (σ=0.6) 142
6.11 Segmentation latency with different σ thresholds ($\tau = 8s$) 142

7.1 Different levels of activities 145
7.2 Our Two-Tier Semantic Activity Inferencing Framework 146
7.3 User tag cloud and hierarchy of HA activity classes 150
7.4 Frame-based MA inference 151
7.5 MA Classification accuracy for $User1$ with fixed (vertical) phone orientation ($T_f = 5$ secs). 155
7.6 MA Classification accuracy for $User1$ with naturalistic (varying) phone orientations ($T_f = 5$ secs) 155
7.7 Confusion matrix of MA classification for one user 156
7.8 Segmentation of accelerometer stream 157
7.9 Example of features explored in our two-tier framework 160
7.10 Segmentation sensitivity analysis of error bound 164
7.11 Segmentation sensitivity analysis of accelerometer frame size 164
7.12 Segmentation results (division line) with the ground-truth tags) 164
7.13 MA classification accuracy via different frame sizes 165
7.14 MA classification accuracy across all 5 users ($T_f = 5$secs) 165
7.15 HA performance of baseline algorithm 166
7.16 HA-Classification accuracy for different feature-extraction approaches in SAMMPLE ($User1$) 166
7.17 Confusion matrix for HA classification of $User1$. 166
7.18 HA accuracy via different MA classifiers and MA frames 167
7.19 HA accuracy via different MA classifiers and HA classifiers 167
7.20 HA-classification accuracy for 5 subjects in SAMMPLE ($K_{max} = 4$; $\Theta_0 = 0.7$). . 167

List of Tables

2.1 Comparison of different trajectory database models/systems 15
2.2 The categories of query processing over trajectory data 18

3.1 Definition of spatiotemporal ontology 41
3.2 Definition of trajectory ontology 42
3.3 Definition of traffic management ontology 43
3.4 Selecting trajectory stops that are inside gas stations 47
3.5 Oracle semantic query using conjunctive query 48
3.6 Q1: Return cars which stopped at point (x,y) at t 48
3.7 Q2: Return cars which stopped today at a gas station 49
3.8 Creating a rule that defines the HomeOfficeTrajectory concept 49
3.9 Q3: Return persons who traveled yesterday afternoon from home to office 50

4.1 Symbol description and definition in TrajARIMA 84
4.2 Trajectory datasets - real-life GPS feeds 88
4.3 People trajectory data from mobile phones 88

5.1 Features for inferring transportation mode 110
5.2 An example of initial probability vector 111
5.3 An example of state transition matrix 112
5.4 Datasets of semantic places (third party sources for annotation) 117

6.1 Notations of symbols in SeTraStream 132

7.1 Descriptions of some non-obvious micro activity labels 150
7.2 Key features used in our micro-activity experiments 153

7.3 Summary of HA data collection campaign for different users 161

List of Algorithms

4.1 Map-Matching for Data Cleaning 68
4.2 Spatio-temporal Trajectory Identification (Begin/End) 73
4.3 getDynamicΔ_{speed} (gpsPoint, obj_{id}, δ) 78
4.4 Velocity-based trajectory structure 79
4.5 Density-based stop discovery - TrajDBSCAN 83

5.1 Trajectory annotation with ROIs 104
5.2 Trajectory annotation with LOIs 111
5.3 Trajectory annotation with POIs 115

Chapter 1

Introduction

Trajectory is the magic angles of projectile motion.

Haiduke Sarafian, 2000

This chapter presents the background and the motivation of this thesis work, as well as the contributions and organization of this dissertation.

1.1 Background

With the ubiquitous mobile positioning and tracking devices (such as wearable chips with embedded GPS – Global Positioning System, PDA – Personal Digital Assistants, and the increasing smartphones), it becomes technically convenient and economically cheap to collect the positioning data generated by several different kinds of moving entities, including humans, animals, and other non-biological moving objects like vehicles (see Figure 1.1 for different kinds of mobility data scenarios). GPS on smartphones is no longer an emerging trend, but almost a must-have feature nowadays. All of the state-of-the-art handsets like iPhone, Android, Nokia N-series and Windows phones can offer such positioning functionalities with embedded sensors. Berg Insight, a well-known IT company offering business intelligence to the telecom industry, forecasts that the shipments of GPS-enabled GSM/WCDMA handsets will grow to 960 million units in 2014, representing an attach rate of nearly 60%[1]. Furthermore, other cutting edge sensor-tracking techniques like GSM, radar, WiFi, RFID can also help capturing and preprocessing a huge amount of trajectory relevant mobility data very conveniently.

These kinds of GPS alike tracking and wireless sensing technologies significantly enhance the capabilities of a large amount of existing applications, and even foster new applications and services with locomotion feeds, e.g., ranging from traffic monitoring and environmental management, to land planning and geo-social networks. In recent years, there has been a tremendous surge in applications and services with locomotion feeds. To give some concrete trajectory application scenarios: (1) scientists implant GPS chips in animals to analyze the gregarious behavior of wild life, e.g., bird migration or monkey habits in forest. (2) smart phones

[1] `http://www.berginsight.com/` (2011)

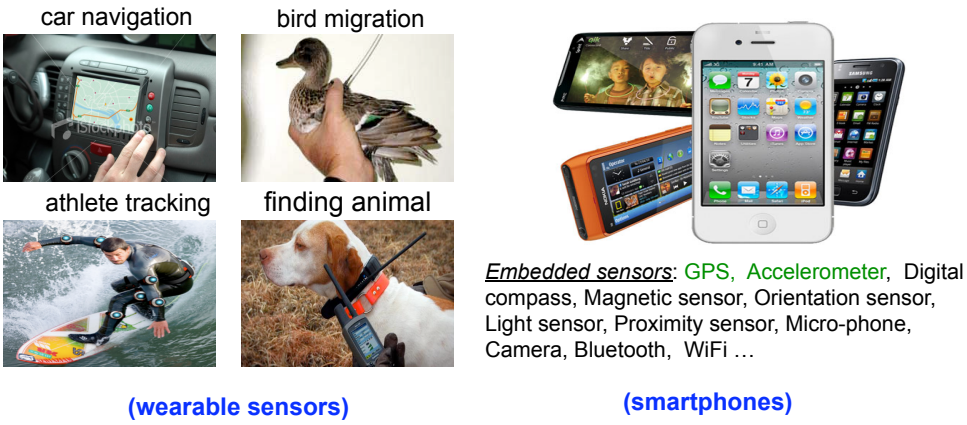


Figure 1.1: Different kinds of mobility data scenarios

(e.g., iPhone, Nokia N series) can help people very conveniently establish geo-social networks (e.g., Google Latitude, Foursquare, Facebook Place, Gowalla, Twitter) and access the interesting location-based services. (3) RFID technology installed in goods can improve the service quality of e-business with better tracking of shipment. (4) GPS-furnished moving vehicles can enhance real-time traffic analysis and provide better road planning to decrease or even avoid massive congestion in cities. These kinds of applications enhanced with trajectory data become more and more practicable and prevalent. Now, the world is already full of trajectories. This thesis is mainly focusing on the study of heterogeneous mobility data generated by various moving entities, e.g., vehicles, people, and animals. In Section 3.2, we will investigate the detailed requirements to provide a comprehensive model for mobility data understanding.

With such increasing GPS alike mobility data, as well as their applications like LBS (*Location Based Services*), there is a growing need for improving the capability to efficiently manage and analyze such huge amount of trajectory data produced by moving objects. Regarding these location-aware applications on trajectory data, the literature mainly focuses on studying the low-level geometric view of trajectory data, in particular in the domains of trajectory data management and statistical analysis of mobility etc.

The data management and database community, MOD (*Moving Object Database*) in particular, has focused on the design of spatio-temporal and trajectory datatypes, e.g., moving point and moving region datatypes by extending the established collections of spatial and temporal datatypes [KSF+03, GBE+00, WSX+99, PTVP06, GS05, GÖ5]; in addition, a lot of ad-hoc techniques for trajectory data indexing [PJT00, TVS96, SJLL00, CJL08] and querying [dAGB06, ZSI02, WXC00] (especially the nearest neighborhood relevant queries [BJKS02, GLC+07, BJKS06]) are proposed. In the meanwhile, mobility data analysis from the statistical perspective has built approximation functions for trajectory data regression and compression [FT07, MdB04, DP73, SRL09, KPT09], and explored data mining algorithms for trajectory patterns discovery [JSZ07, LHW07, JYZ+08, LHLG08], but their major focus is still on the raw tracking data with spatio-temporal views. Until very recently, research in these communities did not reach beyond the realm of handling and manipulating the evolving geometry that characterizes movement. These approaches leave to application developers all the burden of reconstructing and interpreting the meanings (semantics) to understand mobility data.

1.2 Motivation

Based on the previous brief discussion on the background of trajectories, either from moving object databases or with statistical analysis, we can come to a preliminary conclusion that: *moving object database has become a well-studied technology for trajectory indexing and querying, but only with spatio-temporal view*; *statistical analysis* (*e.g., trajectory data mining*) *has addressed low-level semantics about trajectories, but high-level semantics are still missing.* More detailed study on these related works will be discussed in the state-of-the-art in Chapter 2. To the best of our knowledge, most of these trajectory studies do not reach beyond the realm of handling and manipulating the evolving geometry that characterizes movement. Such methods simply assume trajectory as a curve that a moving object follows through a geometric space. However, real-world trajectories need to be reconsidered as the trace of a moving object that has not only the generic *spatio-temporal view* (the evolution of geometric location) but also the meaningful *semantic view* (the underlying meanings for better understanding the movement behavior), which is intensively overlooked in the literature.

Recently, newer research efforts [PSZ06, SPD+08, YMPS08b, ABK+07] have started to explore approaches that would support trajectory-based applications with rich conceptual models (e.g., stops and moves in trajectory) where semantics of movement can be explicitly expressed via application-aware trajectory modeling. Starting from these previous conceptual studies on trajectories, the motivation of this thesis is to further explore a comprehensive semantic approach for analyzing mobility data, not only from spatio-temporal (geometric) view, but also with semantic view. We aim at a *semantic data modeling and computing* method for understanding mobility data, which is able to not only explore the high-level semantic representation of trajectories but also support real-world trajectory application scenarios with a large scale of low-level tracking data (e.g., GPS feeds). Therefore, the core idea of this thesis can be considered as an integrated "*semantic and computing*" approach for trajectory data analysis, to better understand the movement behaviors from mobility data. To achieve this goal, on one side, we build a rich and comprehensive semantic representation for high-level trajectory modeling (called "semantic trajectories"); on the other side, we design a couple of computing techniques to utilize this rich model and reconstruct meaningful trajectories from real-life low-level GPS alike mobility tracking data.

1.3 Core Issues

To put our "semantic trajectory" motivation, namely *trajectory data analysis in terms of a semantic and computing approach*, into more concrete research statements, the following core issues and fundamental questions need to be explored and answered during this thesis work. We identify the six main research questions (Q_1 to Q_6) to be discussed in this dissertation.

(Q1) What are the fundamental modeling requirements for representing trajectory (mobility) data? What are low-level spatio-temporal view and high-level semantic view for trajectories? What is the main difference in these two views, why do we need both of them? What is the gap between the spatio-temporal view and the semantics, and how to bridge such gap?

(Q2) Can we provide a more comprehensive and rich semantic model for mobility data representation? How does this model meet all of the trajectory modeling requirements and represent both spatio-temporal and semantic views aforementioned? Any intermediate models are required and established between the spatio-temporal view and the semantic view?

(Q3) How can the semantic trajectory model be utilized in analyzing real-life trajectory applications and used for the GPS alike movement tracking data? Which kinds of computational solutions and algorithms need to be designed? Can statistical computing or data mining techniques be significantly adopted or redesigned for such computational tasks?

(Q4) How to further enrich trajectory semantics in addition to the trajectory computing? Does such semantic enrichment need additional 3rd party semantic sources, such as the geographic information or the application domain databases? How to annotate such semantics with extra semantic sources? Can we provide a generic and heterogeneous annotation framework to support such semantic enrichment on trajectories?

(Q5) Instead of providing offline trajectory mobility data computing, can we also provide a real-time computation of semantic-aware trajectories from streaming movement data? What is the major difference between online and offline situations, and how to enable our approach to work properly and efficiently in a real-time context?

(Q6) Vehicle trajectories from embedded GPS chips are usually quite stationary and the data is easy to handle with. However, for people trajectories collected from smartphones, the mobility data is quite heterogeneous (e.g., no GPS signals during people's indoor activities, people can take different transportation modes when they move, like car, bike, on foot etc). Can we compute semantic trajectories on heterogeneous people mobility data, e.g., inferring semantic indoor activities instead of outdoor movement? Can we combine different sensors (e.g., both GPS and accelerometer) from smartphones to better infer a complete semantic people daily trajectories?

1.4 Contributions

Towards the motivation and research challenges to establish a semantic approach for computing and understanding mobility data, this thesis formulates five major contributions, corresponding to answering the aforementioned six research questions.

1.4.1 Hybrid Spatio-Semantic Trajectory Models

The first contribution of this thesis is developing a new multi-perspective data abstraction and semantic approach for trajectory data modeling, namely the *Hybrid Spatio-Semantic Trajectory Model*. This model considers not only spatio-temporal view, but also semantic view for analyzing trajectories. The major novelties of this model can be summarized as follows:

- *Complete Modeling Requirement* – By analyzing the requirements to completely model trajectory data, we can claim that a comprehensive trajectory model should support not only *Spatio-temporal View* but also *Semantic View*. Spatio-temporal view is the major focus (or even the only one) in many traditional studies on moving object databases and so called trajectory databases; whilst semantic view explains the underlying meanings to understand trajectory movement behaviors, which is the missing point in most existing works and needs to be highlighted.

- *Spatio-temporal and Structured Trajectories* – Corresponding to *spatio-temporal view* and *semantic view* on trajectory data, we design two kinds of trajectories respectively, named *Spatio-temporal Trajectory* and *Semantic Trajectory*. Instead of directly building *semantic trajectory* from *spatio-temporal trajectory*, we design an intermediate model, i.e., *Structured Trajectory*, which can help bridging the gap between a pure spatio-temporal model and a pure semantic model, by capturing structured episodes in an individual trajectory.

- *Semantic Trajectories* – Regarding the semantic view, we design two types of semantics which can be used for semantically enriching trajectory data. The first one is *Geographic View* (e.g., landmarks, road networks) focusing on the geographic knowledge that ought to be integrated into the raw trajectory data; the second one is *Application Domain View* (e.g., home/office information in employee databases), as additional domain specific application knowledge for understanding trajectory behaviors.

- *Trajectory Ontologies* – Last but not least, we also propose an ontological infrastructure, *Trajectory Ontologies*, which also covers the previous three views (i.e., *spatio-temporal view*, *geographic view* and *application domain view*). Such trajectory ontological framework includes three components, i.e., *Geometry Trajectory Ontology* (GTO), *Geographic Ontology* (GO), and *Application Domain Ontology* (ADO). However, the ontological model is capable of providing even higher-level semantics, to support rich (conjunctive) querying and reasoning on trajectories.

The detailed results of the trajectory modeling will be presented in Chapter 3. Some preliminary results have already been published in [YMPS08a, YMPS08b]. This contribution is for answering relevant research questions in $Q1$ and $Q2$.

1.4.2 Offline Trajectory Computing

As the second major contribution of this dissertation, we design a practical computing platform for applying the previous *Hybrid Spatio-Semantic Trajectory Model* in the real-world tracking movement data. Basically, we present a bottom-up computing approach for constructing different levels of trajectories from real-life GPS-alike raw movement tracks, in terms of a couple of dedicated data computing layers in our offline computing platform.

- *Data Preprocessing Layer* – This is the preliminary task on processing the raw movement data, such as cleaning the GPS tracking records. We apply several data cleaning methods

like filtering and smoothing raw movement data for removing outliers and reducing errors, interpolating missing data points, map-matching GPS points to the underlying road networks, and compressing the data etc. As a result, we can achieve a cleaned version of movement data for the later data computing steps.

- *Trajectory Identification Layer* – This component divides the long cleaned movement data (i.e., the output of data preprocessing) into a set of subsequences, where each subsequence is corresponding to a single meaningful trajectory. Several trajectory identification polices are proposed, such as *Raw_GPS_Gap*, *Predefined_Time_Interval*, *Predefined_Space_Extent* and *Time_Series_Segmentation*. As a result, we can achieve *spatio-temporal trajectories*.

- *Trajectory Structure Layer* – This is the major component in trajectory computing, which tries to further segment each single trajectory into many meaningful units, called *trajectory episodes*. There are different kinds of episodes, such as *Begin*, *End*, *Stop*, *Move* discussed in [SPD+08]. The core issue in trajectory structure is to design relevant and robust stop discovery algorithms, such as *velocity-based* and *density-based* algorithms. As a result, we can achieve *structural trajectories*.

The details of offline trajectory computing will be presented in Chapter 4. Some preliminary results have already been published in [YPSC10]. This contribution is for answering relevant research questions in $Q3$.

1.4.3 Trajectory Semantic Annotation

To further establish the semantic meanings for understanding trajectories, we design a *semantic enrichment* layer upon the previous computing platform. Semantic enrichment can integrate structured trajectories with semantic knowledge from the two semantic viewpoints, i.e., *geographic view* and *application domain view*. As a result, we can achieve much more meaningful *semantic trajectories* compared to the previous spatio-temporal trajectories and structured trajectories. For example, Figure 1.2 shows a semantic trajectory example which has semantic annotations on each individual episodes, including stops (with the tags of *home*, *office*, *market*) and moves (with the tags of *bus*, *metro*, *walk*). In this thesis, we consider the semantic data sources from the geographic and application domain together, and divide them into three categories based on the underlying spatial extent, i.e., *region*, *line*, and *point*. According to these three spatial extents, this thesis designs a *heterogeneous annotation framework* with three dedicated annotation layers to semantically enrich mobility data.

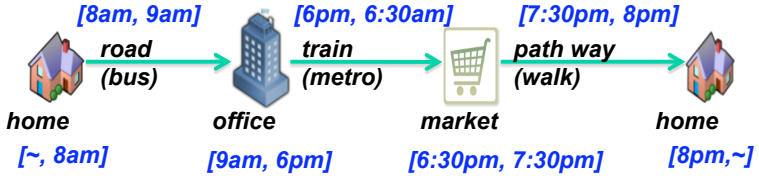


Figure 1.2: A semantic trajectory example

- *Annotation with Regions* – This layer enables annotation of trajectories with meaningful geographic or application domain sources of semantic regions. It does so by computing topological correlations between trajectories and 3rd party data sources containing semantic places of regions (called ROI - *Region of Interest*, or $\mathcal{P}_{region}$). We design a *spatial join* based algorithm, which can work for both regular grid-based regions (e.g., Landuse data) and free-style irregular regions (e.g., EPFL campus).

- *Annotation with Lines* – This layer annotates trajectories with LOIs (*Line of Interests*, or $\mathcal{P}_{line}$) like road networks and considers variations present in heterogeneous trajectories (e.g., vehicles run on road networks, while human trajectories use a combination of transport networks and walk-ways etc). Given data sources of different form of road networks, the purpose is to identify *correct* road segments as well as infer *transportation modes* such as *walking*, *cycling*, *public transportation* like metro. Thus, the algorithms in this layer include two major parts: the first part is designing a global map matching algorithm to identify the correct road segments for the move episodes of a trajectory, and the second one is inferring the transportation modes that the moving object used during the movement.

- *Annotation with Points* – This layer annotates the *stop* episodes of a trajectory with information about suitable *points of interest* (POIs, or $\mathcal{P}_{point}$). Examples of POI are *restaurant*, *bar*, *shops*, *movie theater* etc. For scarcely populated landscapes, it is relatively trivial to identify the objective of a stop (e.g., petrol pump on a high-way, back home in a very sparse residential area). However, densely populated urban areas bring several candidate POIs for a stop. Further, low GPS sampling rate due to battery outage and GPS signal losses makes the problem more intricate. We have designed a *Hidden Markov Model* (*HMM*) based technique for semantic annotation of *stops*. Unlike most other algorithms to identify POIs [ABK+07][XDZ09], an unique novelty of our approach is that it works for densely populated area with many possible POI candidates for annotation, thus catering to heterogeneous people and vehicle trajectories. It also enables identifying the activity (behavior) behind the stop, thus can further semantically annotate the whole trajectory.

The details of trajectory semantic annotation will be presented in Chapter 5. Some preliminary results have already been published in [YCP+11]. This contribution is for answering relevant research questions in $Q4$.

1.4.4 Online Trajectory Computing

Previously, we discuss the problem of trajectory computing, which is evidently necessary for mobility data processing and understanding, including tasks like trajectory data cleaning, trajectory identification, and segmentation to identify meaningful episodes like stops (e.g., while sitting or standing) and moves (while jogging, walking, driving etc). However, such methods in Chapter 4 like many related literatures are typically based on an offline procedure, which is not sufficient for real-life trajectory applications that rely on timely delivery of computed trajectories to serve real-time query answers. Therefore, this thesis alternatively proposes an online platform, namely "SeTraStream", for real-time semantic trajectory construction. Our online

framework is capable of providing trajectory online data *cleaning*, *compression*, *segmentation*, and even *tagging* over streaming movement data.

- *Online Cleaning* – We apply filtering methods like a Gaussian Kernel smoothing process or Kalman Filter to get rid of errors for the real-time streaming mobility data. There are two types of errors we need to deal with: the *systematic errors* (or called *outliers*) which are totally wrong positioning records of moving object from the true values and need to be removed; the *random errors* as the small noisy data need to be corrected and smoothed.

- *Online Compression* – An initial concern in our online setting regards the compression of the vast amounts of incoming location data. Given the limited memory resources, it is essential to reduce the amount of raw data and derive more compact representations describing an object's movement that will be later utilized by the upper levels of the trajectory computing framework. As new location data points arrive, the framework dynamically determines whether to keep a specific object's datapoint or not, by taking into consideration both (a) a *local accuracy* bound expressed as an L1-error threshold. That is, given the last received point of object O_i the next point has the potential to be thrown in case the resulting error does not exceed the previously defined threshold; and (b) a *global accuracy* bound which is needed to ensure that the final compressed representation will not be distorted above a certain tolerance percentage.

- *Online Segmentation* – A time series based online segmentation algorithm that operates on one single stream or multiple streams (e.g., velocity, direction, location, acceleration etc.) of the moving object. In this step, we can apply the online-window processing methods. To achieve parameterless or parametric-insensitive solutions, we design more advanced *similarity measurements* based on the statistical information of the streams (e.g., mean, variance, RV-coefficients), and identify trajectory breaking points based on *the movement similarity* computed from these statistical features.

- *Online Tagging* – Having detected an episode, SeTraStream manages to specify a tuple $(time_{from}, time_{to}, geometry_{bound}, tag)$ describing its spatio-temporal extent and semantics. Given application's context, possible *tag* instances form a set of movement pattern classes and notice that the instances of the classes are predetermined for the applications we consider. Hence, the problem of episode tag assignment can be melted to a trivial classification task, where the classifier can be trained in advance based on the collected episodes (with features like segment *distance*, *duration*, *density*, *average speed*, *average acceleration*, *average heading* etc.) and the detected episode e_i can be timely classified based on the trained model and the episode features.

The details of online trajectory computing will be presented in Chapter 6. Some preliminary results have already been published in [YGK+11]. This contribution is for answering relevant research questions in $Q5$.

1.4.5 Trajectory Computing from Multi-Sensors in Smartphones

In addition to the GPS alike mobility data, this thesis explores mining people trajectories using multi-sensor feeds from smartphones. Our purpose is to enrich people trajectories with semantic information about user's activities using the combination of phone sensors (GPS+accelerometer). While GPS as a sensor has been researched significantly, accelerometers have only been studied for activity recognition. GPS and accelerometers sense the user in partially overlapping dimensions of activities, movements, motion patterns. It is intuitive that knowledge extraction from such co-related streams should enrich the semantics of a trajectory, compared to a single sensor (either GPS or accelerometer). However, mining such information is non-trivial, specially considering completely naturalized settings. Our results reveal that the combination of the two sensors in smartphones can significantly infer semantic people trajectory of a more complete daily behaviors, in particular supporting the indoor semantic activities (e.g., cooking at home, breaking in office).

- *Two-tier inference framework* – We present a two-step process of semantic activity inferencing that (1) uses raw accelerometer streams to derive a sequence of an individual's micro-activities (e.g., sit, stand, walk), and (2) employs statistical feature extraction for mining on the micro-activities to establish the mapping from such micro-activities to the likely semantic activity (e.g., cooking at home). This two-step approach helps providing robustness against noise in the underlying sensor observations, and accommodates daily behavioral variations in semantic activities.

- *Micro-activities from accelerometer* – We explicitly consider the problem of accelerometer-based micro-activity classification in a *naturalistic environment*, where the individual goes about her daily life (i.e., non-control of usage style when people use the phone and data with noisy). The results show that a combination of orientation-independent and orientation-sensitive features provides the best classification accuracy (up to 90% in our studies) for such naturalistic settings.

- *Inferring high-level semantic-activities* – We propose and evaluate two discriminatory feature extraction techniques (utilizing the micro-activity stream) to identify specific semantic activities. The first approach uses features based only on the total *duration* of underlying micro-activities (resulting in a classification accuracy of about 70%), while the second approach additionally considers the *order* (or sequence) of these micro-activities (providing an additional about 10% increase in classification accuracy).

The details of mining semantic trajectories from phone sensors will be presented in Chapter 7. Some preliminary results have already been discussed in [YCM+]. This contribution is for answering relevant research questions in $Q6$.

1.5 Thesis Organization

After the general discussion about motivation and contribution of this thesis, the remainder of this dissertation is organized as follows:

- Chapter 2 presents a broad view of related works towards trajectory data from several different perspectives, including trajectory data management, processing, mining, semantic, and activity recognition aspects.
- Chapter 3 discusses trajectory modeling issues, including the analysis of real world trajectory scenarios and the investigation on model requirements, together with our two semantic models, i.e., the hybrid trajectory model and the trajectory ontologies.
- Chapter 4 focuses on the offline computing issues, which construct different levels of trajectories from the initial raw movement data. The fundamental problems are relevant trajectory data processing and stop identification algorithms.
- Chapter 5 investigates the semantic annotation algorithms to further enrich the semantics of the computed trajectories. It discusses the three dedicated annotation algorithms by using regions, lines and points, repshectively, from any third party geographic or application data sources.
- Chapter 6 discusses the real-time trajectory computing from streaming movement data, focusing on issues regarding online data cleaning, data compression, and trajectory segmentation, as well as timely tagging functionality.
- Chapter 7 explores the application of multiple sensors embedded in the smartphones, which can better infer semantic trajectories of a complete daily movement behavior of people's regular life, including both outdoor and indoor semantic activities.
- Chapter 8 provides concluding remarks. Based on the work in this dissertation, many interesting problems for further theoretical study and practical application exist. We present some interesting open issues and future research directions.

Chapter 2

State of the Art

In the present state of the art
this is all that can be done.

H.H. Suplee, 1910

2.1 Introduction

In this chapter, we review the literature related to this thesis work, in particular studies on managing and analyzing mobility data. We will describe their characteristics, advantages and drawbacks. For a better understanding with clear organization, we present these trajectory studies in terms of five main perspectives as the research background of this thesis, i.e., *trajectory data management*, *trajectory data processing*, *trajectory data mining*, *semantic trajectory analysis*, as well as some studies on *activity recognition* with mobility data.

This chapter is organized as follows: Section 2.2 discusses trajectory data management issues, particularly the study of trajectory data models, indexing and query processing techniques; Section 2.3 presents trajectory data processing techniques, especially these data cleaning, compression, map matching, and segmentation methods that are related to this thesis; Section 2.4 summarizes the trajectory mining methods including clustering, classification, and sequential pattern discovery; in Section 2.5, relevant trajectory studies on semantic aspect are described, including building ontologies and conceptual views for spatio-temporal and trajectory applications; Section 2.6 discusses related work on activity recognition within the context of trajectory or other mobile sensing data; and finally Section 2.7 summarizes the state-of-the-art chapter.

2.2 Data Management for Trajectory

Over the past decades, with the rapid development of consumer electronics (e.g., GPS-equipped PDAs, smartphones, and vehicles, as well as RFID-tag tracking and sensor networks), the database community started to face the larger availability of mobility data that is generated by moving objects and such data is typically called "trajectories". A lot of relevant database researches have been established and become hot topics in the data management field, such as the *Spatio-Temporal Database* [AR99, KSF+03], *Moving Object Database* (MOD) [Wol02, GS05]

and the emerging discipline of *Trajectory Database* [KO07]. Similar to traditional database studies, their main research objectives are to build ad-hoc data representation, storage, indexing and querying techniques for the data of moving objects and the trajectories they generate. There are many research issues, among which the most important ones are summarized in the following subsections, including trajectory data representation models/systems, indexing and query processing techniques.

2.2.1 Trajectory Data Models and Systems

Similar to many conventional database studies, the study on trajectory and moving object databases started from designing specialized datatypes for the modeling of spatial, spatio-temporal and moving objects [KSF+03, GS05]. Based on these specially designed and dedicated trajectory datatypes, a lot of relevant data operators and algorithms are proposed for indexing, querying and processing movements and trajectories [GBE+00, GP08].

In traditional DBMS (*Database Management System*), data is assumed to be constant unless it is explicitly modified, which cannot be applied for modeling moving objects or trajectories. In order to model such continuously varying location, several new data models are proposed. An early and representative data model called *Moving Objects Spatio-Temporal* (or MOST for short) has been proposed by Wolfson in [WXCJ98] for modeling *dynamic attributes* which can change continuously as a function of time without any explicit updates. With this dynamic attribute, query results depend on not only the contents but also the time. In contrast, a *static attribute* of an object is an attribute in the traditional sense, with only explicit update time. For modeling moving objects, the MOST model can represent spatial coordinates as dynamic attributes. A corresponding moving object database prototype, called DOMINO[1] (Databases fOrMovINg Objects tracking) [WSX+99], has been produced. As shown in Figure 2.1, the system can provide temporal capabilities, uncertainty management as well as the location prediction; the system is built upon existing object-relational databases (e.g., Oracle), as well as applying a GIS software for visualization (i.e., ArcView[2]).

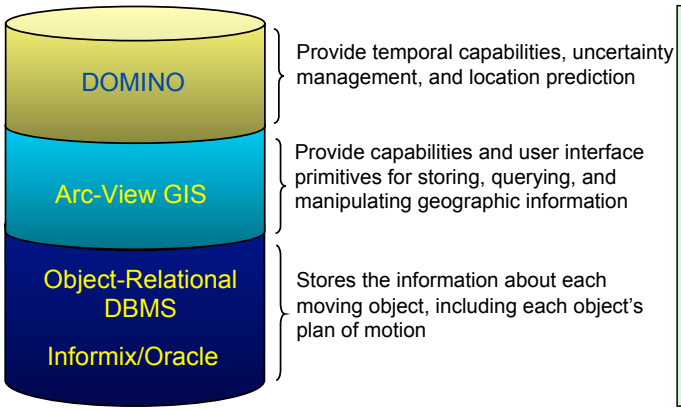


Figure 2.1: DOMINO system [WSX+99]

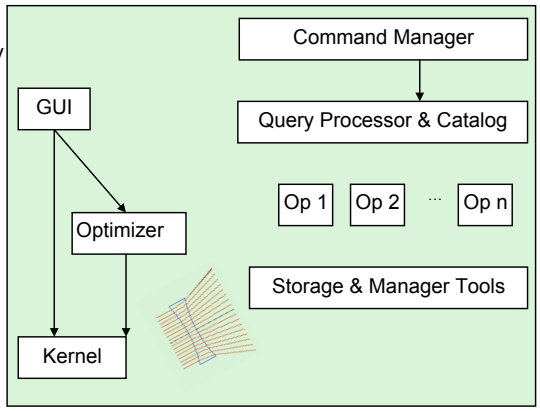


Figure 2.2: SECONDO system [GdAA+05]

[1] http://www.cs.uic.edu/~wolfson/html/mobile.html
[2] http://www.esri.com/software/arcview/

The second well-known moving object data model is built by Güting in an extensible database system called SECONDO[3], which supports non-standard database applications and moving objects in particular. With the extensions on spatial and temporal algebra modules, SECONDO uses spatial, temporal, and moving object datatypes to build moving object database system as non-standard applications [GÖ5, GdAA+05, GS05, GÖ7]. In such system, moving object is treated as a general time-varying geometry data type. New spatio-temporal datatypes are designed and integrated into existing relational, object-oriented, and other DBMS data models; furthermore, operations and algebras with formal semantics are defined for supporting spatio-temporal query languages [GBE+00]. In the implementation of the SECONDO system, there are three major components (see Figure 2.2) written in three distinctive programming languages: the kernel with a set of algebras to support query processing is implemented in C++; the optimizer implemented in PROLOG provides SQL-like query language with conjunctive query optimization; and a graphic user interface is written in Java. In addition, Güting et al. set up a benchmark system (called BerlinMOD[4] [DBG07]) for evaluating moving object databases. Based on the synthetic data generated by SECONDO with a simulated scenario, BerlinMOD can validate moving object databases in terms of two sets of trajectory querying problems, i.e., the range and the nearest-neighbor queries.

Besides SECONDO implementing the idea of time-varying moving object datatypes, HERMES[5] is yet another similar moving object database engine designed by Pelekis et al. The main functionality of HERMES is to support the modeling and querying of continuously moving objects [PTVP06, PFGT08]. The system is built on top of Oracle; a set of trajectory datatypes and corresponding operations are defined in an Oracle data cartridge, called *HERMES Moving Data Cartridge* (Hermes-MDC), based on the extension and combination of Oracle Spatial Cartridge and *TAU Temporal Literal Library* (TAU-TLL) Data Cartridge[6]. Besides working as a moving object database engine, HERMES can also be used either as a pure temporal or a pure spatial database system. A couple of recently emerging trajectory indexing and querying techniques have been already implemented in the HERMES trajectory database system [Fre08, PFGT08]. As shown in Figure 2.3, HERMES is implemented in PL/SQL based on the extension of Oracle spatial; it also supports the network constrained trajectory data, as well as provides a web demonstration for trajectory querying and visualization [PFGT08].

In addition to the above three most well-known trajectory database engines, there are several other research prototypes of trajectory databases. Meng and Ding in [MD03] provide a future-trajectory driven moving object database system, called DSTTMOD (*Discrete Spatio-Temporal Trajectory Based Moving Objects Database*). Different from most prior moving object databases, their claim is to support not only historical trajectories but also future location information. Aref et al. design PLACE (*Pervasive Location-Aware Computing Environments*), a scalable location-aware database server that can support continuous evaluation of queries over spatio-temporal data streams [MXA+04, XMA+04]. As built upon Nile[7] (a data stream management system

[3] http://dna.fernuni-hagen.de/Secondo.html/index.html
[4] http://dna.fernuni-hagen.de/secondo/BerlinMOD/BerlinMOD.html
[5] http://infolab.cs.unipi.gr/hermes/
[6] http://www.oracle.com/technology/documentation/
[7] http://www.cs.purdue.edu/Nile/

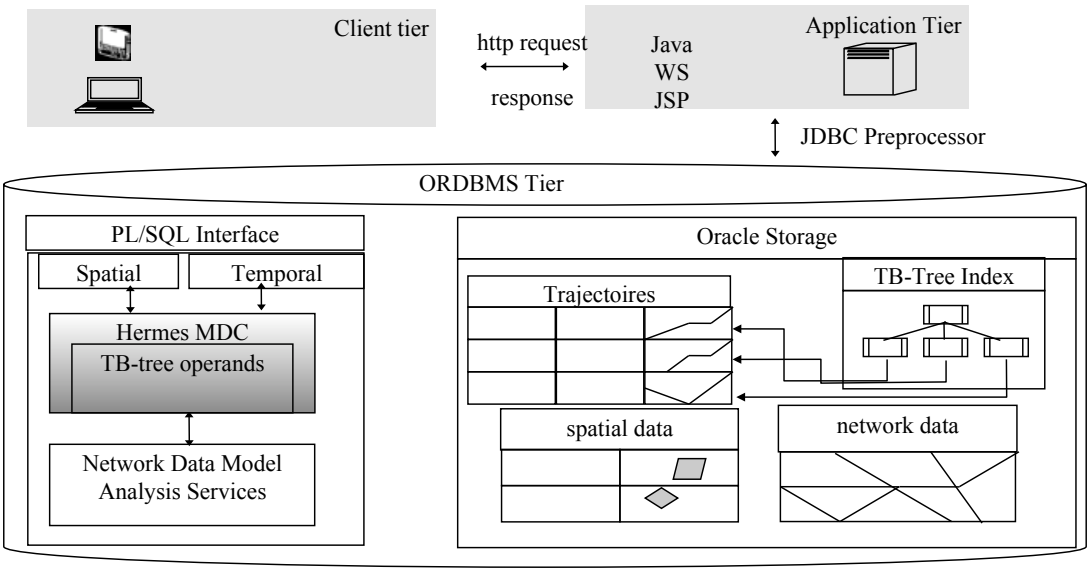


Figure 2.3: HERMES system architecture [PTVP06]

[HMA+04]), the PLACE server can support a wide variety of stationary and moving continuous spatio-temporal queries through a set of pipelined spatio-temporal operators, in terms of an incremental evaluation mechanisms in a real-time setting.

For a realistic trajectory model in real world applications, two additional issues need to be discussed, i.e., modeling with *network-constrained* movement and dealing with *uncertainty*. In real-life, many moving objects are restricted to moving in a given underlying spatial network, e.g., a transportation network like metro lines. Based on their early work on non-constrained moving object database [GBE+00], Güting et al. have extended new network interfaces to the standard time-varying moving object datatypes [GdAD06]. There are also some other relevant network-constrained moving object data models like [VW01] (with dedicated predicts to support network-related trajectory queries) and [SJK03] (with rich network models from computational perspective). Due to GPS alike mobile devices measurements and sampling errors, the recorded position of a moving object does not always represent its precise location [ZG02]. Frentzos in [Fre08] has summarized three possible sources for uncertainty: *imperfect observation of real world*; *incomplete representation language*; *ignorance, laziness or inefficiency*. In order to manage moving object uncertainty, many different uncertainty models have been proposed in the literature [TWHC04, PJ99, dAG05b, KO07]. It is worth noting that privacy is another important issue for real-life systems. For example, Gkoulalas et al. extend the HERMES system to HERMES++ for supporting privacy-aware trajectory tracking queries [GDV08]. In Table 2.1, we briefly summarize and compare these most-cited trajectory database engines. More detailed discussion on the query and index functionalities will be provided in the later two subsections.

These aforementioned moving object database systems are still research prototypes, without industrial strengths. The present commercial database management systems have built a couple of functionalities for supporting spatial or temporal applications separately (e.g., Oracle Spatial, SQL Server Spatial, PostgreSQL, PostGIS), in particular the requirements of spatial data like in

		DOMINO	SECONDO	HERMES	PLACE	DSTTMOD
Data-types	moving point	√	√	√	√	√
	moving region	NO	√	NO	NO	NO
	network	NO	√	√	NO	NO
Implementation	use-exist-db	√(Informix)	NO	√(Oracle)	√(NILE)	NO
	language	—	Java/c++/Prolog	PL-SQL	C++ (server)	—
	visualize	√(ArcView)	√ (Gui in Java)	√ (Web)	√ (Gui in C++)	√ (Gui)
Functionalities	query	√	√	√	√	√
	uncertain	√	NO	NO	NO	√
	predict	√	NO	NO	NO	√
	privacy	NO	NO	√	NO	NO

Table 2.1: Comparison of different trajectory database models/systems

Geographic Information Systems (GIS). There is still a long way to go for the industry towards moving object or trajectory databases.

2.2.2 Trajectory Indexing

Similar to traditional databases, the design and construction of an efficient and effective trajectory indexing structure is a very important and challenging topic, which can ensure high performance for trajectory data management, particularly for achieving efficient data querying. The moving object can move continuously and generate a huge amount of or even unlimited size of trajectories as time proceeds; therefore, trajectory indexing is crucial and becomes a possible bottleneck in many trajectory data application systems.

One of the first, most important and ubiquitous, multi-dimensional spatial indexing technique is R-tree [Gut84], which can be considered as a hierarchical data structure based on B^+-tree in multi-dimensional spaces. With the success of R-tree in spatial databases, it is well-used in various applications, from geographical information system (GIS) and computer-aided design to image and multimedia management systems [MNPT05, HMTT08]. R-tree, with its variants and extensions, actually dominates the domain of spatio-temporal data indexing, where spatial objects are represented by their *minimum bounding rectangles* (MBR). There are many examples such as three-dimensional R-tree [TVS96], TB-tree (Trajectory Bundle Tree) and STR-tree (*Spatio-Temporal R-tree*) [PJT00], and OP-Tree (*Octagon-Prism Tree*) [ZSI02]. Such trees can support efficient trajectory-based queries. Recently, Frentzos et al. further extend the TB-Tree methods and set up TB*-tree and FNR-tree (*Fixed Network R-tree*) [Fre08]. TB*-tree offers more trajectory operations, such as insertions, deletions and compressions; while FNR-tree can be used for indexing movements in a constrained space, which is another interesting study on indexing network-constrained moving objects and trajectories [DAG05a, PJ05].

The aforementioned TB-tree family is focusing on indexing the past trajectories. Another active study of spatio-temporal indexing is the family of TPR-tree (*the Time Parameterized R-tree*) [SJLL00] that is originally proposed by Jensen's group on the extension of R*-trees (a variant of the R-tree) [BKSS90]. In addition to the MBR of the spatial extents, TPR-tree maintains the *velocity bounding rectangles* (VBR), which can in turn index not only current

but also the anticipated future positions of moving point objects. Therefore, TPR-tree benefits to the query of present and anticipated future trajectories. Due to its success in indexing current and future trajectories, TPR-tree has 20+ successors developed by researchers from 2000 to 2008[8]. Among them, TPR*-tree is one of the most dominant extensions [TPS03], which significantly enhances the TPR-tree performance by integrating novel insertion and deletion algorithms during maintaining the index.

The R-tree based trajectory indexing techniques are prone to achieving low efficiency during update compared with traditional B^+-tree, because of the overlaps between bounding rectangles (BRs). Therefore, researchers also investigate traditional B^+-tree based indexing techniques for trajectories directly. For example, Jensen et al. design the B^x-tree with specific mapping functions from four-diminutional points (x,y,z,t) to one diminutional point [JLO04, JTT06a]. In such case, query needs to be transformed in each partition in the B^+-tree. For achieving better query performance, Yiu et al. extend B^x-tree to B^{dual}-tree by mapping both location and velocity information from four dimensions to one dimension [YTM08]. These B^+-tree based trajectory indexing methods are partitioning the space in trajectories instead of partitioning the data in R^+-tree based methods. With such velocity information, they can also support querying on both the present and the anticipated future trajectories.

The previous indexing techniques capture either the past (the position of an object up until the time of the most recent position sample) or the current & the anticipated/near future (the position based on a constant or linear function of time), not both of them. There is a hybrid solution for supporting both past and present (near future), i.e., RPPF-tree [PvJ06], which is based on the modification of partial persistence techniques.

Two excellent surveys have summarized these trajectory and spatio-temporal indexing techniques [MGA03, NDAM10], including a figure summarizing all of the spatiotemporal access methods (see Figure 2.4). There are also several benchmark systems (e.g., [CJL08, DBG07, JTT06b, MK03]) being established to evaluate these different kinds of indexing methods.

2.2.3 Trajectory Querying

Query processing over trajectory data includes several traditional spatial data querying techniques, such as point & region queries, and kNN (k nearest neighborhood) queries. In addition, trajectory and location based applications require advanced trajectory-based queries, which not only search historical and current trajectory data, but also predict future locations. By adopting the classification of spatio-temporal queries from Pfoser [Pfo02] and the moving object database benchmark BerlinMOD [DBG07], we can briefly summarize trajectory queries in terms of two types, i.e., coordinate-based and trajectory-based queries, see Table 2.2.

Coordinate-based queries are similar to traditional spatial database queries that focus on offering information about spatial coordinates, which include:

- *Point Query*: "*find the location of a specific moving object* (*mo*) *at a given time t*";
- *Range Query*: "*find all objects stayed a given area for a while* (*e.g., 10 mins*) *during a given time interval* $[t_1, t_2]$";

[8] `http://www.cs.aau.dk/~csj/tpr-tree-successors`

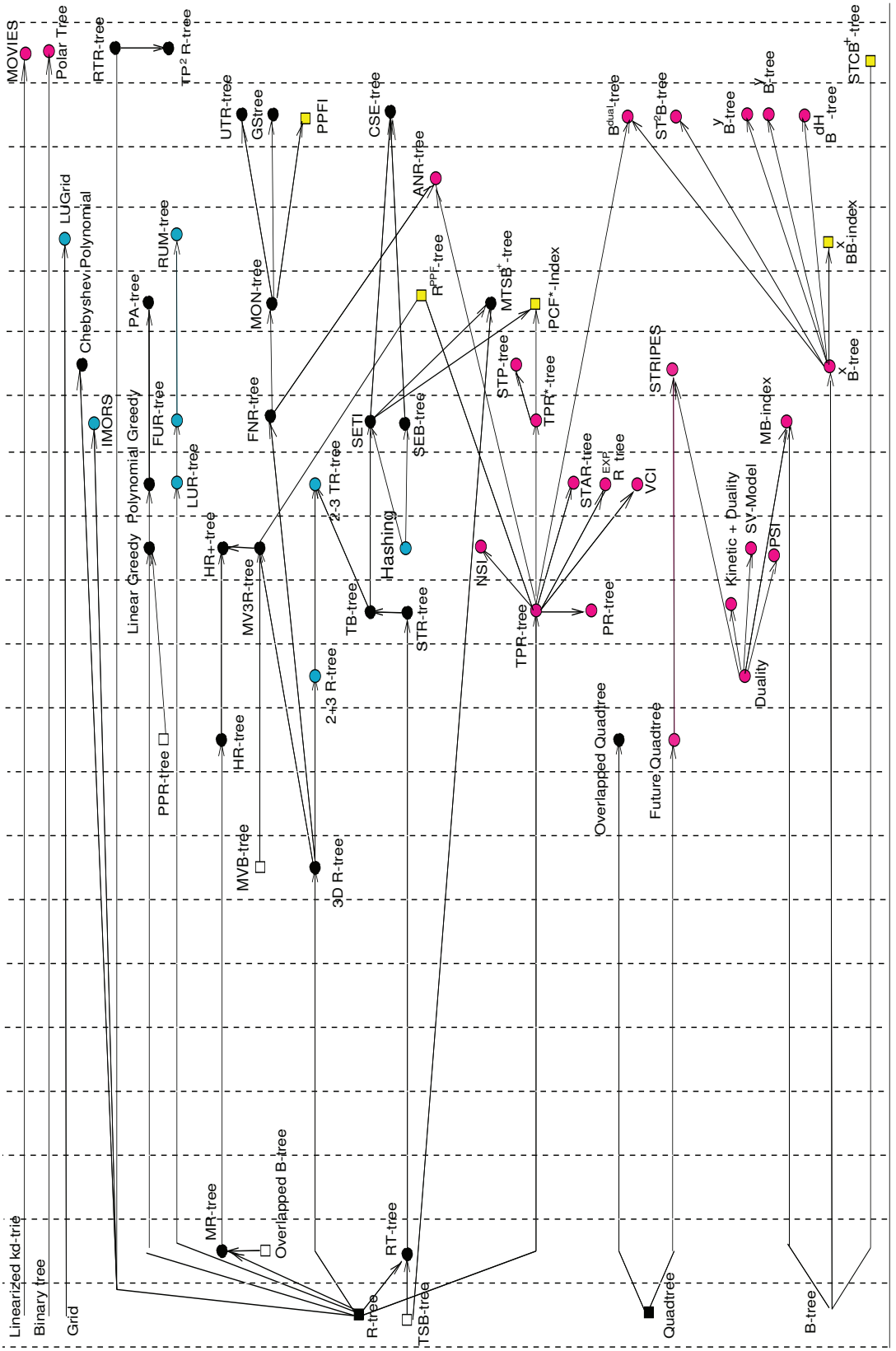


Figure 2.4: Survey of spatio-temporal (trajectory) index methods [NDAM10]

Query Type		*Operations*
Coordinate-based Queries	Point Query	at
	Range Query	overlap, inside etc.
	Nearest Neighbor Query	close, closest
Trajectory-based Queries	Topological Query	enter, leave, cross, bypass
	Navigational Query	traveled distance, covered area (top or average), speed, heading, parked

Table 2.2: The categories of query processing over trajectory data

- *k-Nearest Neighbor Query*: "*find k moving objects which are closest to a given position e.g.,* (x, y) *at a given time t*".

Among these three sub-types of spatial queries, k-nearest neighborhood (denoted "kNN") query is a major focus with many dedicated methods, together with a new type of interesting queries called "Reverse kNN (RkNN)" [FGPT05, BJKS06, GLC+07], e.g., $RkNN(q)$ returns the objects (x) in the database such that q is one of the k-nearest neighbor of x.

Trajectory-based queries become more important because of the nature (i.e., *location continuously varying*) of spatio-temporal and moving object data [PJT00]. For trajectory-based query, the first type is *topological queries* that involve the whole or a part of the trajectory of a moving object, and try to find the topological relations between moving objects and querying object (or space). For example, a typical trajectory-based query is "*to find whether an object enters, crosses, or bypasses a given area during a given time interval*". The second type of trajectory-based query is *navigational queries*, which involves important trajectory characterizations in real life trajectory relevant applications such as speed or heading of a trajectory. As mentioned before, a few dedicated indexing structures such as TP-tree and OP-tree [ZSI02, PJT00] are proposed for efficiently supporting these kinds of trajectory-based queries.

In [The03], Theodoridis categorizes ten benchmark queries for the applications of location-based services, among which many are trajectory related queries. These ten queries are divided into four categories: (*a*) *queries on stationary reference objects* including (1) point queries, (2) range query, (3) distance-based query, (4) nearest-neighbor query, (5) topological query; (*b*) *queries on moving reference objects* including (6) distance-based query, (7) similarity-based query; (*c*) *join queries* including (8) distance-join or closest-pair query, (9) similarity-join query; and (*d*) *queries with unary operators*, i.e., (10) unary operators on spatial and spatiotemporal data type. These queries can be applied for evaluating the capability of the applications of location based services.

2.3 Trajectory Data Processing

Different from the previous trajectory data management section that focused on the issues of database model and system performance, this section reviews the data processing techniques, in particular the problems about *data cleaning*, *map matching*, *data compression*, and *trajectory segmentation*. These are the main research issues as the trajectory reconstruction steps before building more useful trajectory databases and better understanding mobility data.

2.3.1 Trajectory Cleaning

In the literature of moving object and trajectory databases (discussed in Section 2.2), many studies (e.g., indexing and querying techniques) are built upon a common assumption that "*the positioning of moving objects can be precisely provided* ". However, real-life trajectory data is far more unreliable than expected to be used by the various applications. Datasets collected by mobile sensors are often imprecise, and as such incorrect due to noise [ZG02, YPSC10]. The reason resides in the limitation of positioning techniques (e.g., inaccurate GPS measurement and sampling errors, indoor signal loss, smartphone battery runs out) or on the privacy aspect of users (e.g., people do not want to disclose their precise or private locations, and deliberately expose only an approximation of positioning data [BLPW08]). Therefore, trajectory cleaning cannot be overlooked when reconstructing meaningful trajectory episodes from the raw locomotion data.

The main focus of trajectory data cleaning is to *remove or reduce GPS errors*. GPS errors are wrong measurements of the spatial position (i.e., not exactly the correct location). Jun et al. in [JGO06] summarizes two types of GPS errors: *systematic errors* and *random errors*. The systematic errors are the large wrong values that are totally invalid GPS positioning from the actual location, which may be caused by the low number of satellites in view and HDOP (*Horizontal Dilution Of Position*). The random errors are the small wrong values up to ± 15 meters that are caused by the satellite orbit, clock or receiver issues. Both types of errors refer to the spatial domain, as the temporal information is considered precise due to the high calibration clocks that the satellites are equipped with.

For the systematic errors, also named "outliers" in statistical analysis, researchers may resort to visual inspection which obviously is not practical for large datasets. Usually they design automated "filtering" methods to remove them. In this context, Marketos et al. propose a parametric online approach that filters noisy positions by taking into advantage of the maximum allowed speed of a moving object [MFN$^+$08]. A given threshold or parameter is used in order to determine whether a reported time-stamped position from the GPS stream, must be considered as noise and consequently discarded, or kept as a normal record.

On the other hand, random errors are small distortions from the true values and their influence is decreased by *smoothing* methods (e.g., [JGO06, Las06, SA09b]). Such techniques can be categorized based on their statistical backgrounds. A first such approach is based on the *least squares spline approximation* that targets to minimize overall error terms. More specifically, this approach minimizes the residual sum of squared errors, which is similar to regression-based smoothing techniques such as the local polynomial regression, cubic fits, exponential smoothing, and time series approximation models [JGO06]. A second category relies on a *kernel-based smoothing* method which adjusts the probability of occurrences in the data stream to modify outliers [YPSC10]. This approach bases on the idea of nearest neighbor smoothing and locally weighted regression models. The last category smoothes data points by recursively modifying error values by using a Kalman filter, which uses measurements observed over time (the positions coming in the GPS receiver), and predicts positions that tend to be closer to the true values of the measurements [Las06]. Eventually, the Kalman filter smoothes a position by computing a weighted average of the predicted position and the measured position.

In addition to removing or reducing errors, there are also *missing values* due to the signal loss (e.g., in tunnel or indoor) of GPS trajectories, which need to be interpolated in some applications. An example of such interpolation is provided in [HA06] where the authors use polynomial interpolation to estimate the missing data.

2.3.2 Map Matching on Trajectory

The previous trajectory cleaning methods are designed for objects that are allowed to move freely, without any constraints in their movement. However, in many applications moving entities such as people or vehicles are by their nature restricted to move within a given road network (i.e., a transportation network represented as a graph). For example, trains are restricted to move on a railway network, public buses have their own planned routes, and people are not allowed to walk on the highway. This type of trajectory data is called *network-constrained trajectories* [GdAD06]. Some dedicated network-constrained trajectory data models and indexing techniques are also mentioned in Section 2.2.

For analyzing network-constrained trajectory data, there is a very active research area called "map matching", that maps the positional measurements onto a network which it supposes to travel on, estimating the correct position in a node or edge of the network graph [QON07, BPSW05]. Note that the map matching problem is not present only due to the imprecision of the positions received by GPS devices, but also because digital maps (representing the network graph) are also subject to positional errors, while they can suffer from incomplete information (e.g., missing of new constructed roads). In this thesis, we focus on the first case, i.e., GPS positioning errors, and apply map matching to clean such network-constrained trajectory data. Figure 2.5 and 2.6 respectively sketch the raw trajectory data and the data after map matching, where the raw trajectory is $Q_1Q_2Q_3Q_4Q_5Q_6$ which is not necessarily consistent with the underlying road network (A_iA_j) because of GPS errors, whilst the map matched trajectory data $Q'_1Q'_2Q'_3Q'_4Q'_5Q'_6$ should be constrained by the network which would be the real trace of moving object. Therefore, the objective of map matching is to determine (or estimate) the actual trajectory traces in a given road network from the raw GPS tracking points with errors.

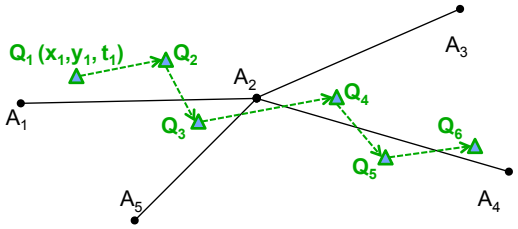


Figure 2.5: Trajectory before map matching

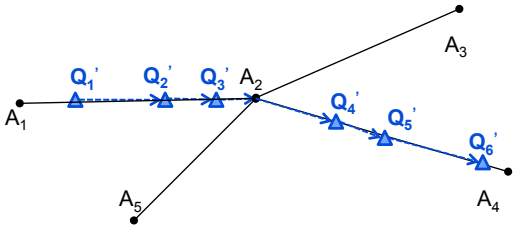


Figure 2.6: Trajectory after map matching

Based on the matching measurement, map matching methods can be simply classified into two categories: *geometric* and *topological*.

- Geometric methods (summarized in [BK96]) utilize only geometric information, by designing relevant distance measurements to identify the correct road segment and estimate the real location. Three types of distance measurements have been largely used, i.e.,

point-to-point (e.g., Euclidian distance), point-to-curve (e.g., perpendicular distance), or curve-to-curve (e.g., Fréchet distance). Point-to-point based map matching is matching the network only based on the road crossing points; point-to-curve based map matching is matching the network based on the road segments; curve-to-curve based map matching uses the distance between trajectory curve and network segments.

- In contrast to geometric approaches, the topological methods additionally account for the connectivity and contiguity of the road networks. For example, Greenfeld in [Gre02] proposes a method based on topological analysis using coordinate information on the observed position of the user; however, the method assumes no knowledge of the expected traveling route and does not use any speed or heading information supplied by the GPS. To avoid this drawback, Quddus et al. improves the algorithm that uses GPS information including position, velocity and time [QOZN03]. This information on the vehicle trajectory is used to avoid switching of mapped locations between unconnected road links. In the same context, Meng et al. analyze the correlations between the trajectory direction (e.g., GPS heading) and the road topology (e.g., U-turns, curvature, connections) [Men06]. By using additional topological information, the topological method can achieve better performance over pure geometric methods that only use distances. Recently, advanced map matching methods start to deal with more complicated GPS trajectories, such as with high ambiguity, noise and sparseness [NK09], in high-resolution networks [SA09a], and with low GPS sampling rate [LZZ+09]. All of these new methods investigate both distance and topology, and aim at a better global algorithm in the noisy data and/or complex network scenarios.

From another perspective, map matching methods can be classified into (1) *local* method [BK96], (2) *incremental* method [BPSW05], and (3) *global* method [Men06]. The local method focuses on individually matching a single GPS point to a path in the road network based on a given distance measurement (e.g., point-to-point/curve); a incremental method improves the local method and matches a portion of the trajectory (i.e., both current point and previous points); a global method can analyze the "entire" view of a trajectory (i.e., both past and forthcoming GPS points) and find the best matching road segment.

These various map matching proposals usually include several post-processing techniques to calibrate and correct the initial matching results. Obviously this could decrease the algorithm's efficiency (i.e., the cost/complexity of computation). Therefore, besides the effectiveness of a map matching algorithm that can be measured by the final matching accuracy, a decent map matching algorithm should also consider the algorithm's efficiency, in particular for working on a large amount of trajectory datasets.

2.3.3 Trajectory Compression

In real world, the trajectory of object movement is continuous in nature; whilst, due to the intrinsic limitations of data acquisition and storage devices, such inherently continuous movement are acquired, represented, and stored in a discrete way (e.g., a sequence of GPS points $\langle x, y, t \rangle$). Trajectory data in applications grow progressively and intensively as the tracking time goes by.

For example, a taxi-tracking system (with 5,000 cabs in San Francisco) collects 7.2 millions GPS points each day [LLLH10]. Such enormous amounts of data can sooner or later lead to storage, transmission, computation and display challenges. Therefore, trajectory data compression is an essential task of trajectory reconstruction. The main objective of trajectory compression is *data reduction*, which is capable of decreasing the computating complexity and introducing no more than a small error bound in trajectory querying.

From a geometric perspective, compression techniques explore the line simplification algorithms that remove positions from a trajectory without warping the trend of the trajectory or distorting the database significantly. The error bounds in trajectory data compression is the fundamental metrics to determine whether the data points can be removed or not. For achieving meaningful trajectory lossy-compression, researchers come up with several useful compression metrics, such as *spatial distance*, *synchronized Euclidian distance* (SED), *safe area* that are determined by speed and/or direction etc. in trajectory data.

Such spatial distances for data compression can be the pure Euclidean distance between two points, or the *perpendicular distance* between point and line[9], which have been essentially applied in line simplification in many domains like geometry, cartography, geospatial etc. For example, one of the standardized methods is the top-down Douglas-Peucker (DP) algorithm, which recursively splits the data series and selects the best position (i.e. the one has the largest perpendicular distance), and removes other positions in each loop. As shown in Figure 2.7, the original trajectory is presented in the dashed line collected by the black circle points, whilst the compressed one is the solid line only with selected white circle points. The original DP algorithm is a batch procedure. For streaming data compression, Meratnia and de By design the *opening window* techniques for online compression, among which there are two choices in threshold violation, i.e., using the point causes the violation (NOPW - *NOrmal Opening Window*) or using the point just before the violation (BOPW - *Before OPening Window*) [MdB04].

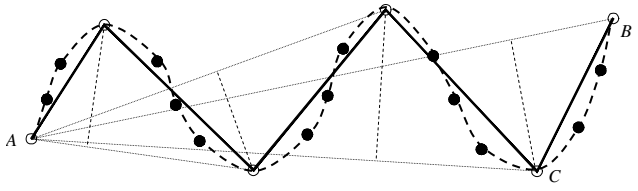


Figure 2.7: Douglas-Peucker algorithm

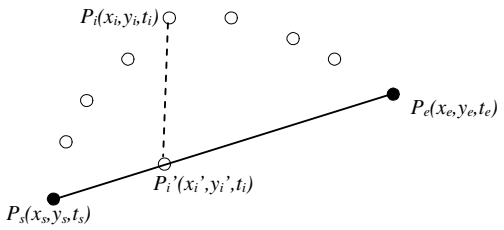


Figure 2.8: Synchronized Euclidian distance

Pure spatial distance and DP related algorithms are not suitable for compressing spatio-temporal trajectories, since both spatial and temporal data should be taken into account during trajectory data compression. Therefore, researchers propose the *Synchronous Euclidean Distance* (SED) [MdB04][CWT06] which combines both spatial and temporal information in the error bound metrics, defined as follows (also see Figure 2.8):

[9] In geometry, the perpendicular distance distance from a point (x_1, y_1) to the line $ax + by + c = 0$ is given by $d = \frac{|ax_1+by_1+c|}{\sqrt{a^2+b^2}}$

$$
\begin{aligned}
SED(P_i, P_s, P_e) =& |P_i, P_i'| = \sqrt{(x_{P_i} - x_{P_i'})^2 + (y_{P_i} - y_{P_i'})^2} \\
where, \quad & x_{P_i'} = x_{P_i} + v^{(x)}_{P_sP_e} \cdot (t_i - t_s) \\
& y_{P_i'} = x_{P_i} + v^{(y)}_{P_sP_e} \cdot (t_i - t_s) \\
& v^{(x)}_{P_sP_e} = (x_{P_e} - x_{P_s})/(t_e - t_s) \\
& v^{(y)}_{P_sP_e} = (y_{P_e} - y_{P_s})/(t_e - t_s)
\end{aligned} \tag{2.1}
$$

By applying the SED metric instead of the *perpendicular distance* between pure spatial Euclidean distance, the Douglas-Peucker (DP) algorithm can be extended for compressing GPS trajectories. For example, Meratnia and de By propose *Top-Down Time Ratio* (TD-TR) and *OPen Window Time Ratio* (OPW-TR) for the compression of spatio-temporal trajectories [MdB04], that are adapted from the previous top-down DP algorithm for in-batch compression and the opening window algorithm for online compression, respectively.

In addition to the SED in space, algorithms like STTrace in [PPS06] can preserve the speed and heading information in a trajectory, which defines the metrics of *safe area* to determine whether the GPS point needs to be kept or to be discarded. The safe area is computed by the speed tolerance and the direction tolerance. The STTrace algorithm is proved to be capable of performing better than pure threshold based sampling or compression techniques [PPS06]. Recently, Hönle [HGmRM10] and Muckell [MHLR10] empirically study these different kinds of compression algorithms and compare their performance.

It is worth noting that Frentzos et al. study trajectory compression from the perspective of its effects on spatio-temporal (time slice) queries [FT07, Fre08], instead of only focusing on the errors introduced by and the compression rate achieved in data compression.

2.3.4 Trajectory Segmentation

In contrast to a large amount of literature on map matching and trajectory compression, there are not so many specific works about trajectory data segmentation. Trajectory segmentation is originally from processing video-based motion trajectory (e.g., [TOY80, MJEM02]), where the input data is the visual tracking (images of tracking sequence of object movement) rather than the GPS-alike trajectory data studied in this thesis. Some GPS-alike trajectory segmentation works are treated as data preprocessing for additional trajectory studies, e.g., (1) from database perspective, evaluating splitting strategies to build better indexing for effective range queries [RSEN05]; (2) from data mining perspective, partitioning trajectory data and extracting more intuitive local patterns for clustering [LHW07] and classification [LHLG08].

Anagnostopoulos et al. firstly study GPS-alike trajectory segmentation [AVH$^+$06], but building segments on the MBR (Minimum Bounding Rectangles, see trajectory indexing in Section 2.2.2) as a simplified trajectory representation, not directly on the original trajectory (x, y, t) sequence. There is application-driven trajectory segmentation algorithm from biological researchers for splitting and identifying the distinctive human adenovirus motions in host cells

[HBK+07]; the drawback of such method is that we need to train the segment features in advance as a supervised learning method.

Nevertheless, trajectory segmentation is one of the most important topics in this thesis for building semantic trajectories and understanding mobility data. Segmentation is to divide a long trajectory into more useful sub-parts (we call them "trajectory episodes") and enables us to further assign meaningful semantic annotations for each episode. In Chapter 4 and Chapter 6, we will in particular discuss a couple of offline and online trajectory segmentation algorithms, which can automatically compute trajectory episodes.

Zheng et al. provide a change point based segmentation for GPS trajectories [ZLWX08, ZZXM09b], where the algorithm is firstly identifying *walk segments* and uses them to get other non-walk segments, and then inferring different transportation modes for each trajectory segment. In the context of GeoPKDD[10] (*Geographic Privacy-aware Knowledge Discovery and Delivery*) and MODAP[11] (*Mobility, Data Mining and Privacy*) projects, the two European projects that study the moving object data modeling and mining, researchers (including this thesis work) propose a couple of threshold (e.g., velocity) and clustering (e.g., spatio-temporal density) based trajectory segmentation algorithms, which is significantly based on the application of our early stop-move model in studying trajectory [GP08, SPD+08, YPSC10].

Recently, Buchin et al. present a theoretic framework that compute optimal segmentation using different criteria (e.g., speed, direction, location disk) from the computational geometry perspective [BDKS10]. However, there is no experimental study to validate such segmentation framework.

2.4 Trajectory Data Mining

From data mining and machine learning perspectives, there are many literatures studying trajectory mining, where the objective is knowledge discovery from such mobility data [GP08, HLGL08]. Similar to the geometric focus of data management and processing, the studies of trajectory mining assume trajectory as a sequence of spatio-temporal point (x, y, t), where (x, y) is the location coordinate in 2D space and t is the timestamp. Similar to most conventional data mining studies, trajectory data mining also aims at typical knowledge discovery issues such as *sequential mining*, *clustering*, *classification*, *outliers detection* and even location *prediction* etc.

2.4.1 Trajectory Sequential Mining

Sequential pattern mining is a hot topic in sequence databases, where the data consists of ordered items; therefore, it is also interesting for the spatio-temporal data and trajectories of moving object. The discovery of so called spatio-temporal sequential patterns (or *trajectory sequential patterns*) can show the cumulative and consecutive behavior of moving objects and help understanding the mobility-relevant repeated sub-sequences [GNPP07, GP08]. For example, a sequential pattern for the movement of a EPFL researcher could be simplified as "*home* → *office* → *home*", as his/her repeated workday movement behavior.

[10] http://www.geopkdd.eu/
[11] http://www.modap.org/

A couple of extensions on traditional sequential mining algorithms such as GSP (*Generalized Sequential Patterns*) [SA96], PrefixSpan [PHMA+01, PHMA+04] and SPADE (*Sequential PAttern Discovery using Equivalent Class*) [Zak01] have been proposed for spatio-temporal and trajectory sequential mining [CMC07, TG01, HYD99, GP09]. Among these sequential mining studies, T-pattern is a representative sequential trajectory pattern being widely cited for mining frequent movement behaviors, which considers both space (i.e., the regions that moving objects have traveled and stayed for a while) and time (i.e., the time *duration* for staying in certain regions during movement, as well as the *order* of the visit) [GNPP07]. Based on such trajectory pattern algorithm, the group of Giannotti further designs a tree hierarchy based on the connection of many T-Pattens, and provides the functionality of location prediction by using this tree structure [MPTG09].

Another interesting sequential study on trajectory is mining periodic behaviors of moving objects [CMC07, LDH+10]. Periodic pattern can be roughly defined as the repeating activities happened in certain locations with regular time interval at periodic time instances; therefore, it can be significantly used for location and behavior prediction. For example, Li et al. design a two stage algorithm (called "Periodica") in [LDH+10], for mining such periodic trajectory patterns: firstly, the periods are detected by reference spots using Fourier transform and autocorrelation; secondly, periodic behaviors are summarized using hierarchical-based clustering.

2.4.2 Trajectory Clustering

Trajectory clustering is the second major topic in trajectory data mining, which aims at discovering the similarity in a set of movement trajectories, grouping similar trajectories into the same cluster, and finding the most common movement behaviors. Many studies focus on extending the well-known clustering algorithms and apply them in trajectory data, e.g., k-means [Llo82], BIRCH (*Balanced Iterative Reducing and Clustering using Hierarchies*) [ZRL96], DBSCAN (*Density Based Spatial Clustering of Applications with Noise*) [EKSX96], OPTICS (*Ordering Points To Identify the Clustering Structure*) [ABKS99], and STING (*STatistical INformation Grid based clustering method*) [WYM97]. Compared with these traditional clustering techniques, the extension for trajectory clustering requires a thorough examination of the grouping semantics along the time dimension. To give a couple of such extension examples: (1) several k-mean based extensions for clustering trajectory data [JLO07, LWC+07, KMB05]; Nanni et al. in [NP06] propose T-OPTICS as an adaptation of OPTICS to trajectory data with another notion of distance between trajectories; (3) DBSCAN-based extensions [PBKA08, BK07] redefine the core concept of *density-reachable* that can include both spatial distance threshold and temporal distance threshold. However, the research focus is still on clustering the space and the time dimension separately, which are more suitable for the generic spatio-temporal datasets (e.g. fixed sensor readings of weather conditions), not directly applicable to the trajectories of the movement data.

Alternatively, new algorithms are proposed for partially trajectory clustering, which firstly split the trajectory into several pieces (like trajectory segmentation, but usually based on a simple method of manual division, e.g., via distance) and then build clusters on partial trajectories. For example, Lee et al. provide a partition-and-group framework for trajectory clustering

[LHW07], where a formal trajectory partitioning algorithm (based on the theory of *minimum description length* - MDL) is proposed and the clustering on sub-trajectories is highlighted; Jueng et al. propose a convoy discovery algorithm in trajectory database [JYZ+08], where trajectory convoy is a group of objects that have traveled together for some consecutive time intervals, not necessarily for the complete trajectory lifespan of the moving object.

2.4.3 Trajectory Classification

As another important mining task, *trajectory classification* has been significantly studied in conventional flock and other biological relevant trajectory datasets, where the focus is on classifying flock trajectories and finding flock movement behaviors such as *leadership*, *convergence*, *encounter* [AGLW07, BGHW08]. Furthermore, there are some studies aiming at defining a comprehensive taxonomy for movement patterns [DWL08, GLW08], which can be used as a guidance for designing trajectory classification algorithms. Many machine learning techniques (in particular the Hidden Markov Model) have been applied in trajectory classification, e.g., [STK02] for biological trajectories, [BKS07, JSZ07, NFM10] for human motion trajectories using unsupervised learning methods on the coefficients of the basis functions. Lee et al. use trajectory partitioning and classify trajectories based on their representative portions (e.g., regions and movement flows) [LHLG08]. Moving object anomaly detection is kind of the special case of the study on trajectory classification, where the objective is to distinguish the normal movement behaviors with irregular ones. Li et al. in [LHL08, LLHL09] propose a trajectory outlier detection method based on motifs, which are the frequently occurring subsequences or patterns that are defined and well-applied in time series study [PKLL02].

A decent trajectory mining system (called "MoveMine"[12]) has been recently released from Han's group [HLGL08, LJL+10], which includes four main mining functionalities, i.e., (1) periodic pattern mining, (2) swarm and convoy pattern mining, (3) trajectory clustering, (4) trajectory outlier detection. Many of these functionalities have been previously discussed. The detailed system architecture is shown in Figure 2.9.

2.5 Semantic Processing of Trajectories

Trajectory data management and processing focus on the low-level performance like indexing, querying and compression; whilst trajectory mining emphasizes on geometric pattern discovery from trajectories, still a bottom-up method driven from data. In contrast, semantic aspect aims at a high-level and top-down method for analyzing trajectories. In this section, we summarize semantic processing of trajectories in terms of three parts, i.e., *ontology based spatio-temporal system*, *conceptual views on spatio-temporal system* and *conceptual views on trajectory*.

2.5.1 Ontology based Spatio-temporal System

The term "ontology" originally comes from the field of philosophy studying the nature of being, existence or reality in general. In computer and information science, it is an explicit specification of a conceptualization [Gru95], consisting of a concept hierarchy and relationships between

[12] `http://dm.cs.uiuc.edu/movemine/`

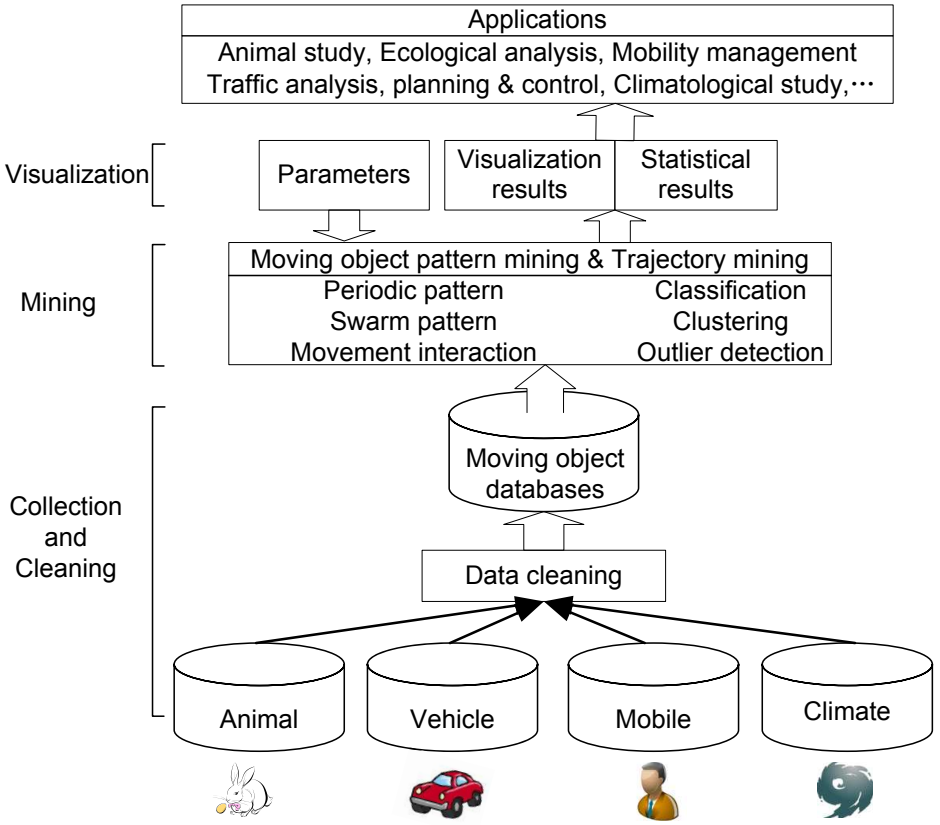


Figure 2.9: MoveMine system architecture [LJL+10]

concepts. The specification of ontology can be very broad, from simple glossaries to complex logical theories. Ontology has been significantly applied in the Semantic Web, in which it can be used to specify standard conceptual vocabularies for data exchange across multiple and heterogeneous data management and application systems.

Spatial ontologies [SCPV04, DP06, GS04] become a major research issue for most semantic-aware GIS (*Geographic Information Systems*) study, especially in some standardization organizations like the Open Geospatial Consortium (2006) and ISO Technical Committee 211 [13], where the objective is to determine a set of formal and sharable concepts of geographic data. Regarding temporal ontology, *instant* and *interval* are the two fundamental concepts, together with a couple of temporal relations (e.g., *begins*, *ends*, *before*, *after*) that need to be specified and determined [HP04].

In addition to these specified ontologies for spatio-temporal data, more formal studies are logic-based reasonings on spatio-temporal systems. *Description logic* (DL) has been proved as a suitable choice for web knowledge because of its open world assumption, and becomes a cornerstone of the Semantic Web in the usage of designing consistent ontologies. The most representative DL languages are OWL-DL and OWL-Lite, which are standardized web ontology languages in the W3C semantic web community [14]. Based on DL, Lutz builds extensions for concrete domains that allow to integrate reasoning the conceptual knowledge with real-world

[13] `http://www.isotc211.org` (Geographic Information/Geometrics)
[14] `http://www.w3.org/2001/sw/` (W3C Semantic Web Activity)

"concrete qualities" such as the age, weight, shape, and temporal extension [Lut02]. The spatial concrete domain in description logics is based on he Randell's $\mathcal{RCC}$ (*Region Connection Calculus*) [RCC92], including eight topological relationships between spatial regions, i.e., disconnected (DC), externally connected (EC), equal (EQ), partially overlapping (PO), tangential proper part (TPP), tangential proper part inverse (TPPi), non-tangential proper part (NTPP), non-tangential proper part inverse (NTPPi). Alternatively, there are several ways for extending description logic with temporal supports [AF00].

Most existing description logic extensions consider the spatial and temporal dimension separately, which is possibly because of its scalability issue of spatio-temporal ontologies in real-world applications. Therefore, some declarative reasoning techniques are proposed for spatio-temporal data, such as STACLP (*Spatio-Temporal Annotated Constraint Logic Programming*) supporting both deductive and inductive reasoning [NRRT04] and MuTACLP (*Multi-theory Temporal Annotated Constraint Logic Programming*) based spatio-temporal reasoning on GIS [RTR02]. Furthermore, Bittner et al. in [BDS09] design a logic-based and top-level spatio-temporal ontology for geographic information integration.

2.5.2 Conceptual Views on Spatio-temporal System

Parent et al. propose a multi-dimensional conceptual data model MADS (*Modeling of Application Data with Spatio-temporal features*) [PSZ06], which provides a thorough and detailed shareable infrastructure for spatio-temporal applications. MADS conceptual model integrates a wide span of topics in temporal and spatial database technologies. MADS can be taken as an object plus relationship spatio-temporal conceptual data model. There are four modeling dimensions in the MADS scheme, namely *thematic data structure*, *spatial dimension*, *temporal dimension*, and the final *multi-representation*. The thematic dimension refers to data structure designed for holding traditional alphanumeric data; the spatial and temporal dimensions are responsible for spatial and temporal information, respectively, both supporting discrete and continuous views; multi-representations are needed for multiple perceptions and resolutions in many real-world spatio-temporal and trajectory applications.

Based on the MADS conceptual model, applications can easily design the spatio-temporal schema. Sotnykova further validates the designed schema by using the reasoning functionality of description logics [Sot06], particularly focusing on data integration issues.

2.5.3 Conceptual Views on Trajectory

From aforementioned discussions, Ontologies or conceptual views for spatio-temporal systems have already explored spatial and temporal semantics. Although these methods provide some support on modeling spatio-temporal dimensions, there are still missing the high-level conceptual model for moving objects and trajectories. Recently, Spaccapietra et al. specifically design a comprehensive conceptual view on trajectory, which aims at exploring semantic application knowledge for modeling trajectory data [SPD+08]. In this conceptual view, trajectory is defined as "*A record of the evolution of the position* (*perceived as a point*) *of an object that is moving in space during a given time interval in order to achieve a given goal*", which can be simply expressed as follows,

$$trajectory : [t_{begin}, t_{end}] \rightarrow space$$

Based on this conceptual view, trajectory can be considered as a sequence of the spatio-temporal path covered by a moving object. In a single trajectory, *stop* and *move* are the two crucial trajectory components. During the temporal extent of *stop*, namely $[t_{beginstop}, t_{endstop}]$, the spatial range of the trajectory is a single point; whilst, during the temporal extent of *move*, namely $[t_{beginmove}, t_{endmove}]$, the spatial range of the trajectory is a spatio-temporal line not a point. Two technical implementations of this conceptual trajectory view are proposed in [SPD+08], i.e., *based on datatypes* and *based on design patterns*. The former is directly to extend the MADS model to cover spatio-temporal dimensions for trajectories; whilst the latter is capable of supporting the potential complexity of trajectory semantics. The conceptual view on trajectories has been significantly adopted in the GeoPKDD project, where many mining techniques have used this stop-move concepts, for example: enriching trajectories with semantic geographical information [ABK+07], a clustering-based approach for discovering interesting places in trajectories [PBKA08], analyzing trajectories using background information [KMOV09], aggregation languages for moving object and places of interest [GKV08]. This thesis further explores semantic trajectory modeling methods, not only providing the high-level trajectory abstraction models, but also discussing the low-level computational issues from real-world mobility tracking data.

2.6 Activity Recognition from Sensor Data

Activity recognition is a very active research topic of many domains, such as web log mining in human-computer interaction, mining activities from GPS and other mobility data, and recently, from geo-social networks. This section reviews some activity recognition studies from mobile sensing data that are related to this thesis.

2.6.1 Activity Recognition from GPS Trajectories

Recently, many studies start to focus on activity inference and annotation using GPS-based trajectory data. The key idea of these approaches is to extract the location (or movement) history of the individual, in conjunction with knowledge about the semantics of the locations (typically from geographic or application data repositories), for inferring the likely higher-level activity of the person. Therefore, the people's trajectory *activity* is typically recognized in terms of identifying the meaningful and significant locations (also called hotspots or points of interest) from their trajectory data.

To identify the important locations from trajectory data, studies like [ABK+07] and [XDZ09] design relevant spatio-temporal join method to infer activities from trajectories, by computing the topological relationships between the trajectory data and a small set of predefined activity hotspots, together with the time constrains. When there are no predefined hotspots available, a clustering method can be used to automatically discover hotspots in trajectory data, e.g., [ZFL+07, PBKA08]. Liao et al. in [LPFK06, LFK07] propose the machine learning and probabilistic reasoning methods to recognize such daily activities from GPS data, in particular

applying the method of conditional random field. Such probabilistic reasoning methods show better performances in detecting and ranking locations, and finding important places. Instead of extracting hotspots individually, researchers start to reconstruct the typical sequence or patterns of trajectory activities (e.g., $hotspot_1 \rightarrow hotspot_2$) in terms of applying sequential mining techniques. For example, Li et al. in [LDH+10] mine periodic behaviors in trajectories, with brief semantics like *home-office*; similarly, Bamis et al. design a new algorithm that can discriminate different activities based on their approximate duration and associated location, for deriving and identifying key human activity patterns [BFS10]. Recently, the research focus shifts from inferring hotspots for individual people to discovering collaborative activities in trajectories of a group of users. The technical focus is on applying reinforcement inference methods, e.g., HITS and PageRank, which are well-used in information retrieval. For instance, Cao et al. in [CCJ10] develop a two-layered graph to model two types of mutual reinforcements (i.e., location-location and location-user) to rank location significance and user authority, for mining significant semantic locations to understand GPS trajectory data. Furthermore, Zheng et al. in [ZZXY10] analyze GPS trajectories with users' comments on their traveled locations via a platform of location-based social networks (e.g., GeoLife [15]), and discover interesting locations and possible activities that can be performed in certain locations. The final objective of this approach is to provide both location recommendation and activity recommendation.

The previous activity recognition study is on the location part of trajectory data, which is about "*what they move for*". Another very interesting study in the literature is the inference of the transportation modes from GPS alike trajectory data, to understand the activity about "*how they move*". For example, researchers design rule based and fuzzy methods to determine the transportation mode from GPS data. For example, Schuessler et al. in [SA09b] use some fuzzy variables (e.g., *median speed*, *95 percentile acceleration*, *95 percentile speed*) and design a set of fuzzy rules to determine the transportation mode. A simple rule is: *when both the 95 percentile acceleration and the 95 percentile speed are high, the transportation mode is car.* However, it is non-trivial to correctly provide such rules, which requires more domain prior knowledge. To avoid manually designing rules, researchers start to apply automatic mining and learning methods on a large number of segment features. Following this direction, Zheng et al. in [ZCL+10] design a three-step framework to automatically infer the means of transportation from GPS trajectories: firstly, apply a simple trajectory segmentation algorithm by identifying change-points; secondly, compute features and build a widely-used inference model like decision tree and support vector machine; thirdly, apply graph-based post processing to refine the results. In their study, the authors consider four types of transportation modes, i.e., walk, drive, bus, bike. Similarly, Reddy et al. in [RMB+10] use mobile phones to determine transportation modes, while the focus is on using different kinds of phone sensors, e.g., WiFi, accelerometer rather than only GPS data. More details of using multiple phone sensors will be discussed in Section 2.6.3.

[15] `http://research.microsoft.com/en-us/projects/geolife/`

2.6.2 Activity Recognition from Accelerometer Data

A triaxial accelerometer is a sensor that can collect a real valued estimate of acceleration along three axes, i.e., x, y and z. Supervised learning from accelerometer data has been largely studied in activity recognition and other sensor data analysis domain. The research has significantly focused on feature extraction and classification of representative movements (e.g., walking, running, climbing steps, using gym instruments etc.) using data samples from accelerometers. For example, as a well-cited work, Bao et al. in [BI04] are the first use of multiple accelerometer sensors worn on different parts of the human body (e.g., hip, wrist, upper arm, ankle, thigh) to detect common activities such as sitting, standing, walking, and running (called "micro-activity" in this thesis). The main drawback of such activity recognition studies using wearable sensors is their limitation to laboratory conditions, which is not fully applicable to naturalized settings. In addition, multiple accelerometer sensors are required to achieve better performance.

Regarding more naturalized or semi-naturalized setting, researchers start to build accurate prediction of an individual's locomotive state using a single accelerometer sensor [RDML05]. In such cases, the training data is also obtained by fixing the accelerometer to a predefined part of the body. Moreover, the techniques assume a *priori* knowledge of the duration (start and end times) of each micro-activity, and are thus concerned solely with classifying a specific instance of unknown micro-activity. The research focus is on finding rich accelerometer features, both in time domain (e.g., mean, variance of the triaxial accelerometer values) and frequency domain (e.g., energy computed by FFT - Fast Fourier Transform). In Chapter 7, we will extensively study these features to build algorithms for micro-activity recognition with high accuracy.

In addition to recognizing micro-activity from accelerometer data, inferring high-level activities (e.g., *home-cooking*) and patterns (e.g., first *home-cooking* and then *home-dinner*) is a very interesting and complicated research topic, because of the inherent challenges like: (1) the noisy nature of accelerometer data input in naturalized setting (e.g., accelerometer data can be totally different between walking with phone at hand and with phone in pocket); (2) the concurrent and interleaved nature of complex activities. Therefore, a couple of rich statistical inference methods like Hidden Markov Model, Conditional Random Field [VLL+10], topic models [HFS08], and emerging patterns [GWT+09] are applied for learning high-level activities from micro-activities.

2.6.3 Activity Recognition from Multiple Phone Sensors

Recently, the smartphone has become the central sensing device for people's daily life, which can continuously capture people's location, motion trace, interaction, communication, and other social interactions. Reality mining [EP06] and people sensing [CEL+08][LML+10] constitute one of the most exciting research frontiers in computational social science [LPA+09]. Reality mining studies human social behavior by using wireless devices like smartphones (with embedded GPS, WiFi sensors etc.), to understand what people do, where they go, and how they interact with each other. For example, Gonzalez et al. study human mobility patterns by analyzing the call data [GHB08]. Along these directions, activity recognition using phones becomes a very active and promising area where the main thrust is to explore multiple sensor data (and even real-time sensing streams) associated with the phone to recognize user's activity and context [CBC+08, MLF+08, LYL+10, CD10]. The primary focus of these studies is on discovering how

a combination of phone sensors can help improving the ability of semantic activity detection; it is natural but needs to be further validated that the ability to recognize heterogeneous semantic activities would increase with additional sensors. For example, Choudhury et al. in [CBC+08] demonstrate that the combination of accelerometers and microphones can provide good features for activity recognition (e.g., walk, run, talk, cook, eat etc). The group of Campbell develop several activity recognition prototypes in smartphones, e.g., CenceMe [MLF+08] and Jigsaw [LYL+10] for 'on-phone' classification to detect events with multiple phone sensors (GPS, accelerometer, microphone).

Recently, a very promising data campaign is finished which is organized by Nokia Research Center at Lausanne [KBD+10]. The campaign targets richer data collection on large-scale and various participants (nearly 200 users including EPFL students, researchers, family, friends etc.) via Nokia N95 over nearly two years, consisting of almost all-known smartphone embedded sensors and data records, like GPS, GSM cell tower, call and SMS logs, WiFi, Bluetooth, accelerometer, audio features. Such rich data can provide new opportunities for future works on mining people activities, as well as the study of privacy issues.

2.7 Summary

This chapter reviewed the background and related works as the context of this thesis study. We investigated these studies in terms of five main aspects. Based on such investigation, we can claim that mobility data analysis and understanding has become a very attractive topic in many domains related to the emerging computational social science. The first three aspects of the literature (i.e., trajectory data management, data processing, and mining) are mainly focusing on the perspective of "geometric" data analysis; whilst the latter two aspects (i.e., semantic aspect and activity recognition) study on the high-level movement behaviors for understanding mobility data. There is an obvious gap between low-level geometric data management & processing and high-level mobility modeling & understanding. Therefore, the main objective of this thesis is to study the combination of top-down semantic mobility modeling and bottom-up data computation, for filling the gaps and building semantic trajectory methods to better understand mobility data.

Chapter 3

Semantic Trajectory Modeling

For the scientist, a model is a way in which the human thought processes can be amplified.

C. West Churchman, 1968

3.1 Introduction

In this chapter, we present the first main contribution of this thesis, i.e., *semantic trajectory modeling*. The major objective of trajectory modeling is to provide a comprehensive mobility data representation, which can clarify requirements and define models to support the mobility analysis of moving objects. This thesis defines two types of models: *an ontological trajectory framework* and *a hybrid trajectory model*. The ontological trajectory framework provides a modular ontology for structuring, modeling, and querying trajectories; whilst the hybrid trajectory model offers different levels of data abstractions for mobility data understanding. The first model focuses on querying and reasoning on high-level trajectory semantics; whilst the second model serves the data representation for different levels of trajectory computing.

This chapter is organized as follows: Section 3.2 investigates the detailed requirements of modeling trajectory and mobility data, based on the study of several different trajectory scenarios with real-life GPS tracks; Section 3.3 provides the modular trajectory ontologies; a case study for utilizing trajectory ontologies is discussed in Section 3.4; Section 3.5 presents the hybrid trajectory model; finally, Section 3.6 summarizes this chapter and further compares the two different trajectory models.

3.2 Modeling Requirements

In this section, we analyze the requirements of modeling trajectory data. Firstly, several trajectory scenarios are investigated with corresponding GPS tracking datasets that we have studied; afterward, two different types of trajectory modeling requirements are discussed, namely *spatio-temporal knowledge* and *semantic knowledge* for trajectories.

3.2.1 Trajectory Scenarios

From the background introduction in Chapter 1, we observe that the world is now full of trajectory data, which is induced by the advance and the increment of GPS-enabled handsets and sensor-tracking devices. In our trajectory study, we focus on three different kinds of important trajectory scenarios that have a large amount of real-life GPS alike moblity datasets.

The first scenario is ***vehicle tracking***. Regarding vehicle movement scenario, we analyze two large datasets: more than 2 million GPS records of 17,241 Milan cars tracked for one week from GeoPKDD[1] - an European FP6 project, and 2 Lausanne taxis GPS records (about 3.3M) in 5 months from Swisscom[2]. In addition, we study some public datasets, like trucks and buses in Athens from the R-tree portal[3]. For these datasets of vehicle trajectories, we observe the following characteristics,

1) When the vehicle is not moving for a very long time duration, the GPS devices could possibly automatically stop recording or be manually turned off by drivers, which can help us find the important breaking points and identify trajectories from the original long sequential tracking data. Each sequence between two consecutive breaking point (without any unreasonable big GPS recording gaps in the middle) can be considered as a single trajectory.

2) For each single trajectory, the GPS tracks are usually continuous that have small and stationary gaps between every two neighboring GPS records, e.g., 30 seconds in Lausanne taxi data and a few minutes in Milan car data. Thanks to such high-frequency in GPS recording, we can approximately calculate instant speeds and use velocity-based algorithms for splitting trajectories and identifying meaningful trajectory subparts, e.g., a fast move episode, a slow move episode, or a complete stop episode.

3) For vehicle movement, the trajectory is usually constrained by some underlying network infrastructures, e.g., car in the high-way, bus in the scheduled path, which means the position of a trajectory point should be limited in the network. Therefore, analyzing such network-constrained trajectory data can design and apply additional map-matching alike algorithms, e.g., calibrating GPS data to clean the raw data and reduce errors.

The second scenario is ***people movement***. Thanks to the GPS-embedded smartphone techniques, people daily movement data can be easily collected, which is much more efficient than traditional questionnaire based survey methods for mobility analysis. We study a very large smartphone dataset provided by Nokia Research Center - Lausanne[4] [KBD+10]. They distributed smartphones (Nokia N95) to students/researchers at Lausanne, collecting several *people sensing* data including GPS feeds. We analyze 185 users, with 23,188 daily trajectories with 7.3M GPS records in total. Recently, there is also a publicly available smartphone dataset about people trajectories collected by Microsoft Research Asia in their GeoLife project, which tracked 165 users in a period of over two years (from April 2007 to August 2009)[5]. In addition to

[1] `http://www.geopkdd.eu/`
[2] `http://www.swisscom.ch/`
[3] `http://www.rtreeportal.org/`
[4] `http://research.nokia.com/locations`
[5] `http://research.microsoft.com/en-us/projects/geolife/default.aspx`

holding the basic spatio-temporal information (e.g., $\langle x, y, t \rangle$ GPS records) like vehicle tracking data, people trajectory data is far more *heterogeneous* because of a couple of distinguished features as follows,

1) People can choose different transportation modes in their different daily trajectories or even in one single day trajectory. For example, people can choose bus, car, bicycle, train, plane for their journey. Of course people walk, and walking is also the main transition between taking two consecutive transportation modes, e.g., *for changing the transportation mode, John moves from a bus stop to a train station.* Therefore, when analyzing people trajectory data, we need to identify these different transportation modes, which recently also becomes an important and hot topic.

2) Compared with vehicle trajectories, people tracking data is usually more noisy. There are a couple of reasons: (1) People trajectories are usually collected by smartphones with very limited power capacity, therefore the GPS sampling frequency is really much lower than vehicle ones; (2) Vehicle trajectories are usually from outdoor activities, whilst a large amount of people trajectories are generated during indoor movement. Most indoor movement data cannot be captured by GPS-alike sensing devices because of the signal-loss; (3) As discussed, people can take different transportation modes, which could in turn cause non-stationary GPS sampling records.

3) Privacy of people mobility is a very challenging topic. Analyzing people trajectories is far more sensitive and complex than the study of vehicle or animal trajectories that we will describe subsequently. This thesis focuses on building semantic trajectories of people movement data for better understanding people's movement behaviors, while privacy is beyond the scope of this thesis.

The third scenario is ***biological trajectory***, where the tracked moving objects are neither vehicles nor people but biological entities – e.g., animals and flocks. For this kind of trajectory scenario, the most representative example is bird migration, which is the regular seasonal journey undertaken by many species of birds such as white storks (*Ciconia*). For example, they leave the north hemisphere (e.g., Europe) and migrate south (e.g., Africa) in each autumn; and vice versa in each spring. In addition, we also study some biological trajectories generated by animals, such as 6 years tracking data on Bornean orangutans. Some distinguished observations on these biological trajectories include:

1) The GPS recording frequency in biological trajectory is usually much lower compared to the previous vehicle and people trajectory scenarios. For instance, in our datasets, the positioning of monkeys is recorded at the interval of 30 minutes; whilst for the bird migration data, the interval is even higher, say a few hours or even a couple of days. Therefore, in this case, the velocity of moving object is not as informative as the vehicle ones. Therefore, in the trajectory computing chapters, we design density based algorithms for understanding this type of trajectories, rather than the velocity based methods.

2) For trajectory analysis on animal movement, there might be no direct need to incorporate strict network constraints. Therefore, map-matching is not always meaningful in this scenario. Nevertheless, some sort of fuzzy networks may still restrain animal movements in the wild, such as habitat structure or social competition with neighboring groups.

3) For biological trajectory, many species usually move together, and even keep in a stable *formation* (i.e., the shape of bird flocks like the V-formation) when they move in company with each other. Actually, we have this observation not only in the flock data but also in the monkey data. Therefore, movement in group is another important observation in biological trajectory data, and many works study such kind of flock patterns.

All of these three types of trajectory scenarios are movement as change of spatial location, which means trajectory is a evolution of the physical position (perceived as a $\langle x, y \rangle$ *point* in 2D space) of a moving object. In real-life, there is another interesting type of trajectory scenarios where the movement is the change of non-spatial features, such as career trajectory (e.g., *a researcher from PhD student, to Postdoc, to Professor in her/his academic career*). This kind of trajectory data is about movement in abstract space (in the previous example, the abstract space is a set of possible *research positions*). We call this type of trajectories "metaphorical trajectory", while the previous scenarios discussed are "point-based trajectories". In this thesis, we study the point-based trajectory data in terms of spatial change, not the metaphorical ones in abstract space.

3.2.2 Spatio-Temporal Knowledge

As mentioned, this thesis focuses on the point-based trajectory that tracks the location evolution of moving objects in physical space, not the metaphorical trajectory. For such point-based trajectory, many literatures define it as "*a record of the evolution of the position* (*perceived as a point*) *of an object that is moving in space during a given time interval*" [SPD+08]. It is a function from a temporal domain to a range of spatial values, i.e., $trajectory : [t_{begin}, t_{end}] \rightarrow space$. In this trajectory's time-space function, the time domain is restricted in the duration between the beginning time (t_{begin}) and the ending time (t_{end}), not necessarily the complete lifespan of the moving object that generates this trajectory. For the space extent, it can be unlimited free 2D space (e.g., $[longitude, latitude]$) or 3D space (e.g., $[longitude, latitude, altitude]$), or the network constrained space (i.e., points can only be in the road segments or crossings).

From such time-space function, we only observe some basic spatial and temporal information without any additional semantics. Therefore, the fundamental requirement of modeling trajectories generated by moving objects is the *spatio-temporal knowledge*, which captures the generic spatial and temporal information during objects' movement. With spatio-temporal knowledge of trajectories, we can answer the basic trajectory queries like *range query* in the moving object databases, e.g., "*Return all trajectories that have passed a given geometric area determined by a bounding box* $[x_1y_1, x_1y_2, x_2y_2, x_2y_1, x_1y_1]$".

Such movement data is usually collected by GPS-alike positioning devices the object bears, and is stored as raw data from which trajectory information can be computed. This raw data

usually consists of a sequence of raw sptaio-temporal points $\langle x, y, t \rangle$, and bears no further semantics except for the identification of the moving object it comes from $\langle id, x, y, t \rangle$. For example, the traffic management application in our case studies relies on a number of raw data files, where each file records the position of a given car over a period of time. Raw data has first to be cleaned to correct acquisition errors and to interpolate missing positions due to malfunctioning of the devices or of the transmissions. Cleaned spatio-temporal data represents the movement of the associated object. To "upgrade" this data into trajectories we firstly need to split long raw tracking data into separate trajectories and identify the time duration $[t_{begin}, t_{end}]$ in trajectory's time-space function. In other words, we need to identify the *beginning* point and the *ending* point of a trajectory. Afterwards, automatic trajectory segmentation algorithms can be defined, based for example on an analysis of speed of the moving object and of the duration of non-moving periods. Frequently, however, application directives are needed to separate for example a non-meaningful stop (e.g., a car stopping at traffic lights) from a meaningful one (e.g., the car stopping to serve some application purpose). The output of segmentation procedures consists in the trajectory path, with the associated stops and derived moves. All of these trajectory computing tasks analyze the generic *spatio-temporal knowledge* of the trajectory, which only focus on the geometric locations of movements, ignoring their underlying semantics.

3.2.3 Semantic Knowledge

To better understand mobility data, applications need to apprehend the semantics of the trajectory, therefore additional information and processes are required. Most frequently, the first additional task is to turn the geometric line representing the trajectory path into a meaningful geographical trace. This means, for example, turning a move from a spatio-temporal point $\langle x_1, y_1, t_1 \rangle$ to another spatio-temporal point $\langle x_2, y_2, t_2 \rangle$ into a move from $\langle Lausanne, t_1 \rangle$ to $\langle Zurich, t_2 \rangle$. With such transformation, geometric points $\langle x, y \rangle$ are replaced by more meaningful geographic locations. The transformed representation makes the semantics of the move more explicit to users of an application involving tracking and analyzing moves between cities. To enable such information upgrade, knowledge about the relevant geography is needed. What exactly is relevant geography depends on the application perspective, in particular the application requirements in terms of granularity (are city names enough or shall the information be at the street level of detail?). Therefore, even if pre-existing ontologies are imported into the system, this geographic component necessarily is application-dependent, as it has to contain the geographic elements relevant to the application. For example, a city traffic management application obviously needs knowledge about the streets, but may also use more detailed information such as position of traffic lights, one way streets, gas stations, etc. This forms what we call the *geographic knowledge* of the trajectory, the first kind of *semantic knowledge*.

Thanks to this *geographic knowledge*, trajectory applications become possible to answer more meaningful and useful trajectory queries like: "*Return identity of cars which stopped today at a gas station*". In such case, many public geographic dataset like Google Map[6], Openstreetmap[7] can be used to enrich trajectory semantics.

[6] `http://maps.google.com/`

[7] `http://www.openstreetmap.org/`

In addition to the geographic knowledge, the understanding and management of trajectories heavily relies on their relationships to application objects (not just the moving object). For example, traffic analyses based on average travel time computed from cars' trajectories would not be really meaningful without knowing what events (e.g., football matches, pop concerts, road works) within the city may have influenced the fluidity of the traffic. The semantic representation of a trajectory via its link to application objects can also be used to better clean the initial data and to support more sophisticated methods that may significantly reduce the size of the trajectory dataset. Another example is that semantic stops of delivery trucks are only allowed if within or nearby the premises of company customers. Therefore, trajectories can be semantically enriched with the data about the application domain. This forms what we call the *application domain knowledge* of the trajectory, the second kind of *semantic knowledge*. The semantic enrichment due to application knowledge may concern the moving object itself, or any other objects in the application database.

With such domain knowledge, trajectory applications may allow answering more complex and semantically interesting trajectory queries, e.g.: "*How many persons in our lab traveled yesterday afternoon from home to office by bus?*, "*How many delivery trucks delivered goods this morning to any one of our major customers in Switzerland?*".

In summary, we can claim that a comprehensive trajectory model should support different views of knowledge for explaining object movements. Spatio-temporal knowledge is the only geometric view of trajectories, which focuses on the evolving geometric position where a moving object temporally resides; whilst geographical and application domain knowledges are the semantic view of movement that can provide the explanation of high-level knowledge for better understanding of movement behaviors. In the later sections of this chapter, we present the two different models that are designed in this thesis to best meet these modeling requirements: one is *the ontological framework* and the other is *the hybrid spatio-semantic model*.

3.3 Trajectory Ontologies

This section proposes an ontological infrastructure that addresses the modeling requirements identified in the previous section with the goal of supporting creation, management and analysis of trajectory data. As shown in Figure 3.1, the needed ontological infrastructure is composed of three ontology modules[8]: *the geometric trajectory module*, *the geography module*, and *the application-domain module*. The benefit of using a modular structure, rather than one all-encompassing ontology, mainly is easier design and maintenance, together with opportunities for query optimization.

The geometric trajectory module holds generic concepts for the description of the geometric component of a trajectory. It includes spatio-temporal concepts used to specify spatial, temporal and spatio-temporal features needed for a full description of application data. This enables to state, for example, that the spatial extents of buildings and cities are represented as points and areas, respectively. Similarly, addresses can be given an associated timeframe specifying the period where each specific address is valid. Regarding trajectories, the needed concepts include

[8] A module is a sub-ontology of a larger ontology [PSS08]

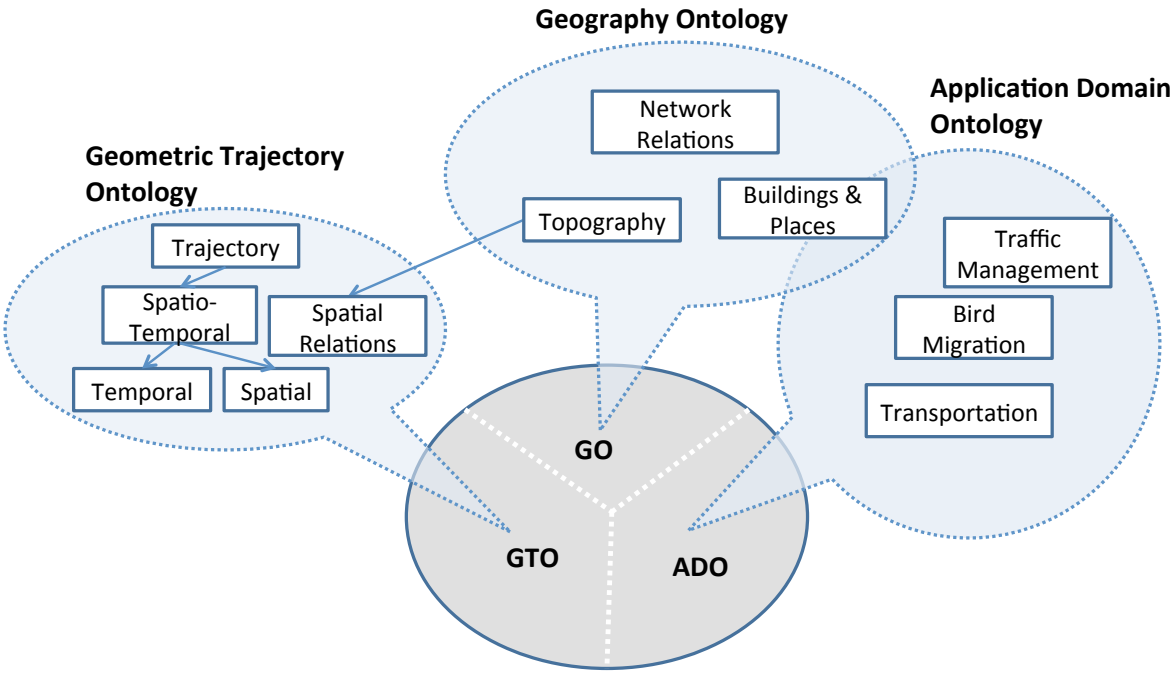


Figure 3.1: Ontological infrastructure for modeling trajectories

those used to specify trajectory paths as spatio-temporal lines formed by moving points during a trajectory, as well as those used to structure each trajectory path into a sequence of episodes (e.g., stops and moves). How stops in a specific trajectory are identified depends on application rules. This is described by establishing the application-specific links between the geometric module and the other modules.

The geography module holds the concepts that participate into the application description of land coverage. Concepts are likely to include those describing the topography of the land, natural (e.g., hydrological) and artificial (e.g., routes, trains, facilities) networks, buildings, landmarks, vegetation and anything else that is of interest to the application. This module is tightly interrelated with the geometric trajectory module as each application element that has a spatial connotation is to be linked to the type of geometry that is used by the application to specify the corresponding spatial extent. The module is also tightly interrelated (or overlapping) with the application domain module as its concepts may also have a thematic description providing application information beyond the geographic and geometric facets. For example, concepts about buildings and places may include standard classification schemes defined in the geography module plus other features specific to the application domain.

The application domain module gathers all application dependent concepts. Examples of such modules include traffic management ontologies, bird migration ontologies, and transportation ontologies. These ontologies can provide more meaningful knowledge of certain applications.

Ontological knowledge is usually partitioned into two levels of abstraction. The TBox holds abstract descriptions of the concepts, their (data type) properties and the links (roles). The ABox holds instances of the elements in the TBox. This is similar to the schema (metadata) and data duality in databases. Applications need both levels of knowledge. Our approach builds on the increasingly popular hypothesis that the ABox (all instances) is stored in a database.

Moreover, we assume that also the TBox is stored in the database, as allowed by recent advances in semantic data management technology.

The following subsections detailedly describe the proposed modules, including their inner-structures and their inter-relationships.

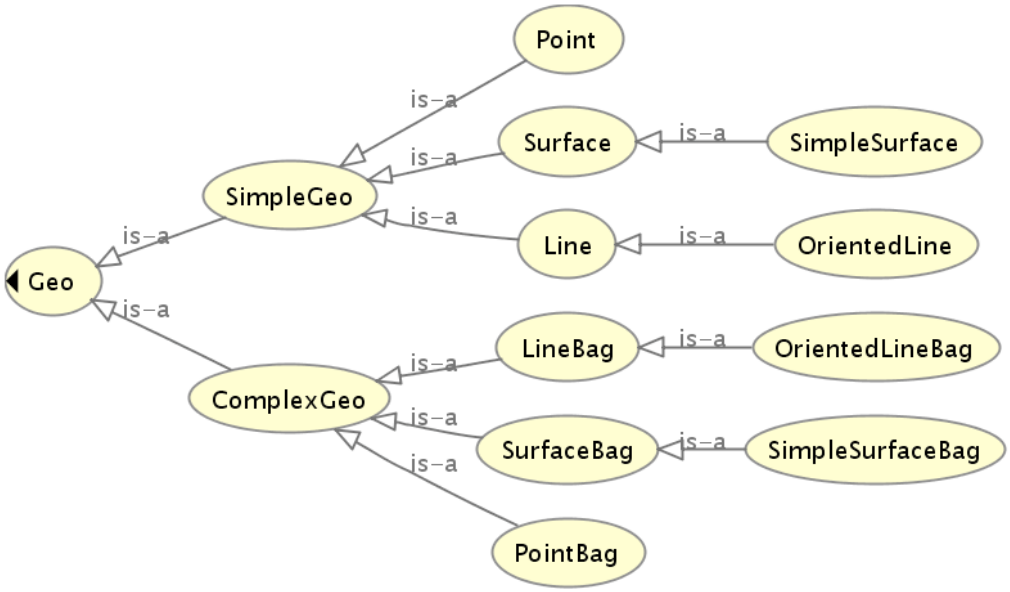
3.3.1 Geometric Trajectory Ontology

The geometric module may itself be seen as composed by several dedicated ontologies, namely the *Spatial Ontology*, *Temporal Ontology*, *Spatiotemporal Ontology*, and *Trajectory Ontology*. As illustrated in Figure 3.2, the spatial and temporal ontologies contain the concepts that are traditionally used for the description of spatial and temporal features. The basic concepts in Figure 3.2 are self-explanatory. The *SimpleGeo* concept generalizes any kind of simple spatial features, like points, lines, and surfaces. The *ComplexGeo* concept generalizes the concepts denoting bags of points, lines, and surfaces, and also denotes any heterogeneous complex extent. Similarly for the *SimpleTime* and *ComplexTime* concepts. The concept terminologies in the figure are adopted from MADS specifications [PSZ06]. For space, similar but not identical concepts have been proposed by the Open Geospatial Consortium [OGC06] and ISO Technical Committee 211 (Geographic Information/Geomatics)[9]. These concepts are expected to become universal standards and whenever that happens this dedicated ontology needs to be matched, as concepts may be replaced by standard data types. The same is on its way for the temporal domain (see e.g., [OS95] and the recent standard for SQL with temporal features).

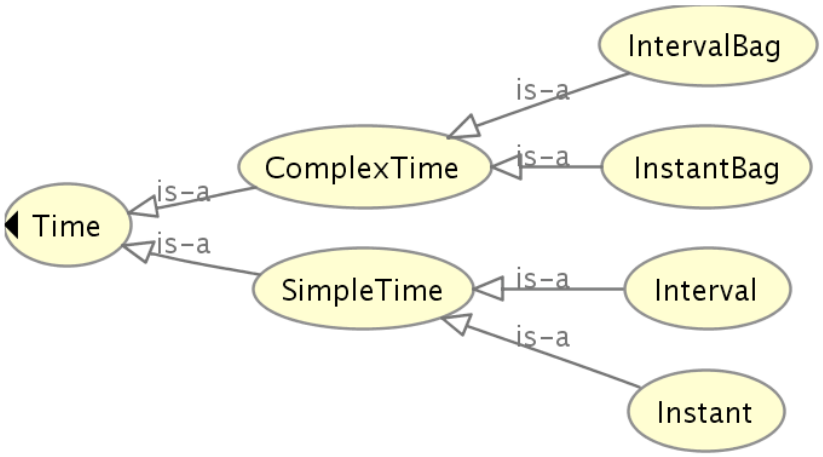
The *Spatiotemporal Ontology* builds on the spatial and temporal ontologies to provide concepts for describing phenomena with time-varying geometry. For example, to track the movement of a car, the car geometry (i.e., its spatial extent) is described using the concept of time-varying point, i.e., a point whose $\langle x, y \rangle$ position changes over time. While movement is continuous in real life, its data record usually consists in a sequence of timestamped points (i.e., $\langle x, y, t \rangle$ points) and interpolation functions as needed to compute the $\langle x, y \rangle$ position given any time t_1 not necessarily stored in a $\langle x, y, t \rangle$ triple. Abstracting from interpolation aspects, the time-varying geometry can be defined in description logic (DL) as a concept with at least one instance of the *listOf* role that links it to *TimePoint* instances (representing the $\langle x, y, t \rangle$ triples). Similarly for moving lines, moving surfaces and moving objects of undefined spatial type (time-varying Geo geometry). See Table 3.1 for the formal definitions. For a full and rigorous discussion about the definition of spatio-temporal concepts the reader can refer to the foundational work of Güting on moving object databases [EGSV99] [GBE+00] and recent work on trajectory databases [KO07].

Finally, the *Trajectory Ontology* holds the concepts that participate in the description of how movement can be understood as a set of structured trajectories. The needed concepts have been identified and discussed in our previous work [SPD+08]. They include the already mentioned *stop* and *move* concepts, which define a segmentation of the trajectory. They also include the *Begin* and *End* concepts denoting the $\langle x, y, t \rangle$ triple that identifies where and when the trajectory starts and the $\langle x, y, t \rangle$ triple identifying the end of the trajectory. We also include the B.E.S. concept that was introduced to generalize common features of the begin, end, and stop concepts. This set of concepts allowed us to define a design pattern for structured trajectories and supports

[9] `http://www.isotc211.org/`



(a) Spatial ontology



(b) Temporal ontology

Figure 3.2: Example concept hierarchies in spatial and temporal ontologies

Table 3.1: Definition of spatiotemporal ontology

Concept	Definiton in Description Logic
$TimePoint$	$\sqsubseteq \exists$ hasGeometry.Point $\sqcap \exists$ hasTime.Instant
$TimeVaryingPoint$	$\sqsubseteq \exists$ listOf.TimePoint
$TimeLine$	$\sqsubseteq \exists$ hasGeometry.Line $\sqcap \exists$ hasTime.Instant
$TimeVaryingLine$	$\sqsubseteq \exists$ listOf.TimeLine
$TimeSurface$	$\sqsubseteq \exists$ hasGeometry.Surface $\sqcap \exists$ hasTime.Instant
$TimeVaryingSurface$	$\sqsubseteq \exists$ listOf.TimeSurface
$TimeGeo$	$\equiv$ TimePoint $\sqcup$ TimeLine $\sqcup$ TimeSurface
$TimeVaryingGeo$	$\equiv$ TimeVaryingPoint $\sqcup$ TimeVaryingLine $\sqcup$ TimeVaryingSurface

linking trajectory elements to application elements, thus allowing semantic enrichment on the trajectory data for providing better understanding.

Table 3.2 shows the formal definition of trajectory and its related concepts. These definitions relate to moving point trajectories and consider a point type geometry for *begin*, *end*, and *stops*. From the temporal perspective, *begin* and *end* are associated with an instant time type, while *stops*, which may have a temporal duration, are associated with a temporal interval time type. *Moves* are a non-stop section of a trajectory, leading from one *stop* to the next *stop*, with the exception of the first move that starts at *Begin* and the last move that ends at *End*. Each *trajectory* has exactly one *Begin*, exactly one *End*, one or more *Move*, and 0 or more *Stop*. Obviously the number of *Stop* determines the number of *Move*.

Table 3.2: Definition of trajectory ontology

Concept	Definition in Description Logic
Begin	$\sqsubseteq$ TimePoint
End	$\sqsubseteq$ TimePoint
Stop	$\sqsubseteq \exists$ hasGeometry.Point $\sqcap \exists$ hasTime.Interval
B.E.S	$\equiv$ Begin $\sqcup$ End $\sqcup$ Stop
Move	$\sqsubseteq$ TimeVaryingPoint $\sqcap \exists$ from.B.E.S $\sqcap \exists$ to.B.E.S
Trajectory	$\equiv \exists$ hasBegin.Begin $\sqcap =1$ hasBegin $\sqcap \exists$ hasEnd.End $\sqcap =1$ hasEnd $\sqcap \exists$ hasMove.Move $\sqcap \forall$ hasStop.Stop

3.3.2 Geography Ontology

This ontology holds concepts about the geographical environment, both generic and application dependent [MEB+99, Rei05]. To build this ontology existing specifications and standards can be reused. The *World Wide Web Consortium* (W3C)[10], and the *Ordnance Survey* (OS)[11], for example, have devised some ontologies for describing geographical concepts based on the Open Geospatial Consortium [OGC06] standards. For example, a domain ontology *BuildingsAndPlaces* contains concepts and properties describing human constructions in geographical space, such as banks, churches, bowling clubs, etc. Typical properties in this context are *hasName*, *hasHistoricInterest*, *hasFootprint*, etc. Rather than designing a new geography ontology from scratch, importing an existing ontology or a part of it provides a first version that can then be manually augmented with extra concepts while deleting concepts that are irrelevant to the application at hand.

3.3.3 Application Domain Ontology

An application domain ontology holds the wide range of concepts that make up the *universe of discourse* for a set of applications in the same domain. Domain ontologies have been intensively investigated in the literature. Examples in the traffic management domain include the *Ontologies of Transportation Networks* (OTN) [LOY05], *Towntology*[12], and *Spatially Aware Information*

[10] http://www.w3.org/2005/Incubator/geo/XGR-geo-ont-20071023/#crs/
[11] http://www.ordnancesurvey.co.uk/oswebsite/ontology/
[12] http://www.towntology.net/

Retrieval on the Internet (SPIRIT)[13]. These approaches do not make an explicit separation between the *Geography Ontology* and the *Traffic Management Ontology*, but focus on one of them or combine the two. While the ultimate result may be the same, the initial separation of concerns provided by a modular approach promotes reusability and understandability. Separation of concerns does not entail disjointedness. We can assume the geography and application domain ontologies may overlap in some applications. This thesis mainly focuses on the geography knowledge which can be easily extracted from many free map data, e.g., openstreetmap., and additionally apply some available application knowledge like *office*, *home* which can be inferred or provided by additional sources.

Table 3.3 illustrates some axioms for a traffic management application. The first axiom defines a *HomeOfficeTrajectory* as a trajectory that starts at home and ends at office. The third axiom defines a *BlockedStreetT* as a street where there is some long term road work ongoing. The fourth axiom describes the concept of a high congestion trajectory as a trajectory that has more than 10 stops with a long time interval.

Table 3.3: Definition of traffic management ontology

Concept	Definition in Description Logic
$HomeOfficeTrajectory$	$\equiv$ Trajectory $\sqcap$ $\exists$hasBegin.$\exists$isLocatedIn.Home $\sqcap$ $\exists$hasEnd.$\exists$isLocatedIn.Work
$LongTermRoadWork$	$\sqsubseteq$ RoadWork
$BlockedStreetT$	$\equiv$ StreetT $\sqcap$ $\exists$ hasRoadWork.LongTermRoadWork
$HighCongestionTrajectory$	$\equiv$ Trajectory $\sqcap$ $\exists_{>10}$ hasStop. $\exists$ hasTime.LongTimeInterval
$LongTimeInterval$	$\sqsubseteq$ Interval

3.3.4 The Complete Modular Ontology

Combining the *Geometric Trajectory Ontology*, *Geography Ontology* and *Traffic Management Ontology* ontologies together leads to the final overall *Semantic Trajectory Ontology*. This final ontology provides the full semantic description of application-relevant trajectories with their domain specific semantic meaning. Figure 3.3 shows the final semantic trajectory ontology for a traffic management application. For the sake of clarity, the figure shows a very partial version of this ontology with only the most important concepts and relationships.

3.4 Case Study on Trajectory Ontologies

We test our trajectory ontological modeling approach using a real case study in the traffic management domain. In this application scenario, information about trajectories of cars equipped with GPS devices are collected and stored in a relational database. Car trajectories are used to evaluate traffic flows on some specific predefined paths. To this end, a classification of trajectories according to their level of congestion helps analysts to identify possible traffic problems.

[13] `http://www.geo-spirit.org/`

Figure 3.3: Semantic trajectory ontology for a traffic management application

Last but not least, information about events (e.g., football match) occurring within the city is used to possibly explain traffic problems.

As part of our approach, we used a commercial relational DBMS, equipped with support for spatial and ontological data, to store the semantic trajectory ontology with its instances into a database and to make it easily available for querying. Indeed, while ontologies are the best candidates for realizing the conceptual layer, relational DBMSs are natural candidates for the management of the data layer due to their efficiency in managing very large quantities of data.

To implement the ontological framework we have discussed, we considered the two currently available approaches for implementing ontologies. The first one relies on the ontology editor tools (e.g., Protégé[14] – a free, open source ontology editor and knowledge-base framework) and reasoners (e.g., Racer[15], Pellet[16]) that have been developed for ontology definition and management. These DL-based systems provide typical reasoning services and dedicated query languages, e.g., SPARQL, OWL-QL. A known limitation of this approach is that current reasoners perform well only for TBoxes and ABoxes of limited size. They cannot offer affordable performance for trajectory data sets whose size tends to be ever-growing (even when using techniques to reduce the volume of data describing each trajectory).

We therefore decided to adopt the alternative approach that relies on database technology enriched with reasoning capabilities. Database technology supports huge data sets and query optimization techniques. A representative example of this approach is the recent semantic extension of Oracle's DBMS[17], which supports OWL-Prime, a subset of OWL-DL, and logical rules [ora07]. Oracle semantic technologies store ontologies, represented as RDF triples, in relational tables. Both defined and inferred knowledge is stored, for the TBox as well as for the ABox, which results in faster processing of queries as all inferences are readily available. Reasoning includes traditional inferencing (classification, satisfiability, subsumption, and instance checking). Reasoning services are executed on demand. For example, the *create_entailment* function is available for computing inferences.

For our case study we created the ontological knowledge using Oracle's OWLPrime formalism. OWLPrime specifications include support for:

- Basics concepts: *class, subclass, property, subproperty, domain, range, type*
- Property characteristics: *transitive, symmetric, functional, inverse functional, inverse*
- Class comparison operators: *equivalence, disjointness*
- Property comparison operators: *equivalence*
- Individual comparison operators: *same, different*
- Class expressions: *complement*
- Property restrictions: *hasValue, someValuesFrom, allValuesFrom*

[14] `http://protege.stanford.edu/`
[15] `http://www.sts.tu-harburg.de/~r.f.moeller/racer/`
[16] `http://clarkparsia.com/pellet/`
[17] `http://www.oracle.com/technology/tech/semantic_technologies/`

OWLPrime is a subset of OWL-DL. Limitations in the expressive power of OWLPrime stem from the pragmatic concern about implementability without excessive complexity. Features that would lead to having to manage incomplete information (unknown individuals) and uncertain reasoning (open alternatives for reasoning paths) have been left out. For example, let us assume that R denotes a role and C $\sqsubseteq \exists$ R is an axiom of the TBox. Each time a new individual x is inserted in C, C(x), the reasoner should generate an unknown individual to which x is linked by R: R(x, ?). As another example, let us now assume that C $\sqsubseteq$ A $\sqcup$ B is another axiom of the TBox. Each time a new individual x is inserted in C, C(x), the reasoner infers that x should also be inserted in A or B, or in both: It cannot know in which one.

In DL terms, this means that intersection ($\sqcap$) and value restriction ($\forall$) are not accepted in the left hand side of inclusion axioms, and existential quantification ($\exists$) and union ($\sqcup$) are not accepted in the right hand side of inclusion axioms. OWLPrime also does not support the use of cardinality in property restrictions. Finally, set operations (union, intersection) and enumeration are not permitted in class expressions.

For example, some of the axioms defined in Table 3.3 cannot be expressed in OWLPrime, e.g., *HomeOfficeTrajectory* and *BlockedStreetT*. However these limitations can be overcome by developers thanks to the possibility to define application-specific logical rules to complement the OWLPrime axioms. Section 3.4.2 presents a detailed example to design a rule for the *HomeOfficeTrajectory* concept.

3.4.1 Building the Semantic Trajectory Ontologies

Building the *Semantic Trajectory Ontology* (STO) for a given application is a multi-step process. First, the three module ontologies have to be set up. The *Geographic Ontology* (GO) and the *Traffic Domain Ontology* as the *Application Domain Ontology* (ADO) in our case study may be sub-ontologies extracted from more general existing ontologies. Indeed, several techniques are available to extract a module from an ontology such that (1) the module describes a set of concepts relevant for a given sub-domain, and (2) the module is an ontology itself. These techniques range from logical approaches that automatically extract a module based on given criteria, to languages that module designers use to manually select all the concepts and roles of the ontology that they want to have in the module [PSS08]. If no suitable ontology exists to start with, we have to build the STO from scratch.

The second step is to integrate the GO and ADO component ontologies with the Geometric Trajectory Ontology (GTO), whose TBox is application independent. One problem is how to deal with concepts and roles that are common to several components. For instance, the street concept is described in both GO and ADO as shown in Figure 3.3. One solution would be to keep the street concept in only one component, say GO. But that means suppressing in the other component all the roles that link this concept (e.g., isLocatedIn of ADO). This would be an important loss of semantics. Therefore, we prefer keeping the common concept into the two components, rename them streetT and streetG, and put a role in between, *sameAs*, that links the corresponding instances. The instances of sameAs are computed by a procedure that relies on the fact that the two street concepts share a common identifier. Other inter-component roles

have to be set up during the integration step, especially those defining the spatial and temporal properties (described in GTO) of the concepts of ADO or GO (e.g., hasGeometry).

Then the instances of GTO have to be computed as follows: (1) raw data (i.e., series of $\langle carId, x, y, t\rangle$) are preprocessed in order to correct acquisition errors and interpolate missing positions; (2) automatic segmentation algorithms divide the raw data series into trajectories; (3) we compute the stops and moves for each trajectory; and (4) we "upgrade" the trajectory semantics with more meaningful geographic and domain knowledge, by computing the role instances that link stops and moves to geographic and application objects. For the raw data containing 8'027'330 records in our traffic application, we have computed 364 semantic trajectories whose stops and moves have been linked to the corresponding objects of interest in ADO and GO, e.g., street segments, gas stations, or events.

It is worth noting that the computation of spatial relationships between the trajectory components (stops and moves) and the geographic and application objects can be performed by a GIS or a spatial DBMS such as Oracle. In addition, for these computations, we also use the semantic features of Oracle, like in the following query which computes the couples $\langle stop, gas\ station\rangle$ such that the stop is geographically located inside the gas station. These couples will be used to populate the role *isLocatedIn* that links B.E.S to *PointOfInterest*.

Table 3.4: Selecting trajectory stops that are inside gas stations

```
SELECT  Stop.id, poi.id
FROM Stop, (SELECT *
              FROM PointOfInterest p
              WHERE SEM_RELATED (p.type,
                  '<http://www.w3.org/2000/01/rdf-schema#type>',
                  ':GasStation>',
                 SEM_MODELS('trajectory_ontology'),
                 SEM_RULEBASES('OWLPRIME')) = 1
           ) poi
WHERE (SDO_RELATE(Stop.geometry, poi.geometry, 'MASK=INSIDE')='TRUE')
```

This query uses Oracle semantic reasoning (SEM_RELATED) to check if the point of interest is a gas station, and the Oracle spatial cartridge (SDO_RELATE) to check if the location (geometry) of the stop is inside the location (geometry) of the point of interest.

3.4.2 Querying the Semantic Trajectory Ontologies

To further validate the case study, we review the example queries discussed in the previous section of trajectory modeling requirements (see Section 3.2) and explain how these queries can be answered based on the STO ontology shown in Figure 3.3. We use SQL augmented with Oracle semantic features. The main advantage of this approach is to allow querying conjointly the ontology and relational data. An ontology query is specified by a conjunctive query using variables prefixed with a question mark (?), and concepts and roles prefixed with a colon (:). For

example, the query shown in Table 3.5 retrieves all stops that are located in a point of interest of type *Home*.

Table 3.5: Oracle semantic query using conjunctive query

```
SELECT s
FROM TABLE (SEM_MATCH(
        '(?s  rdf:type :Stop) (?s :isLocatedIn ?p)
         (?p rdf:type :Home)',
     SEM_MODELS('geopkdd_owl_ontology'),
     SEM_RULEBASES('OWLPRIME'),
     SEM_ALIASES(SEM_ALIAS('','http://lbd.epfl.ch/TrajectoryOntology#')),
    NULL));
```

Note that the conjunctive query (in Table 3.5) uses two variables, named s and p, respectively representing stops and points of interest. The SEM_MATCH constructor is used to define the conjunctive query. The SEM_MODEL and SEM_RULEBASES constructors respectively specify the name of the ontology and the set of of rules to be used for answering the query. The SEM_ALIASES constructor defines the namespace for the ontology.

The first query "*Return cars which stopped at point* $\langle x, y \rangle$ *at time* t", only involves basic trajectory knowledge from GTO and the role that links trajectory to the moving object (car) in ADO. The following expresses the English formulation as a conjunctive query on STO:

Table 3.6: Q1: Return cars which stopped at point (x,y) at t

```
        (?car     :hasTrajectory     ?traj)
        (?traj    :hasStop           ?stop)
        (?stop    :hasGeometry       <x,y>)
        (?stop    :hasTime           t)
```

Notice that this query does not call for reasoning tasks and could be expressed in SQL suitable for spatial data.

Like the first query, the second one "*Return cars which stopped today at a gas station*" uses GTO and ADO. It also uses the already computed semantic links that relate the stops to the points of interest in which they take place, i.e., the role isLocatedIn from B.E.S to PointOfInterest (see Figure 3.3). The corresponding conjunctive query is shown in Table 3.7.

The semantic description of the trajectory holds the knowledge that stops are located in points of interest, and gas stations are a special type of points of interest. This reliefs users from the tedious task of finding all locations of gas stations and testing if any one of the stops of today's trajectories is located inside a gas station.

The third query, "*Return persons who traveled yesterday afternoon from home to office*" requires more semantics than the previous ones. The concept of *HomeOfficeTrajectory* can be partially

Table 3.7: Q2: Return cars which stopped today at a gas station

```
(?car     :hasTrajectory   ?traj)
(?traj    :hasStop         ?stop)
(?stop    :isLocatedIn     ?poi)
(?poi      rdf:type        :GasStation)
(?stop    :hasTime         ?t)
(?t       :in              today)
```

defined in DL as was shown in Table 3.3. But this DL axiom suffers two problems: (1) It cannot be expressed in OWLPrime because of OWLPrime limitations; (2) The DL axiom defines HomeOfficeTrajectory as a trajectory that goes from a home to an office. It does not assert that the home and the office should correspond to the same person. These two problems can be overcome by using the logical rules facility of Oracle Semantic Technologies. Table 3.8 shows how to create a rule that defines another version of the concept HomeOfficeTrajectory. This new version defines a *HomeOfficeTrajectoryV2* as a trajectory that goes from the home of a person to the office of this person. Note that a rule is composed by two parts: The antecedent is first defined (lines 5-8) and then the consequent (line 10).

Table 3.8: Creating a rule that defines the HomeOfficeTrajectory concept

```
1. EXECUTE SEM_APIS.CREATE_RULEBASE('trajectory_rb');
2.
3. INSERT INTO mdsys.semr_trajectory_rb VALUES (
4.    'HomeOfficeRule',
5.    ' (?x rdf:type :Trajectory) (?x :hasBegin ?b)
6.      (?b :isLocatedIn ?bpoi)  (?x :hasEnd ?e)
7.      (?e :isLocatedIn ?epoi) (?p :hasHome ?bpoi)
8.      (?p :hasWork ?epoi) ',
9.    NULL,
10.   '(?x rdf:type :HomeOfficeTrajectoryV2)',
11.   SEM_RULEBASES('OWLPRIME', 'trajectory_rb'),
12.   SEM_ALIASES(SEM_ALIAS('','http://lbd.epfl.ch/TrajectoryOntology#'))
13. );
```

Then users' queries can directly refer to this concept to denote trajectories going from home to office, and query 3 can be rewritten in Table 3.9.

3.5 A Hybrid Spatio-Semantic Trajectory Model

In addition to the previous ontological framework for trajectory modeling, this thesis provides an alternative trajectory model, named "*Hybrid Spatio-Semantic Trajectory Model*". Different

Table 3.9: Q3: Return persons who traveled yesterday afternoon from home to office

```
(?person  :hasCar          ?car)
(?car     :hasTrajectory   ?traj)
(?traj     rdf:type        :HomeOfficeTrajectoryV2)
(?traj    :hasBegin        ?begin)
(?begin   :hasTime         ?t)
(?t       :in              yesterday_afternoon)
```

from the *trajectory ontological framework* in Section 3.3 that mainly focus on semantic analysis (including semantic trajectory modeling, querying and reasoning) and the moving object database models in literatures (e.g., [GS05]) that purely focus on data management (including data types, storage, indexing, querying), this hybrid *Spatio-Semantic* trajectory model can: (1) encapsulate raw GPS spatio-temporal trajectory data; (2) allow for progressive abstraction of the raw data to higher-level semantic representation of such data; (3) encapsulate well-known concepts used in literature for trajectories, e.g., stop-move in [SPD+08]. The model is usable by several applications requiring varying degrees of such comprehensive trajectory representations from data as well as semantic perspectives (and hence hybrid). It offers mobility data representation for trajectory computing, providing different levels of trajectory data abstraction. The key design considerations of such hybrid model are:

- *Raw Data characteristics:* Model should consider characteristics of raw mobility tracking data (e.g., spatial and temporal gaps, uncertainties) to create simple *low-level* representations (e.g., hourly, daily, monthly and geo-fenced trajectories).
- *Progressive computation possible:* Model should be designed so that a layered computing platform can systematically generate higher-level semantic abstractions from underlying lower-level trajectory representations.

As shown in Figure 3.4, this hybrid trajectory model consists of three sub models. In the low-level, it is the *data model*, which builds the *spatio-temporal trajectory* as a sequence of GPS points $\langle x, y, t \rangle$, corresponding to the modeling requirement of the spatio-temporal knowledge in Section 3.2; in the top-level, it is the *semantic model*, which builds the *semantic trajectory* as a sequence of episodes with semantic annotations, corresponding to the modeling requirement of the semantic knowledge in Section 3.2. In the middle-level, we build an intermediate *conceptual model* to better bridge the gap between low-level mobility data and high-level trajectory semantics. The conceptual model is an extension on the *trajectory conceptual view* in [SPD+08], and builds *structured trajectory* as a sequence of trajectory episodes.

3.5.1 Data Model

Building the spatio-temporal trajectory in the data model is the first level in trying to understand real-life trajectory data of moving objects. In this level, the focus is on the raw movement data,

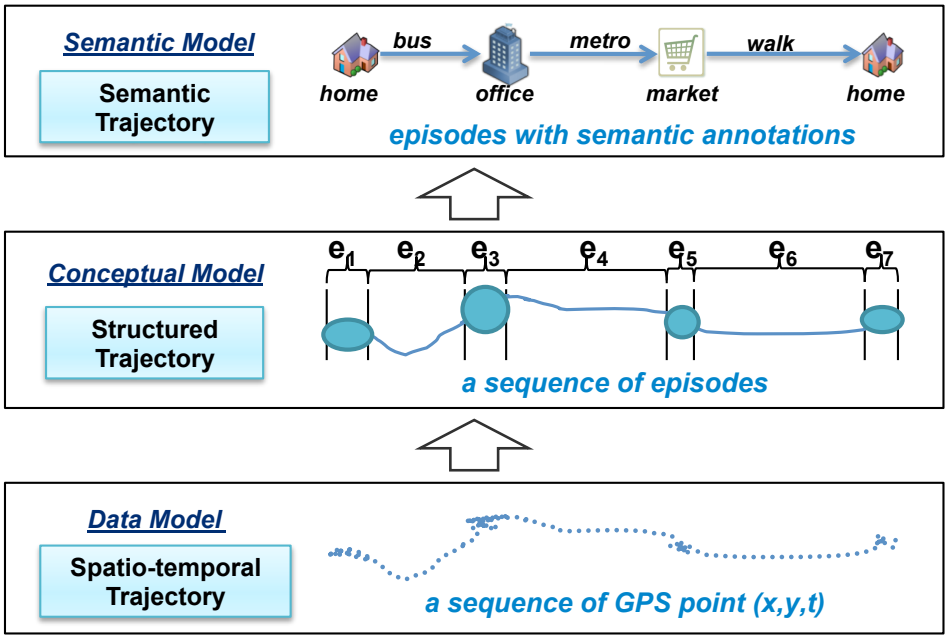


Figure 3.4: A hybrid semantic trajectory model

which is usually captured by emerging ad-hoc mobile tracking devices that record the evolving position where a moving object temporally resides. Such raw movement data is usually defined as "a sequence of spatio-temporal tuples $(position, time)$". For simplicity, but without loss of generality, $(position, time)$ can be redefined as (x_i, y_i, t_i) in a 2D space (similar for the high-dimensional moving space), where (x, y) is the 2D spatial values and t is the distinct and increasing time instant. Most real world tracks are GPS alike records $(longitude, latitude, timestamp)$, which is also the main data format in our experimental case study. From now on, we simply use (x, y, t) to represent the movement records; and use the term *GPS feed* to represent the *raw* sequence of (x, y, t) spatio-temporal mobility data.

It is worth clarifying that the raw GPS feeds often need to undergo a *data preprocessing* process, e.g., *data cleaning* to correct GPS errors that commonly exist in many trajectory data acquisition devices, *data compression or interpolation* when the data sampling frequency is too high or too low. This trajectory data preprocessing step will be concretely discussed in Section 4.3 in the trajectory computing chapter.

In addition, the sequence of real-life GPS recording tuples (x_i, y_i, t_i) is usually very long (e.g., the 3M and 7M GPS records respectively mentioned in vehicle & people trajectories). A key concept in computing the spatio-temporal trajectory ($\mathcal{T}_{spa}$) is to identify the meaningful *dividing points* in a raw GPS feed that separates two temporally ordered $\mathcal{T}_{spa}$. A dividing point identifies the *end* of one trajectory $\mathcal{T}_{spa}$ and the *begin* of the next one. Consequently, a trajectory has a starting point and an ending point that delimit the sequence, and a trajectory span over the time interval $[t_{begin}, t_{end}]$ occupied by the sequence (same as the time domain in the time-space function that previously discussed in Section 3.2.2). Note that the exact begin and end coordinates may not be co-located (e.g., due to data collection gaps). Typical examples of such dividing points could be temporal (daily, hourly trajectories) or spatial (trajectory of a car in a city) or places in the raw feed where there are large spatial or temporal disconnections. The

detailed methods of identifying such dividing points and computing $\mathcal{T}_{spa}$ from the long sequence of GPS feed will be presented in Section 4.4 in the trajectory computing chapter.

In terms of the two steps of *data preprocessing* and *trajectory identification* in trajectory computing, we can build the spatio-temporal trajectories $\mathcal{T}_{spa}$ (see Definition 3.1) from the initial raw movement data. Figure 3.5 sketches a spatio-temporal trajectory, which is a sequence of spatio-temporal points.

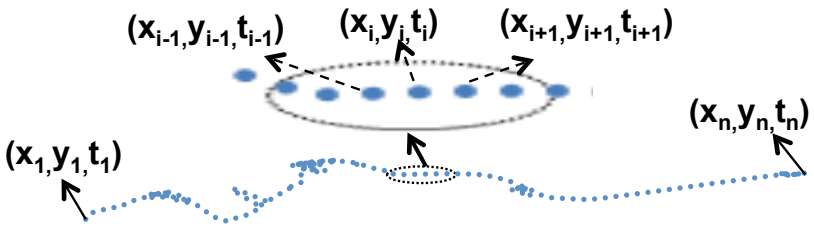


Figure 3.5: Spatio-temporal trajectory (a sequence of spatio-temporal points)

Definition 3.1 (**Spatio-temporal Trajectory** $\mathcal{T}_{spa}$)**.** A spatio-temporal trajectory $\mathcal{T}_{spa}$ is a cleaned subsequence of raw movement track (e.g., GPS feed) for a given moving object in a given time interval [t_begin, t_end]. It is a list of triples (x, y, t) such that all the t are different and the list is ordered by increasing t (see Figure 3.5), i.e., $\mathcal{T}_{raw} = \{p_1, p_2, \ldots, p_n\}$ where $p_i = (x_i, y_i, t_i)$ represents a spatio-temporal point.

3.5.2 Conceptual Model

Intuitively, conceptual model is the logical partitioning of a *single* spatio-temporal trajectory $\mathcal{T}_{spa}$ into a series of non-overlapping temporally separated *episodes*. A $\mathcal{T}_{spa}$ having such aggregated episodes is called a *Structured Trajectory* ($\mathcal{T}_{str}$).

Definition 3.2 (Structured Trajectory $\mathcal{T}_{str}$)**.** A structured trajectory $\mathcal{T}_{str}$ consists of a sequence of trajectory units, called "episode", i.e., $\mathcal{T}_{str} = \{e_1, e_2, \ldots, e_m\}$

- A episode *(e)* groups a subsequence of $\mathcal{T}_{spa}$ with k consecutive GPS points having some similar characteristics $\{p_1^{(e_i)}, \ldots, p_k^{(e_i)}\}$ derived from $\mathcal{T}_{spa}$.
- For data compression, episode is a tuple that stores only the subsequence's temporal duration and spatial extent. $e_i = (time_{from}, time_{to}, bounding_{rectangle}, center)$.

A trajectory *episode* abstracts those subsequences of spatio-temporal tuples that show a high degree of correlation w.r.t. some identifiable feature (e.g., velocity, angle of movement, density, time interval etc). This generic *structural* representation enables us to compute such sequences using *trajectory segmentation* techniques (described in Section 4.5). By trajectory segmentation to get a more informative view of trajectories, a $\mathcal{T}_{spa}$ can be structured into segments corresponding to meaningful steps in the travel. The underlying idea is that the moving object may, during a trajectory, iteratively stop for some relevant purpose and resume traveling later until arriving at the end of the trajectory. The result of such segmentation is a structured trajectory $\mathcal{T}_{str}$ in terms of a sequence of trajectory episodes. An *episode* has the following salient features:

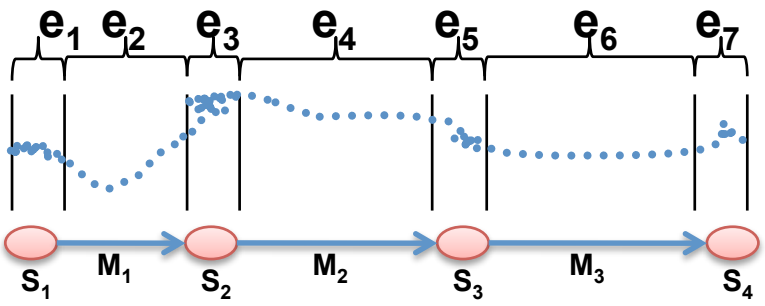


Figure 3.6: Structured trajectory (a sequence of episodes)

- *Encapsulates semantic trajectory concepts:* High-level trajectory concepts such as *begin*, *end*, *stops*, *moves* [SPD+08] become *sub-classes* of *episode*. Moreover, it also encapsulates additional meaningful trajectory concepts such as *jumps*, *pattern-driven movement* sequences (not necessarily *stop* or *move*), which is defined in literature for domains such as trajectory of wild life [SYRS05].

- *Computed automatically:* Episodes can be computed with relevant *Trajectory Structure* algorithms. This is because the correlations are essentially geometric characteristics of GPS feed like *velocity*, *acceleration*, *orientation*, *density*, or other *spatio-temporal* correlations.

- *Enables Data Compression:* Instead of semantic annotation of each GPS records directly (which is possible), episodes essentially enable single semantic tagging of correlated GPS tuples having similar features. This reduces the data size to represent trajectories at the conceptual level. For example, in Figure 3.6, a spatio-temporal trajectory is segmented into seven episodes, from e_1 to e_7, among which there are four stops and three moves identified. Obviously, semantic annotation of seven episodes in the conceptual model which is more efficient than annotation of each GPS tuple in the data model.

For a specific trajectory application, not all kinds of episodes are interesting or useful. For example, some applications like tourist trajectories only put interests in *stops* (named stay points [ZZXM09a] or point of interests [PBKA08][GKV08]), where *moves* can be simply determined as the linkage between two consecutive *stops*.

3.5.3 Semantic Model

In semantic model, we can build *semantic trajectory* $\mathcal{T}_{sem}$ which is an annotated enhancement of a structured trajectory $\mathcal{T}_{str}$ enriched with semantic knowledge. Such annotations can be made on episodes in the $\mathcal{T}_{str}$ as well as on the whole $\mathcal{T}_{str}$. A typical example of a semantic trajectory is shown in Figure 3.7, which is annotated based on the previous structured trajectory in Figure 3.6. In this semantic trajectory, the GPS feeds are enriched with knowledge of an employee's spatio-semantic movement pattern: she/he goes to work from *home* (morning); after *work* (later afternoon), he leaves for shopping in *market*, and finally reaches *home* (evening).

Definition 3.3 (Semantic Trajectory $\mathcal{T}_{sem}$)**.** A semantic trajectory $\mathcal{T}_{sem}$ is a structured trajectory with added semantic annotation: episodes are enriched in terms of semantic episodes

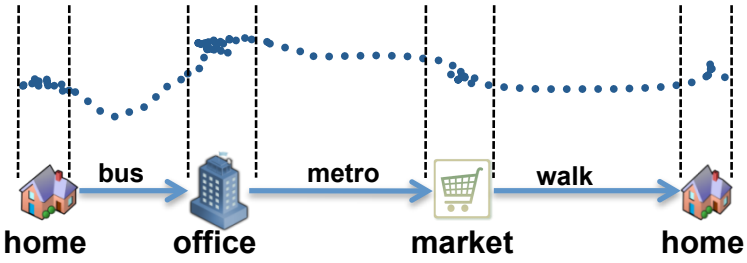


Figure 3.7: Semantic trajectory (a sequence of semantic episodes)

(se) with geographic or application knowledge. i.e., $\mathcal{T}_{sem} = \{se_1, se_2, \ldots, se_m\}$, where semantic episode $se_i = (sp_i, t_{in}^{(sp_i)}, t_{out}^{(sp_i)}, tags)$

- sp_i *(semantic position)* is a meaningful location object, which can be real-world objects from geographic knowledge (e.g., building, roadSegment, administrativeRegion, landuse), or more application domain knowledge (e.g., home, office that belong to specific people in a given application). From the underlying spatial extents, sp_i can be categorized into three types, i.e., ROI (region of interest), LOI (line of interest), POI (point of interest) according to the shape of region, line, and point respectively.

- $t_{in}^{(sp_i)}$ is the incoming timestamp for trajectory entering this semantic position (sp_i), and $t_{out}^{(sp_i)}$ is the outcoming timestamp for trajectory leaving sp_i. They can be approximated by $time_{from}$ and $time_{to}$ in episode.

- $tags$ is a set of other annotations associated the whole episodes, e.g., walk (in the road), cycling (in the road), work (at home), break (at office).

- From data compression point of view, many episodes in one or more structured trajectories $\mathcal{T}_{str}$ can be located in the same sp_i in $\mathcal{T}_{sem}$.

Our model is designed so that it can integrate data from third party sources, including geographic knowledge (e.g., landuse data, road network, points of interest) and application domain knowledge (e.g., application databases, social network data related to locations). The model instances can even be inferred through learning patterns from underlying GPS feeds. Chapter 5 will present the semantic enrichment methodologies that can be applied on such third party data towards computing instances of the semantic model, i.e., the semantic trajectories.

3.6 Summary

This chapter presented the first major contribution of this thesis, i.e., the semantic trajectory modeling. As mentioned, this thesis has intensively built two important trajectory modeling methodologies: one is the trajectory ontological infrastructure, and the other is the hybrid spatio-semantic trajectory model. The initial result of trajectory ontologies is published in [YMPS08a], with its extended version for supporting high-level trajectory queries and reasonings in [YMPS08b]. The hybrid spatio-semantic model is initially published in [YPSC10].

The two trajectory models share the same emphasis on semantically trajectory modeling, covering the modeling requirements of semantic knowledge, including both geographic knowledge and application domain knowledge for better understanding mobility data. The trajectory ontological infrastructure includes three module ontologies, i.e., geometric ontology, geographic ontology, and application domain ontology; whilst the hybrid spatio-semantic model consists of data model with spatio-temporal trajectory, conceptual model with structured trajectory, and semantic model with semantic trajectory. Therefore, both models discuss the data view and the semantic view.

The major difference between the two models is: the ontological infrastructure is focusing on the high-level semantic querying and reasoning on trajectories, which is significantly limited to a small dataset because of the time complexity in semantic reasoning; the hybrid spatio-semantic model is focusing on the different levels of data representation, from the low-level raw data generated by positioning devices to the high-level semantic abstraction, which works for a large scale of real-life GPS alike tracking data.

Therefore, the choice between such two models depends on the application requirements: if the trajectory application is data intensive for better data abstraction to understand basic trajectory semantics, the hybrid model is a good choice; otherwise, if the trajectory application is semantics intensive and needs complex conjunctive trajectory querying and reasoning, the ontological framework can work better.

Chapter 4

Offline Trajectory Computing

Computing is the study of information processes, natural and artificial.

Peter J. Denning, 2008

4.1 Introduction

In this chapter, we present *offline trajectory computing* as the second major contribution of this thesis. The objective of trajectory computing is to exploit the potential of the *hybrid spatio-semantic trajectory model* in Chapter 3, for reconstructing meaningful trajectories at different levels of data abstraction from the real-life raw mobility tracking data. In this chapter, we focus on computing *spatio-temporal trajectories* and *structured trajectories* from large-scale GPS alike movement datasets. Regarding the techniques for computing *semantic trajectories*, the dedicated semantic enrichment functionality will be specifically addressed in Chapter 5. All of the detailed computing algorithms (e.g., mobility data cleaning, compression, and segmentation) in this chapter are designed based on an offline procedure, which means the raw trajectory data is collected in advance and then processed together in batch mode. On the contrary, Chapter 6 will address the complementary strategy for online trajectory computing in the real-time trajectory application context.

This chapter is organized as follows: Section 4.2 presents the offline trajectory computing framework; Section 4.3 addresses the *data preprocessing* algorithms such as trajectory data cleaning and data compression; Section 4.4 discusses the *trajectory identification* polices that are used for dividing a long sequence of tracking data into individual spatio-temporal trajectories; Section 4.5 is the fundamental part of trajectory computing, i.e., the *trajectory structure* layer that builds relevant trajectory segmentation algorithms; Section 4.6 shows some experimental studies; finally Section 4.7 summarizes this chapter.

4.2 Trajectory Computing Framework

During the investigation of modeling requirements on trajectory data in Section 3.2, there is one important aspect of "*spatio-temporal knowledge*", which only focuses on the geometric view to interpret and understand mobility data. For such geometric point of view, our hybrid spatio-semantic model defines two levels of trajectory data abstraction, i.e., the *spatio-temporal trajectory* ($\mathcal{T}_{spa}$) and the *structured trajectory* ($\mathcal{T}_{str}$). To apply this hybrid model in real-life trajectory applications, this thesis designs a trajectory computing framework, for subsequently constructing $\mathcal{T}_{spa}$ and $\mathcal{T}_{str}$ from the raw GPS feeds that are collected by positioning sensors (see Figure 4.1). This is an offline computing procedure, which assumes the raw data is collected in advance. We additionally design an online computing framework in Chapter 6, which can work directly on the streaming movement data in real-time applications.

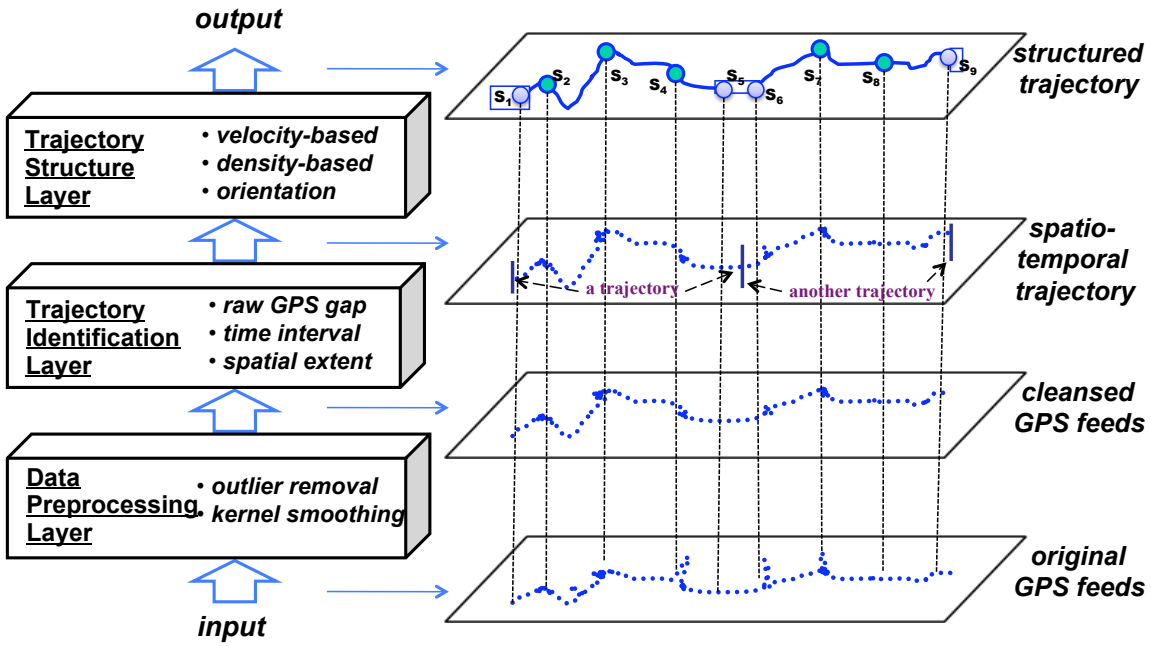


Figure 4.1: Offline trajectory computing framework

As shown in Figure 4.1, the offline trajectory computing framework consists of three main computing layers, i.e., *data preprocessing*, *trajectory identification*, and *trajectory structure*.

1) **Data Preprocessing Layer:** As mentioned in Section 2.3.1, GPS alike raw movement data is usually noisy with errors. Therefore, raw data needs to be cleaned essentially to provide accurate mobility data computing and understanding. Therefore, the objective of this data preprocessing layer is to clean the raw GPS feeds, in terms of designing a couple of preliminary tasks such as *outliers removal* and regression-based *kernel smoothing*, *map matching* for cleaning network constrained trajectory data etc. The outcome of this step is a cleaned sequence of new spatio-temporal points (x', y', t'). In addition, some extra data preprocessing techniques like *data format transformation* and *data compression* are required before the main steps of trajectory computing.

2) **Trajectory Identification Layer:** This layer is designed for dividing the long sequence of cleaned (x', y', t') GPS spatio-temporal points into several meaningful subsequences, where each subsequence is corresponding to a separate spatio-temporal trajectory $\mathcal{T}_{spa}$. Such division exploits gaps present in the underlying data and in addition, applies well-defined policies of spatial (e.g., trajectories in the EPFL campus, Lausanne city area etc.) and temporal demarcations (e.g., time interval for a day, week). The policy of automatic division without prior spatial/temporal knowledge is also investigated in this layer.

3) **Trajectory Structure Layer:** This is the core layer in the whole trajectory computing procedure, which is responsible for identifying the *episodes* present in each spatio-temporal trajectory and generates the structured trajectories $\mathcal{T}_{str}$. Trajectory structure layer in essential is to design and apply relevant *trajectory segmentation* algorithms to segment each single $\mathcal{T}_{spa}$ into a set of episodes, in terms of using the correlations between temporally occurring sequence of GPS points (e.g., velocity, density, orientation etc.).

The detailed computing techniques with dedicated algorithms in these three layers will be presented in the following three sections.

4.3 Data Preprocessing Layer

The objective of data preprocessing is to design relevant preliminary tasks before the main steps of trajectory data analysis. Typical data preprocessing tasks focus on cleaning the raw mobility data originally coming from the mobile positioning devices. In this thesis, we define the following four main aspects of GPS-alike mobility data preprocessing.

(a) *Data Transformation* – This is the task of converting the raw GPS data in the geographic coordinate system (in the form of $\langle longitude, latitude, altitude \rangle$ positioning data point) into the cartesian coordinate system (in the form of the $\langle x, y, z \rangle$ data point). The cartesian data format can be easily used in data analysis compared to the geographic format, e.g., calculating the distances, velocity and other GPS features.

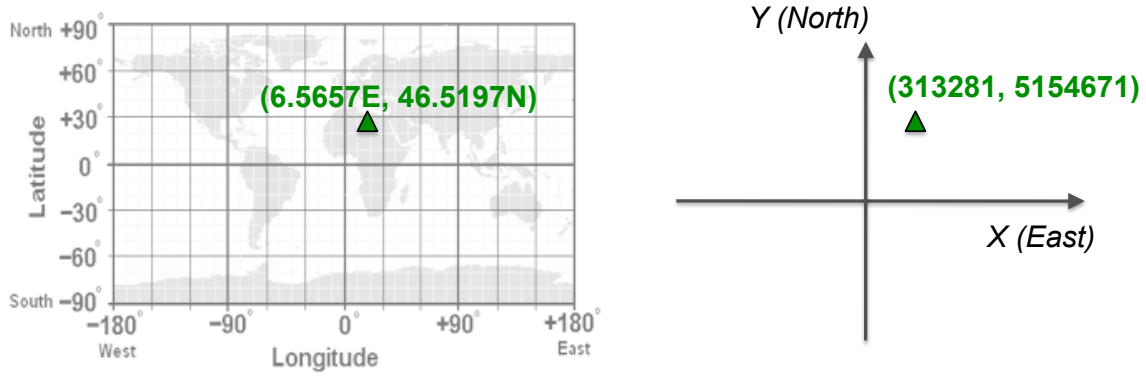
(b) *Cleaning via Smoothing and Filtering* – As real-life movement data, GPS has two types of errors: *systematic errors* (i.e., the large errors that are totally invalid positioning from the actual location) and *random errors* (i.e., the small errors up to ± 15 meters that are caused by measurements) [JGO06]. We design both *filtering* methods to remove GPS points caused by systematic errors and *smoothing* methods to reduce the effects of random errors.

(c) *Cleaning via Map Matching* – The previous data cleaning via smoothing and filtering is applied for trajectories of freely-moving objects. However, for mobility data that are generated by network constrained moving objects [GdAD06], we need to apply relevant *map matching* techniques to clean the raw GPS tracks and derive the new ones that need to be consistent with the underlying road networks.

(d) *Data Compression* – The size of GPS alike tracking data can go to infinity as tracking time goes by, therefore data compression is required for computing trajectories from the increasing amount of movement data. The trajectory data compression algorithm in essence is to remove data points that are "repetition" in some sense, and keep only informative ones for achieving better performance of computation and storage.

4.3.1 Data Transformation

In most of the existing trajectory application scenarios, the mobility datasets are collected by GPS-alike positioning sensors and use the *Geodetic Coordinate* system as the data format, namely the *longitude* and the *latitude* that apply the Earth angular units to represent the location. For example, Figure 4.2-(a) shows a GPS point located in EPFL campus with a coordinate $\langle longitude = 6.5657East,\ latitude = 46.5197North \rangle$. For achieving convenient data analysis and computation, such $\langle longitude, latitude \rangle$ data points need to be converted into the *Cartesian Coordinate* system. Different countries typically use their own cartesian coordination systems according to the specific time zone, e.g., Swiss coordinate system CH1903[1] in Switzerland. In our experiment, we apply the UTM [2] (*Universal Transverse Mercator*) cartesian system that work on various datasets. With such transformation, the EPFL $\langle longitude, latitude \rangle$ is transformed to the new point $\langle x, y \rangle = \langle 313281m, 5154671m \rangle$ (see Figure Figure 4.2-(b)), where the unit is meter; x is the projected distance of the position eastward from the central meridian, while y is the projected distance of the point north from the equator. Compared with the raw geodetic coordinates, the converted cartesian coordinates are much easier for further trajectory computing, such as the tasks of calculating GPS features such as velocity, acceleration, distance, direction, and heading change. There are a couple of open-source solutions that support such kind of coordinate transformation, for example the JCoord API[3] that we have applied.



(a) Geodetic coordinate system (b) Cartesian coordinate system

Figure 4.2: Data transformation of coordinates – from $\langle longitude, latitude \rangle$ to $\langle x, y \rangle$

[1] http://en.wikipedia.org/wiki/Swiss_coordinate_system
[2] http://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system
[3] http://www.jstott.me.uk/jcoord/

4.3.2 Cleaning via Filtering and Smoothing

Due to GPS measurements and sampling errors from mobile devices, the recorded positions of a moving object are not always precise and usually contain errors [ZG02]. Applications need to apply techniques to identify possible causes for such errors and further remove them or reduce their influence. Therefore, one of the main component in the *Data Preprocessing Layer* is to deal with such real-life noisy data and clean them before fitting the *hybrid spatio-semantic* model on computing trajectories at different levels.

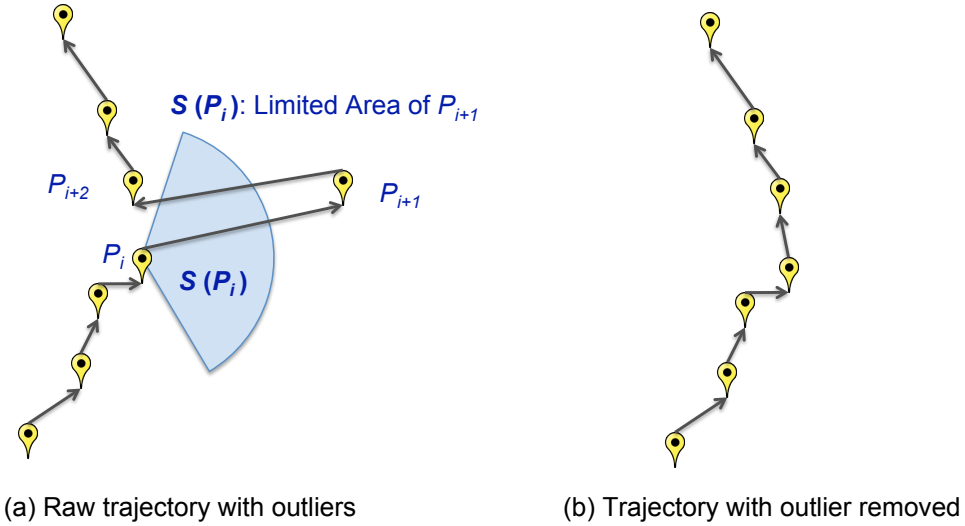
By adopting the summary of GPS errors in [JGO06, SA09b], we focus on dealing with both types of GPS errors, i.e., the *systematic errors* and the *random errors*, with the methods of *filtering outliers* and *smoothing* respectively.

(1) ***Filtering outliers*** – Figure 4.3 sketches a real trajectory example visualized in Google Earth by using KML (*Keyhole Markup Language*) – an XML notation and file format for modeling and storing geographic features such as points, lines, images etc. In this figure, there are tens and hundreds of GPS records with two outliers (data points in the triangle) that are far away from the real trajectory data (data points in the rectangle).

Figure 4.3: A trajectory example with two outliers

To remove such outliers as systematic errors, we design *threshold-driven* data filtering strategies. The idea is to build a "limited area" $S(P)$ for a GPS point (P) that needs to be checked, and according to the topological *inside* relationship between the point P and the area $S(P)$ to determine whether P is outlier or not. For example, to check the point P_{i+1} in Figure 4.4-(a), we compute the "limited area" $S(P_i)$ based on the previous point P_i by using the maximum speed, time duration $(t_{i+1} - t_i)$, and the previous moving direction. We get an arc area as the "limited area" of P_{i+1}; as P_{i+1} is out of such area (i.e., $P_{i+1} \notin S(P_i)$), the data point P_{i+1} needs to be removed as outlier. After removing such outlier point, we can get a new filtered trajectory data sequence as shown in Figure 4.4-(b).

(2) ***Smoothing random errors*** – Regarding the *random noises*, we design a Gaussian kernel based local regression model to smooth the GPS feed. The smoothed position $(\widehat{x}, \widehat{y})$ is the weighted local regression based on the past points and future points within a sliding time window,



(a) Raw trajectory with outliers (b) Trajectory with outlier removed

Figure 4.4: Filtering the outliers in trajectory data

where the weight is a Gaussian kernel function $k(t_i)$ and the window size is the kernel bandwidth σ (Formula 4.1). To control the smoothing related information loss, we adopt a reasonably small value for σ (e.g., 5 $\times$ GPS sampling frequency) so that only nearby points can affect the smoothed position. This is necessary as we want to calibrate the technique to not only handle the noise but also to avoid *over-fitting*, which means the smoothing technique is over-performing and the new data is too smoothed, i.e., loosing the initial data characteristics significantly.

$$\widehat{x} = \frac{\sum_i k(t_i) \times x_{t_i}}{\sum_i k(t_i)}$$
$$\widehat{y} = \frac{\sum_i k(t_i) \times y_{t_i}}{\sum_i k(t_i)}$$
$$where\ k(t_i) = e^{-\frac{(t_i - t)^2}{2\sigma^2}} \tag{4.1}$$

Figure 4.5, 4.6, 4.7 show an example of our smoothing algorithm applied on a real dataset taken from wild-life tracking data of a monkey movement on 15/03/2008. This trajectory contains 52 GPS (x,y,t) records. Figure 4.5 shows the original longitude (actually the transformed X value in cartesian coordinate) and the smoothed value, together with the Gaussian regression curve. Similarly, Figure 4.6 shows the original and the smoothed latitude (Y in cartesian coordinate). Figure 4.7 plots the trajectory (the two dimensions together), including both the original GPS feed before smoothing and the new one after smoothing. From Figure 4.7, we do observe that the new trajectory sequence is much smoother than the original one.

4.3.3 Cleaning via Map Matching

The previous trajectory data smoothing techniques are designed for freely moving objects. However, in many trajectory scenarios, objects (people or vehicles) have to move along certain network constrained paths (e.g., transportation network like bus routes or metro lines) [GdAD06]. To provide a concrete example, Figure 4.8 shows trajectory data points produced by moving

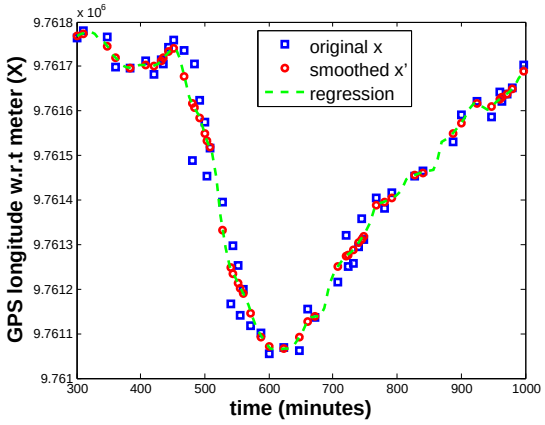


Figure 4.5: Smooth GPS (x)

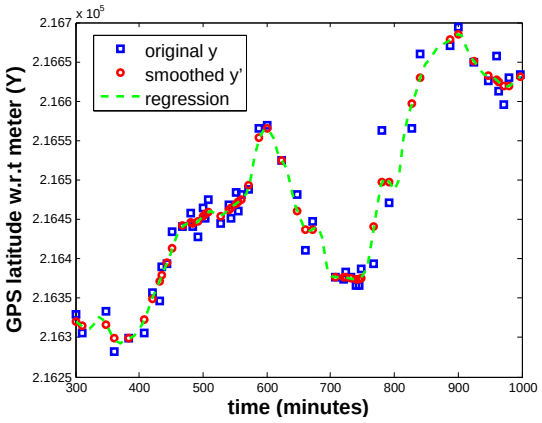


Figure 4.6: Smooth GPS (y)

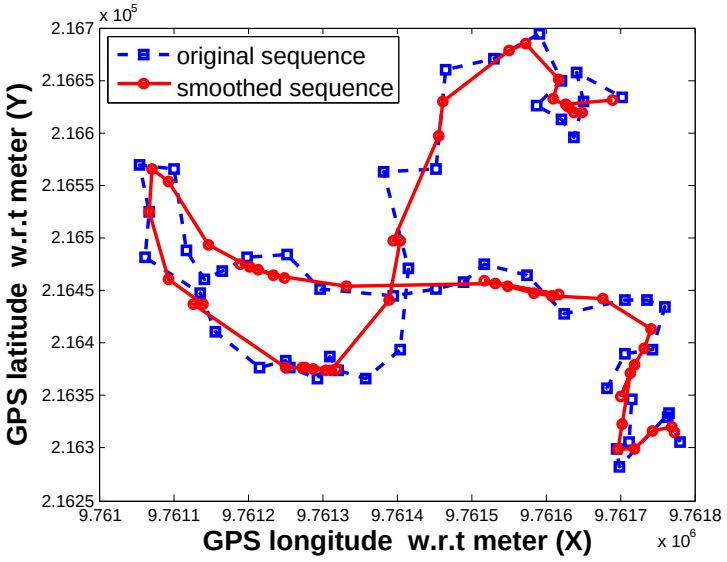


Figure 4.7: Original sequence and smoothed new sequence

objects with network constraints, which is generated by a moving object data simulator by Brinkhoff [Bri03]. We can observe such trajectory data needs to be consistent with the underlying road network. Therefore, regarding network-constrained trajectory data, map matching can be applied for data cleaning, by integrating positioning data with spatial road network to identify the correct link on which a vehicle is traveling and to estimate the correct location of a vehicle on that matched link [QON07, BPSW05].

Before presenting the detailed procedure of data cleaning via map matching, we need to provide the definition of network constraints, which determine the underlying network infrastructure that moving objects need to follow and keep inside. This network constraint can be represented as a graph, including a set of points (i.e., road crossings) and the edges (i.e., road segments) linking two connected points.

Definition 4.1 (Network Constraints)**.** A network for constraining trajectories is a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, in which $\mathcal{V}$ is the set of vertex $\{v_1, v_2, \ldots, v_n\}$ (i.e., road crossings), and $\mathcal{E}$ is a set of road segments connecting vertex $\{e_1, e_2, \ldots, e_m\}$. $\mathcal{E}$ can be represented as a $n \times n$

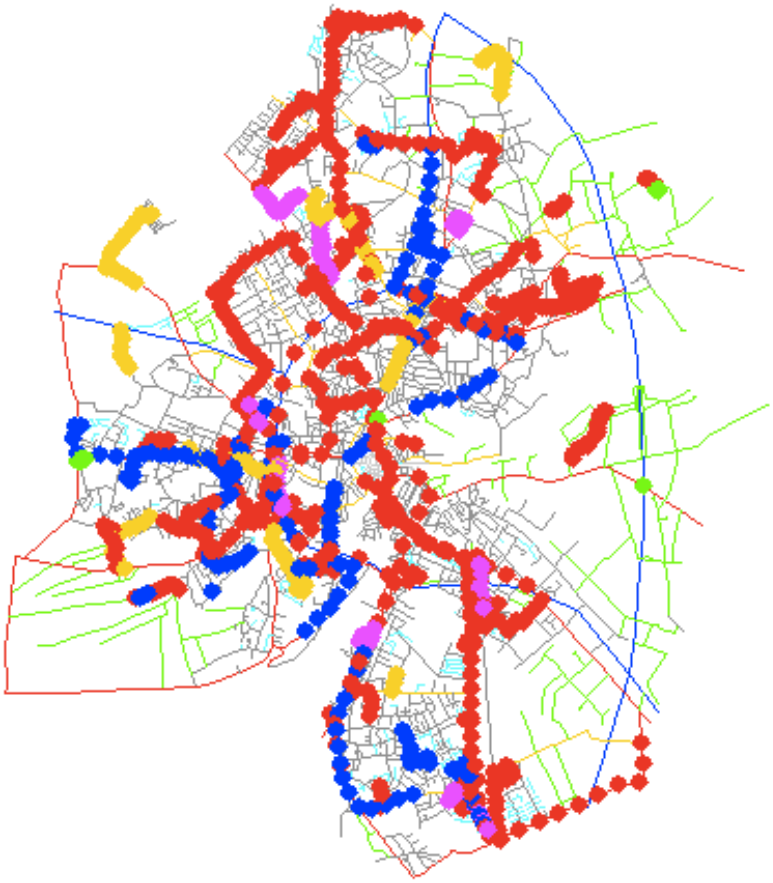


Figure 4.8: Network constrained trajectories

matrix of vertex, i.e., $[e_{ij}]_{n\times n}$, in which e_{ij} can be $0, 1, \infty$ for representing direct-connection, self-connection and no-connection, respectively.

In real-world network scenarios, the number of edges is usually significantly less than the total number of any possible combinations between vertex, i.e., $|\mathcal{E}| << |\mathcal{V}|^2$. Therefore, we may simply use the road segment list instead of a matrix to represent $\mathcal{E}$ as it is not efficient to store too many ∞ values in a sparse matrix.

Definition 4.2 (Network Constrained Trajectory)**.** The trajectories under network constraints can be defined as $\mathcal{TN} = \{\mathcal{TS}, \mathcal{G}\}$, where $\mathcal{TS}$ is a set of trajectories, which might belong to one moving object or many different moving objects $\mathcal{TS} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m\}$; $\mathcal{G}$ is the underlying constraining network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. For the correct (cleaned) network-constrained trajectory datasets, given any spatio-temporal data point $\langle x, y, t\rangle \in \mathcal{T}$, its location (x, y) should be in a road edge in $\mathcal{E}$ (with a special case in the crossing point in $\mathcal{V}$), but not outside the network $\mathcal{G}$.

According to such network-constrained trajectory definition, we can clean the original datasets via map matching: by using the underlying network constraints $\mathcal{G}$ (e.g., roadway centerlines), the original location (x_i, y_i) in a trajectory data point needs to be map-matched into road segments $e_j \in \mathcal{G}$ on which a vehicle is traveling, and to estimate a new location (x'_i, y'_i) on that road segment e_j. Therefore, a point (x_i, y_i, t_i) that does not obey the constraint is replaced by (x'_i, y'_i, t_i) where $(x'_i, y'_i) \in e_j$. This can be done by analyzing relevant spatial correlations

between the (x_i, y_i, t_i) point and the neighboring road segments. An intuitive idea is to find the closest edge with the shortest distance from the given point, and *perpendicular distance* is frequently applied. In geometry, the perpendicular distance from a point (x_1, y_1) to the line $ax + by + c = 0$ is given by $d = \frac{|ax_1+by_1+c|}{\sqrt{a^2+b^2}}$ (see the left side of Figure 4.9).

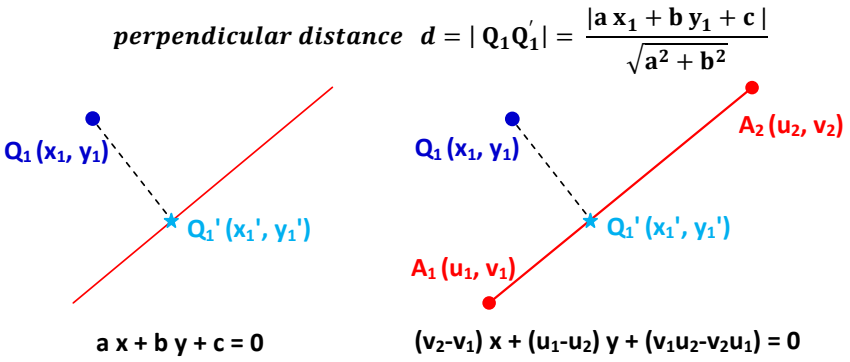


Figure 4.9: Perpendicular distance: between point and edge

For the road network, the line (road segment) is determined by two crossing points (u_1, v_1) and (u_2, v_2) – see the right side of Figure 4.9, in terms of the formula $(v_2 - v_1)x + (u_1 - u_2)y + (v_1u_2 - u_1v_2) = 0$. Therefore, the perpendicular distance between Q_1 and the line A_1A_2 is defined in Formula 4.2,

$$d = \frac{|(v_2 - v_1)x_1 + (u_1 - u_2)y_1 + (v_1u_2 - u_1v_2)|}{\sqrt{(v_2 - v_1)^2 + (u_2 - u_1)^2}} \tag{4.2}$$

The perpendicular distance can not be directly used for map matching, because in the network constraint the road segment is only a subpart of a line. In Figure 4.10, for the distance between point Q_1 and the five neighboring road segments (i.e., $A_1A_2, A_2A_3, A_3A_4, A_4A_5$), only A_2A_3 can accept the perpendicular distance, because the other projected points $Q'_{A_1A_2}$, $Q'_{A_3A_4}$, $Q'_{A_4A_5}$ do not belong to the corresponding road segments (i.e., $Q' \notin A_iA_j$).

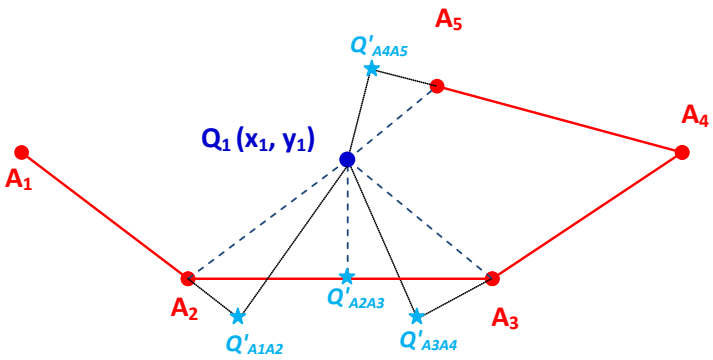


Figure 4.10: Distance between point and edge segment

Therefore, our minimum distance between a given GPS position $Q(x, y)$ and a road segment A_iA_j is calculated as follows,

$$d(Q, A_iA_j) = \begin{cases} d(QQ') & \text{if } Q' \in A_iA_j \\ min\{d(QA_i), d(QA_j)\} & \text{otherwise} \end{cases} \tag{4.3}$$

where Q' is the projection of Q on A_iA_j, $d(QQ')$ is the perpendicular distance between Q and the line segment determined by the two crossings A_i and A_j, and $d(QA)$ is the euclidean distance between the GPS point and the road crossing.

Instead of using only local point to infer the matched edge, many global map matching algorithms have been proposed [BPSW05] [QON07], to further improve the matching accuracy as the global methods consider the context of neighboring GPS points. For instance, in Figure 4.11, the two points Q_2 and Q_3 are matched to segments A_2A_5 and A_2A_4 respectively based on the local point-edge distance. However, from a global viewpoint (e.g., forward or backward or both), the correct matching of Q_2 should be in segment A_1A_2 because its previous points Q_0 and Q_1 match segment A_1A_2; similarly, the correct matching of Q_3 should be in segment A_2A_4 because its forthcoming points Q_4 and Q_5 match segment A_2A_4.

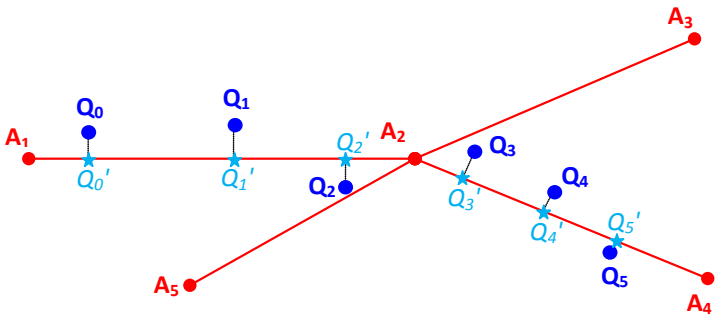


Figure 4.11: A global map matching approach

Therefore, for achieving better map matching results, we need to design a global matching algorithm, which determines the edge for each GPS point according to its $2N$ neighborhood points (N before and N after). Hereby, we normalize the distance between GPS position and road segment $d(Q, A_iA_j)$, and get the normalized local distance *localScore* between them. The *localScore* between a point Q and a segment A_iA_j is the basic measurement for mapping Q to A_iA_j without consideration of its previous and forthcoming points (as the initial but wrong map matching result in Figure 4.11 where Q_2 matches A_2A_5 and Q_3 matches A_2A_3).

$$localScore(Q, A_iA_j) = \begin{cases} \frac{d_{min}(Q)}{d(Q,A_iA_j)} & A_iA_j \in candidateSegs(Q) \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

where $d_{min}(Q)$ is the shortest distance from Q to all possible edges A_iA_j; for better performance, only the neighboring edges (*neighbor*, say in 20m or the top 6 closest edges) need to be considered for each GPS point, not the complete edge set.

Hereinafter, we can compute the *globalScore* between a point Q and the segment A_iA_j to be matched, in a global view as including the previous N points and the forthcoming N points. The distance between those points and Q should be less than a given radius R (say 40 meters), otherwise, it has no effects by setting zero weight.

$$globalScore(Q, A_iA_j) = \frac{\sum_{k=-N}^{N} w_k \cdot localScore(Q_k, A_iA_j)}{\sum_{k=-N}^{N} w_k} \tag{4.5}$$

where Q_k is the k^{th} neighbor point of Q (e.g., Q_0 is Q itself, Q_{-1} is the previous point whilst Q_{+1} is the next point); w_k is the corresponding weight determined by a Kernel smooth function using the aforementioned radius R as the Kernel bandwidth, which is

$$w_k = \begin{cases} exp(-\frac{d(Q_0Q_k)^2}{2R^2}) & d(Q_0Q_k) < R \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

The detailed computing procedure of map matching is shown in Algorithm 4.1, including six main steps: (1) select candidate road segments, (2) calculate the point-segment distance, (3) normalize the distance as *localScore*, (4) compute the weight and calculate *globalScore*, (5) determine the map matching segment for each point based on *globalScore*, and finally (6) the projected position (x', y') of that GPS point in e is computed.

In the algorithm, since each GPS point considers only the neighboring road segments as a set of candidate segments to match (in terms of applying the R*-tree spatial data indexing technique), the candidate set size is significantly smaller than the total size of road segments in the network for real-life datasets. This makes the algorithm, besides having better matching quality, also working efficient with linear complexity on the total size of the GPS points, i.e., $O(n)$. In addition, the global map matching parameters (e.g., radius R and kernel width σ) need to be tuned in the experiment.

4.3.4 Data Compression

This step calls for data compression and reduction algorithms in data preprocessing. As already briefly discussed in Section 2.3.3 in the state-of-the-art chapter, trajectory data is usually generated continuously with a high sampling frequency; therefore the data will sooner or later go beyond the holding capacity of trajectory application systems. Giving another example, in one of the traffic scenarios we used as a test case, vehicle movement is tracked at one record per second. For such data, the size of the raw NME files[4] (the file format for recording GPS data) for recording one car trajectory during one week is approximately 2.4 megabytes. This kind of huge volume of trajectory data makes trajectory computing expensive, in terms of both the space complexity and the time cost. Therefore, *data compression* is a prerequisite technique to be applied for supporting the scalability of large trajectory applications.

Section 2.3.3 has already summarized a couple of existing data compression algorithms, especially the two representative solutions, i.e., the Douglas-Peucker extensions with the application of SED (synchronized Euclidian distance) [MdB04] and STTrace with additional metrics that are computed by using information like moving speed and direction [PPS06]. In this section, we discuss the detailed techniques to apply such compression techniques for achieving offline trajectory data compression.

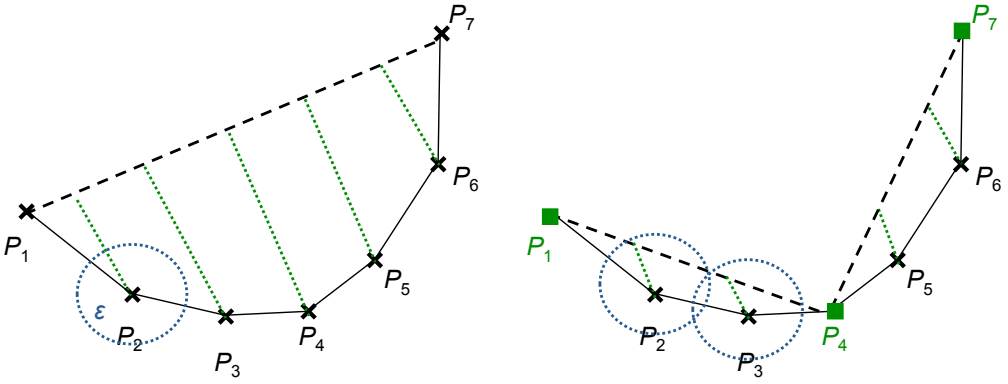
[4] `http://www.gpss.force9.co.uk/nmefiles.htm`

Algorithm 4.1: ***Map-Matching for Data Cleaning***
Goal: To map the GPS data $\{\langle x, y, t\rangle\}$ to the underlying road network. For each spatio-temporal point compute its corresponding road segment and the nearest road crossing, and if needed replace the point with a newly computed point on the road network.

Input: a GPS track, i.e., a sequence of spatio-temporal points $\{(x, y, t)\}$
the road network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Output: a cleaned GPS sequence $\{(x', y', t, segmentId)\}$ with the matched edge $segmentId$ and the corrected position (x', y')
begin
/* initialize: load GPS data and the road network */
ArrayList$\langle x, y, t\rangle$ $gpsList \leftarrow$ loadGPSList($gpsFile$);
Graph $roadNetwork \leftarrow$ getNetwork($networkFile$);
$gpsListNew \leftarrow$ new ArrayList$\langle x', y', t, segmentId\rangle$;
/* main procedure of map matching */
forall the $Q_i = (x, y, t) \in \mathcal{Q}$ **do**
// select candidate roads for Q_i (R*-tree)
$candidateSegs(Q_i) \leftarrow \{r_1^{(i)}, \cdots, r_n^{(i)}\}$; // select only neighboring road segments
// calculate point-segment distance, normalize it as localScore
compute the distance between point Q_i and $\forall r_j^{(i)} \in candidateSegs(Q_i)$;
choose the closest segment $min\{d(Q_i, r_j^{(i)})\}$ (by Formula 4.3);
normalize distance $localScore(Q_i, r_j^{(i)})$ $\forall r_j^{(i)} \in candidateSegs(Q_i)$ by Formula 4.4;
// calculate globalScore: (point, segment)
choose global points $(Q_{-N_1}, \cdots, Q_{+N_2})$ in radius R;
compute their Kernel smoothing weights by Formula 4.6;
compute the $globalScore(Q_i, r_j^{(i)})$ for $\forall r_j^{(i)} \in candidateSegs(Q_i)$ by Formula 4.5;
// compute Q' with road position
rank the computed $globalScore(Q_i, r)$
choose the highest score to match $segmentId$ for Q_i;
compute the corrected position (x', y') if needed ;
create a new GPS point $Q_i' \leftarrow (x', y', t, segmentId)$;
$gpsListNew$.add(Q_i'); //add the new point
return the cleaned version of GPS points $gpsListNew$;

By using the SED measurement, we apply the SED based Douglas-Peucker extension for trajectory data compression. The main idea is recursively computing the SED distances for each sequence's in-between points, and compare it with a given threshold ε to judge whether the point can be removed or not. As shown in Figure 4.12-(a), we compute the SED values of all the points in the middle (i.e., $sed(P_1, P_i, P_7)$ for all $i \in [2, 6]$), and find the *most representative* data point P_4 as it has the largest SED values compared with other points (i.e., P_2, P_3, P_5, P_6); therefore, the algorithm keeps P_4 and recursively goes to the next loop for subsequence P_1P_4 and P_4P_7. Figure 4.12-(b) shows the later steps for computing the new SED values for in-between points in segment P_1P_4 and P_4P_7, i.e., $sed(P_1, P_2, P_4), sed(P_1, P_3, P_4)$ in P_1P_4 & $sed(P_4, P_5, P_7), sed(P_4, P_5, P_6)$ in P_4P_7, and discovers that all of these $sed < \varepsilon$, therefore, we remove all of these four in-between points (P_2, P_3, P_5, P_6) and the new compressed trajectory is just $P_1 \rightarrow P_4 \rightarrow P_7$. Thus, the final compressed trajectory data has only three spatio-temporal points instead of the original one

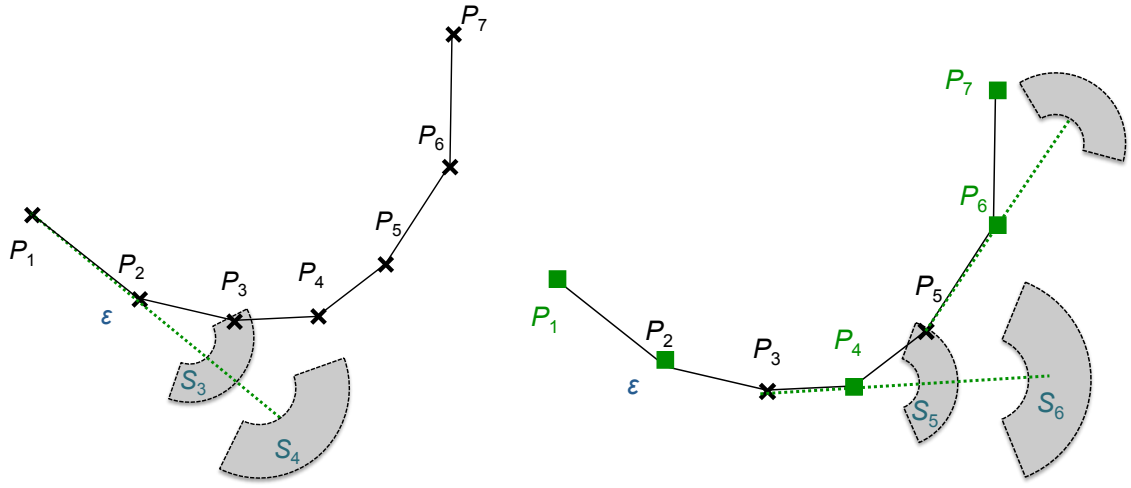
with seven points, achieving the compression ratio of 3/7, approximately 42.86%.



(a) Loop 1 (b) Loop 2

Figure 4.12: Data compression of DP extensions using SED

In addition, we adopt the idea of STTrace to build algorithms for compressing the trajectory data. This method applies the *safe area* to judge whether the points can be removed or not. This safe area is calculated by the moving object's moving *velocity* and *direction*. For example, in Figure 4.13-(a), based on the $speed_{min}, speed_{max}$ and previous direction P_1P_2, we can compute a *sector* area for future points P_3, P_4, we find the sector $P_3 \in S_3$ while $P_4 \notin S_4$, therefore P_3 can be removed whilst P_4 needs to be kept; similarly in Figure 4.13-(b), the algorithm can remove P_5 but need to keep P_6. Finally, we achieve the compressed trajectory data as $P_1 \rightarrow P_2 \rightarrow P_4 \rightarrow P_6 \rightarrow P_7$, in terms of the compression ratio as 5/7, approximately 71.4%, bigger than the previous SED based method.



(a) Determine P_3, P_4 (b) Determine P_5, P_6, P_7

Figure 4.13: Data compression using STTrace via direction & speed

The first SED based method is offline that recursively calculates SEDs for effective data compression; while the second STTrace based method is an online procedure where computing safe sectors is more complex rather than calculating SED. For trajectory online computing in

Chapter 6, we also design an online solution for SED based data compression, which further considers the non-spatial features than only the spatio-temporal distances.

In addition to the above four tasks of trajectory data preprocessing (i.e., data transformation, data cleaning, map matching, and data compression), there is an extra task of *data interpolation* in some trajectory applications. Different from very high-frequency of GPS data sampling in many applications where data needs to be compressed, the functionality of *data interpolation* is to deal with uncertain data points with low sampling frequency, designing certain interpolation techniques to approximate the missing tracking data. This thesis does not investigate the uncertainty analysis of trajectory data.

4.4 Trajectory Identification Layer

This trajectory computing layer analyzes the preprocessed data and extracts relevant non-overlapping spatio-temporal trajectories $\mathcal{T}_{spa}$ (corresponding to the *data model* in Figure 3.4) from the cleaned long sequence of mobility data. The central issue here is to determine reasonable identification policies, to identify *dividing points* (x_i, y_i, t_i) that can split the continuous GPS sequence and divide it into consecutive trajectories at appropriate positions. In other words, the objective of trajectory identification is to identify the starting point p_1 and the ending point p_n for each spatio-temporal trajectory. As shown in Figure 4.14, the input is the cleaned trajectory data, whilst the output is a set of spatio-temporal trajectories (the figure shows only two $\mathcal{T}_{spa}$).

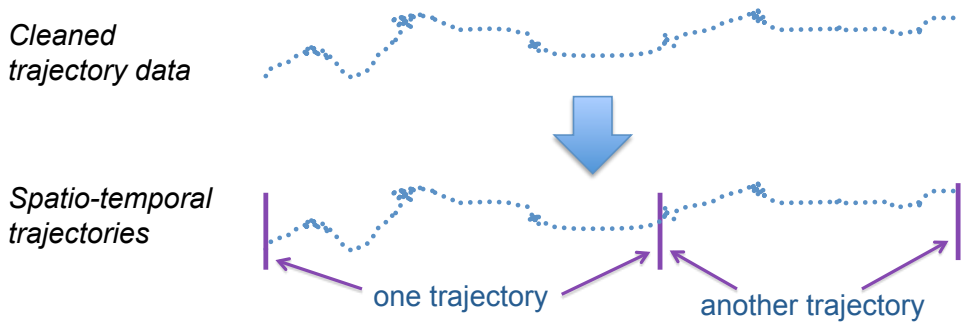


Figure 4.14: Trajectory identification (from cleaned movement data to $\mathcal{T}_{spa}$)

In this section, we present different types of trajectory identification policies that have been implemented in various trajectory scenarios in our study: *Raw GPS Gap*, *Predefined Time Interval*, *Predefined Space Extent*, and (semi-) automatic *Correlation-based Identification*.

4.4.1 Identification via Raw Gap

Policy 1 (Raw GPS Gap)**.** Divide the (x,y,t) sequence into several spatio-temporal trajectories $\mathcal{T}_{spa}$ according to the raw GPS gaps. There are two sub-conditions we can separately apply for trajectory datasets: one is the temporal gap, while the other is both temporal and spatial gaps.

(1) Given a large time interval $\Delta_{duration-large}$, for any two consecutive GPS records $p_i(x_i, y_i, t_i)$ and $p_{i+1}(x_{i+1}, y_{i+1}, t_{i+1})$, if the temporal gap $t_{i+1} - t_i > \Delta_{duration-large}$, then p_i is the ending point of current trajectory whilst p_{i+1} is the staring point of the forthcoming trajectory.

(2) Given both time interval $\Delta_{duration}$ and spatial distance $\Delta_{distance}$, for any two consecutive GPS records p_i and p_{i+1}, if the temporal gap $t_{i+1} - t_i > \Delta_{duration}$ and the spatial gap $\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} > \Delta_{distance}$, then p_i is the ending point of current trajectory whilst p_{i+1} is the staring point of the forthcoming trajectory.

This policy applies significant temporal (and spatial) gaps in GPS feed to logically separate two consecutive and adjacent spatio-temporal trajectories. This is because GPS-alike tracking devices are usually turned off (automatically or manually) when the object does not move for a long while (e.g., to save power). The first sub-policy exploits large temporal gaps $\Delta_{duration-large}$ to extract spatio-temporal trajectories. This is typically relevant for vehicle movement scenarios. For example, our dataset of 17,241 car GPS traces (2,075,213 GPS records) resulted in 83,134 spatio-temporal trajectories. The second sub-policy uses both temporal and spatial gaps, where the two parameters are determined by statistical analysis of GPS feeds (e.g., gap distribution, type of movement - vehicular, pedestrian etc).

4.4.2 Identification via Prior Knowledge

The second type of spatio-temporal trajectory identification policies are using some prior knowledge that is meaningful to certain trajectory applications. Such prior knowledge can be *predefined time intervals* or *predefined spatial extents* etc.

Policy 2 (Predefined Time Interval)**.** Divide the stream of GPS feed into several subsequences contained in given time intervals, e.g., hourly trajectory, daily trajectory, weekly trajectory, monthly trajectory, and even seasonally or yearly trajectory.

This policy allows us to meaningfully divide a GPS feed into periods for analyzing mobility behaviors. Short-term period is particularly relevant for human movements, e.g., analyzing the daily trajectory to distinguish people's daily movement behaviors between weekdays and weekends. Wild-life monitoring on the other hand usually requires longer-term trajectory behaviors such as monthly or seasonal patterns, in particular in the bird migration scenario.

Policy 3 (Predefined Space Extent)**.** Divide the stream of GPS feed into several subsequences according to a spatial criteria, e.g., fixed distance, geo-fenced regions, movement between predefined points in network constrained trajectories.

This policy allows us to divide a GPS feed in terms of somehow fixed spatial extents, such as: (1) fixed distance (e.g., each trajectory has 500 meters), (2) in a specific domain zone (e.g., EPFL campus, Lausanne downtown), where trajectories ought to be created according to their entering and leaving the zone, (3) between two given positions of crossings (e.g., finding all of the home-office trajectories of $John$, a research assistant at EPFL), (4) predefined routes in the networks (e.g., analyzing the congestions from all of the bus trips of $Bus33$).

4.4.3 Correlation-based Identification

Different from the previous prior knowledge (according to the trajectory application scenarios) driven policies for trajectory identification, we can also apply pure *correlation-based* methods to identify the spatio-temporal trajectory. In such case, no prior knowledge like time interval or spatial extent needs to be provided in advance. The division of the cleaned trajectory data feed is purely depending on the correlations between the data points $\langle x, y, t \rangle$. For example, we can apply conventional time series segmentation techniques for finding division points between two consecutive spatio-temporal trajectories.

Policy 4 (Time Series Segmentation)**.** Divide the stream of GPS feed into several subsequences according to a (semi-) automatic algorithm for segmenting time series, based on spatial (or temporal, or spatio-temporal) correlations.

Trajectory data can be considered as a special kind of time series, where the values are the locations $\langle x, y \rangle$ as time continues. Therefore, conventional time series segmentation algorithms can be applied for trajectory identification in this section as well as for trajectory structure in Section 4.5. Keogh et al. [KCHP04] categorize time series segmentation methods into three types, i.e., sliding windows, top-down, and bottom-up. The previous Policy 2 and Policy 3 can be roughly considered as the sliding window methods; while the top-down and bottom-up methods can bring much over-fragment of trajectories, which means it might find too many division points and generate a lot of small-length spatio-temporal trajectories. Therefore, such method is not good for the *trajectory identification* step, but could be more meaningful in the *trajectory structure* computing stage in Section 4.5.

Based on these designed policies for spatio-temporal trajectory identification, we can build an algorithm to automatically process the cleaned mobility data and generate the spatio-temporal trajectories, reaching the data model for computing and understanding mobility data. The detailed procedure for this trajectory identification layer is shown in Algorithm 4.2.

4.5 Trajectory Structure Layer - Segmentation

After identifying the spatio-temporal trajectories, the next task is to compute $\mathcal{T}_{spa}$'s internal structures, for building structured trajectory $\mathcal{T}_{str}$ that consists of meaningful episodes. As shown in Figure 4.15, the input spatio-temporal trajectory is segmented into a structured trajectory with seven episodes, among which, there are four *stops* and three *moves*. The core issue in *trajectory structure* is to group continuous GPS points into an episode, where the spatio-temporal points in each episode is in some sense homogeneous as sharing similar movement characteristics to form a single movement *activity/behavior*. We have implemented *velocity*, *density*, *orientation* and *time series* based algorithms for identifying episodes, which has been validated in many real life GPS trajectory datasets.

Algorithm 4.2: ***Spatio-temporal Trajectory Identification*** **(*Begin/End*)**
Goal: To divide the cleaned mobility data list *newGPSList* into several non-overlapping sub-series, and each one is representing a spatio-temporal trajectory $\mathcal{T}_{spa}$.

Input: a clean list *newGPSList*, a selected rule *SegmentRule*
Output: a list of spatio-temporal trajectories $\{\mathcal{T}_{spa}\}$, i.e., *spaTrajList*
begin
  `/* STEP 1: Cut the input list` *newGPSList* `into a set of sub-series` *subSeriesSet* `according to the selected rule. */`
  switch *the value of SegmentRule* **do**
    case *Policy1 Raw_GPS_Gap*
      cut *newGPSList* each time there is a big temporal gap with no record (e.g., 5 hours for a vehicle trajectory).
    case *Policy2 Predefined_Time_Interval*
      cut *newGPSList* into sub-series contained in given time intervals, e.g., morning trajectory, daily trajectory, weekly trajectory.
    case *Policy3 Predefined_Space_Extent*
      cut *newGPSList* according to a spatial criteria: e.g., fixed length, crossing the borders of a partition of the space into regions, passing by predefined points for network constrained trajectories.
    case *Policy4 Time_Series_Segmentation*
      cut *newGPSList* according to a (semi-) automatic algorithm for segmenting time series, based on (spatial or temporal) correlations.
  `/* STEP 2: For each sub-serial, compute a new starting point` *begin* `and new ending point` *end*`. */`
  forall the $subSerial \in subSeriesSet$ **do**
    $begin \leftarrow$ computeBegin($subSerial$);
    $end \leftarrow$ computeEnd ($subSerial$);
    $\mathcal{T}_{spa} \leftarrow$ getRawTrajectory ($subSerial, begin, end$);
    $spaTrajList$.add($\mathcal{T}_{spa}$);
  return *spaTrajList*;

4.5.1 Trajectory Segmentation as Stop Discovery

Trajectory structure in essence is segmenting the sequence of spatio-temporal points in $\mathcal{T}_{spa}$ into many episodes. In many trajectory applications, such segmentation tasks can be reformulated as the problem of stop discovery. This is because most of the trajectory applications focus on two kinds of meaningful episodes, i.e., *stop* and *move* [SPD+08]. As shown in Figure 4.15, the structured trajectory has 4 stops and 3 moves.

(1) Benefits of Stop Discovery

In the hybrid spatio-semantic model discussed in Section 3.5, we have already clarified a couple of advantages of using *episodes*, such as *encapsulation of semantic trajectory concepts like stop/move*, *automatic computation*, *enabling of data compression* etc. Similarly, these are the advantages of stop discovery in analyzing trajectory data. For achieving a better data abstraction on the spatio-temporal trajectory data (i.e., the sequence of GPS points $\langle x, y, t \rangle$), we need to divide it into a sequence of episodes, and pick up the important episodes – *stops*;

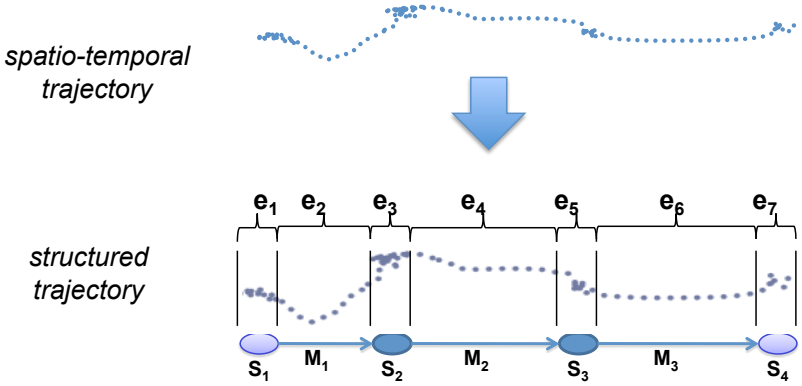


Figure 4.15: Trajectory structure (from $\mathcal{T}_{spa}$ to $\mathcal{T}_{str}$)

once the *stops* are identified, we can easily identify the *moves* in between. There are several remarkable features for the notation of *stops* in trajectory data abstraction:

- *Easily understandable*: A sequence of stops can provide a better abstracted view for understanding mobility trace, rather than the original sequence of $\langle x, y, t \rangle$ points. As shown in Figure 4.15, only important places (e.g., S_1 and S_4 are at *home*, S_2 at *EPFL* and S_3 at a shopping market *Coop*) need to be highlighted for this trajectory.

- *Efficient data compression*: Instead of keeping the whole mobile tracking points, mobility data can be represented and restored in terms of a sequence of stops. As shown in Figure 4.15, tends and hundreds of GPS points can be compressed and displaced by four stops, actually only three stop places as S_1 and S_4 are in the same place (e.g., *home*).

- *Automatic stop computation*: These important parts of trajectories (*stop* episodes) can be computed automatically and efficiently, based on the relevant trajectory data discretization/segmentation methods, which are the focus of trajectory computing.

The definition of *"stop"* is typically emphasized as "not moving". However, real-life stops are not purely depending on whether it is moving or not. For example, short time of stilling like a congestion in the middle of a road is not a meaingful stop; shopping with wondering around in a store can be considered as a stop even the user is moving. Therefore, we define stops as *the important places where a trajectory has passed and stayed for a reasonable time duration*. Stops can help people summarize trajectory and provide better understanding of mobility. Thus, many studies do not distinguish among "stops", "hotspots", "important/meaningful/interesting parts" of trajectories etc. The focus of this section is on designing robust segmentation algorithms to automatically discover stops in heterogeneous trajectories of various moving objects.

In addition to the basic advantages, stop has some additional characteristics in several trajectory applications, which require more robust and efficient stop discovery algorithm.

(2) Shared Stops vs. Personalized Stops

Shared stops are the common places where many trajectories of different moving objects pass and stay for a while (e.g., a *shopping mall* like Coop), whilst *personalized stops* are individual places where user stays for his/her own purpose (e.g., *home*). For example, there are three

trajectories in Figure 4.16, where each one has four stops. In addition, there are three common places including *EPFL*, *Coop*, and *SportCenter* being correlated by these three trajectories as shared stops; whilst the other six stops are personalized stops that belong to each trajectory separately, which could be their individual *home* for instance.

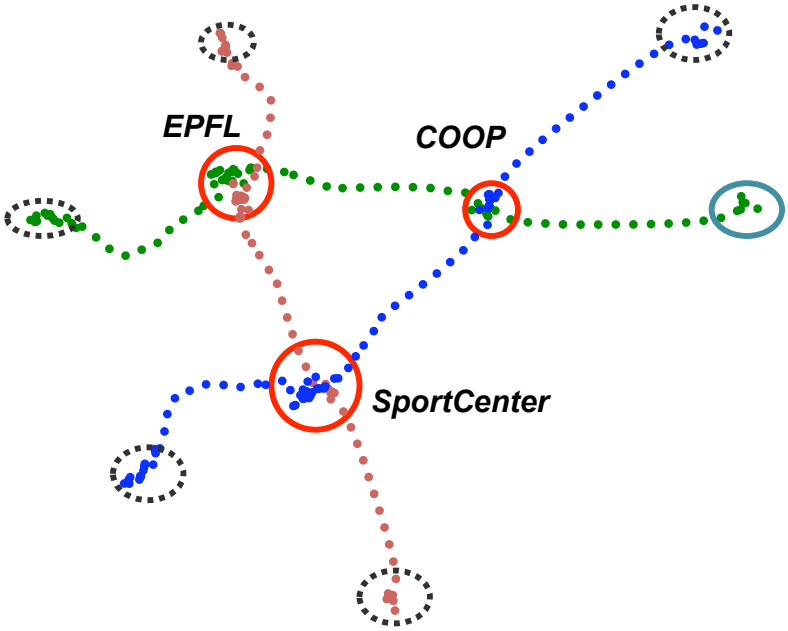


Figure 4.16: Shared stops vs. personalized stops

Therefore, a robust discovery algorithm needs to consider both shared stops and personalized stops, and identify the difference between them. For designing stop discovery algorithms, personalized stops usually can be discovered from single trajectory, whist shared stops can be extracted from a group of trajectories that belong to different moving objects.

(3) Hierarchical Stops (Generic Stops vs. Concrete Stops)

Another challenging problem in stop discovery is the granularity of the stops, i.e., determining the suitable stop size. Stop discovery in trajectory might be meaningless when it produces too many stops or too few stops. Each stop should have a suitable size, which means it should consist of a reasonable set of GPS points with very high correlations. Some applications require large stops whilst others may prefer smaller ones.

- *Generic Stops*: Taking a trajectory of recording students' movement in EPFL for example, a student came to *Classroom1* for taking the Database course, and then went to the Algorithm course in *Classroom2*. Precisely, there are two small stops in *Classroom1* and *Classroom2*. However, application might be more interested in the generic stop (i.e., EPFL) when analyzing student's daily behavior, where the detailed locations of individual classrooms are nonsignificant. To handle this problem, we can merge small and nearby stops into a more generic one.

- *Concrete Stops*: In contrast, some applications care more about concrete stops rather than the generic ones. For example, people visit a big commercial centre that inside has many

interesting places such as restaurant, supermarket, cinema etc. For trajectory applications, the commercial center is too generic. Applications want to know exactly which place people visited, which can reflect the exact movement behaviors (e.g., stay in restaurant for dinner in the first floor or do some exercise in a gym in the second floor, or something else). Therefore, one large stop needs to be split into many concrete and meaningful ones in some applications.

To identify both *generic* and *concrete* stops, and set up suitable stop size, a *hierarchy-based* stop discovery approach is required. Thus, stops can be represented in different levels. The stop in the lower level usually has small size with more detailed information; whilst stops in the higher level are in a big area with more generic positioning. Consequently, trajectories can be abstracted in terms of stop sequences at different levels. Figure 4.17 sketches the refinement of discovered stops (left), in terms of a tree-based stop hierarchy established (right), by using a *split-merge* approach, i.e., the big stop can be divided into small ones while the adjacent small stops can be merged together as a big stop.

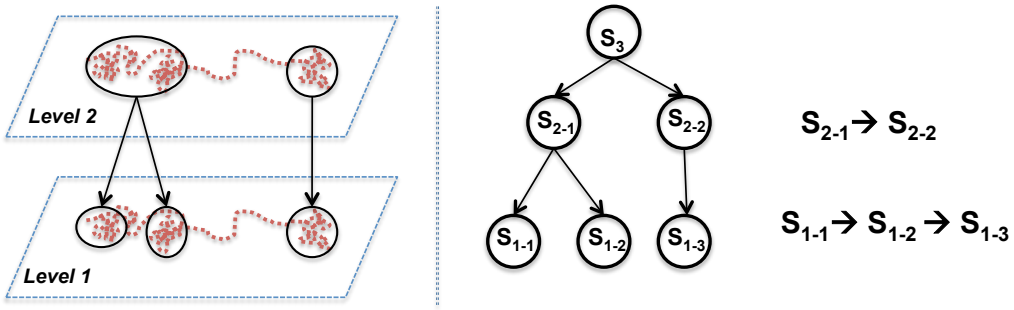


Figure 4.17: Building tree-based hierarchal *stops*

(4) Stop Discovery in Heterogeneous Trajectories

Existing works like [ABK+07, PBKA08, ZKS09] typically only focus on discovering stops on homogeneous trajectories, which means there are a couple of common assumptions required for such GPS datasets: (1) the moving object is identical like a taxi, a truck, or a ship, therefore the speed/acceleration of trajectory is more or less stationary; (2) the GPS sampling frequency is stationary, e.g., the time gap between the continuous GPS points is almost fixed (e.g., every one second or several minutes). Therefore, these stop discovery algorithms can not work properly for GPS data with big recording gaps. However, the real-life trajectories are far more heterogeneous. Taking people trajectories from smartphones for instance, GPS from such people-centric tracking devices are much heterogeneous with ambiguity, e.g., (1) people can take many different kinds of transportation modes, such as *walking*, *cycling*, *driving*, and *public transports like metro or bus*, therefore, GPS of people trajectories are much more irregular compared with the tracks of vehicles like taxi and truck. (2) due to GPS signal loss during people indoor activities[5] and the limited power issue of smartphones, GPS sampling of people trajectories are very sparse. Therefore, the objective of a robust stop discovery algorithm is to work properly for the heterogeneous trajectory datasets as well.

[5]From Nokia report, people spent nearly 80% of their time indoor without GPS recordings.

In the following subsections, we address the detailed trajectory structure (also called "stop discovery") algorithms proposed in this thesis, including the velocity-based, density-based, and further advanced stop discovery algorithms for trajectory segmentation.

4.5.2 Velocity based Trajectory Structure

In this approach, we focus on two kinds of episodes (i.e., *stops* and *moves*). The idea is to determine whether a GPS point $p(x, y, t)$ belongs to a stop episode or a move episode by using a speed threshold (Δ_{speed}). Hence, *if the instant speed of p is lower than* Δ_{speed}*, it is a part of a stop, otherwise it belongs to a move.* Figure 4.18 traces the speed evolution of a vehicle, showing how stops can be determined by a given stop threshold Δ_{speed}. Besides Δ_{speed}, we also use a second parameter – *minimal stop time* τ – in order to avoid false positives, i.e., short-term *congestions* with low velocity should not be a stop episode as shown in the figure.

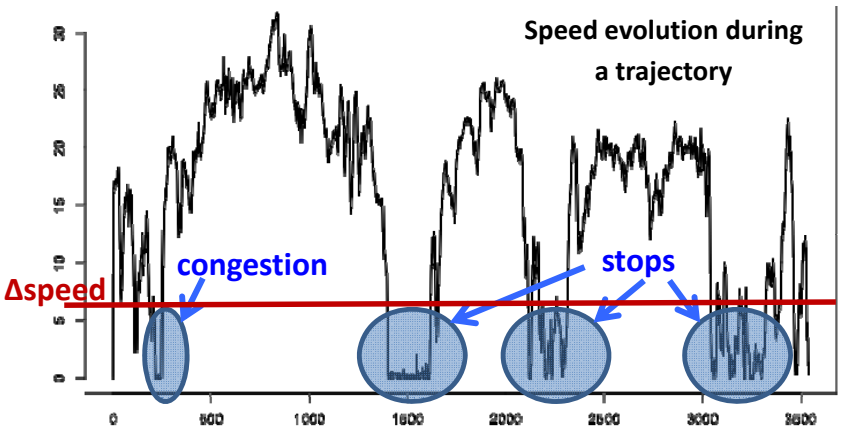


Figure 4.18: Velocity-based stop identification

Determining a suitable value for Δ_{speed} is a challenging problem: *if* Δ_{speed} *is too high, many stops appear; on the contrary, if* Δ_{speed} *is too low, probably no stops are computed.* Figure 4.18 simply shows a constant Δ_{speed} applied all across the trajectory. This is not practical in real-world scenarios, where the value of Δ_{speed} should rather be flexible according to the context of the moving object. For example, vehicles with different levels of performance (bicycles or motor cars), different road networks (on a highway or a secondary road path), different weather conditions (sunny or snowy days) call for diverse speed thresholds. However, it is not always easy to get these contextual information for the mobility datasets. To overcome this problem, we design a generic method for determining Δ_{speed}, based on the class of moving objects being monitored (which is available) and the real-time underlying movement area. This can be applied to most real-life GPS feeds.

Definition 4.3 (Dynamic Velocity Threshold - Δ_{speed})**.** For each GPS point $Q(x, y, t)$ of a given moving object (obj_{id}), the Δ_{speed} is dynamically determined by the moving object (by using $\overline{objectAvgSpeed}$ – the average speed of this moving object) and the underlying context (by $\overline{positionAvgSpeed}$ – the average speed of most moving objects in this position $\langle x, y \rangle$); i.e., $\Delta_{speed} = min\{\delta_1 \times \overline{objectAvgSpeed}, \delta_2 \times \overline{positionAvgSpeed}\}$, where δ_1 and δ_2 are coefficients.

In this definition, $\overline{objectAvgSpeed}$ is easy to calculate as the average speed of the moving object. Regarding $\overline{positionAvgSpeed}$, we need to approximate it by using space division. We divide the mobility space into regular cells (or directly using the available landuse data) and calculate the average speed in each cell $\overline{cellAvgSpeed}$ as the contextual information. For the application of network-constrained trajectory data, we can apply the speed condition on the underlying network (e.g., the average passing speed of the nearest road crossing $\overline{crossingAvgSpeed}$ and the average passing speed of the map matched road segment $\overline{segmentAvgSpeed}$), instead of the $\overline{cellAvgSpeed}$. Algorithm 4.3 provides the pseudocode to determine Δ_{speed}. We analyze sensitivity of the coefficients δ_1 and δ_2 (e.g., 30%) through experiments.

Algorithm 4.3: getDynamicΔ_{speed} (gpsPoint, obj_{id}, δ)

input : gpsPoint $p = (x, y, t)$, moving object obj_{id}
output: dynamic speed threshold Δ_{speed}
get the average speed of this moving object obj_{id}: $\overline{objectAvgSpeed}$;
if *network-constrained trajectory* **then**
  get the average speed of the nearest road crossing to p: $\overline{crossingAvgSpeed}$;
  get the average speed of the map matched road segment of p: $\overline{segmentAvgSpeed}$;
  $positionAvgSpeed \leftarrow min\{\overline{crossingAvgSpeed}, \overline{segmentAvgSpeed}\}$
else
  get the average speed of the cell that (x,y) belongs to: $\overline{cellAvgSpeed}$;
  $positionAvgSpeed \leftarrow \overline{cellAvgSpeed}$
compute the dynamic speed threshold by Definition 4.3;
return Δ_{speed}

In some scenarios, GPS tracking data have available instant speed (s) values that are captured by the positioning devices. We can use them directly for calculating Δ_{speed} and identifying stops; otherwise, s needs to be approximated by the average speed between the previous spatio-temporal point $(x_{i-1}, y_{i-1}, t_{i-1})$ and the next one $(x_{i+1}, y_{i+1}, t_{i+1})$ (see Formula 4.7). This is possible as GPS data is usually sampled frequently (e.g., a few minutes or even every second) like the scenarios of vehicles in our case study.

$$s_i = \frac{\|\langle x_{i+1}, y_{i+1}\rangle - \langle x_{i-1}, y_{i-1}\rangle\|_2^2}{t_{i+1} - t_{i-1}} = \frac{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}}{t_{i+1} - t_{i-1}} \tag{4.7}$$

After the calculation of *instant speeds* (s) and *speed threshold* (Δ_{speed}), the velocity-based algorithm for stop discovery can tag each GPS point $\langle x, y, t\rangle$ with 'M' or 'S' by the condition whether the instant speed s_i is bigger than the velocity threshold Δ_{speed_i} or not. Stops and moves are then computed based on contiguously located 'M'/'S' tags, which means: *a continuous sequence of $\langle x, y, t\rangle$ points having all 'M' tags is integrated into a single move, whilst, a continuous sequence of $\langle x, y, t\rangle$ points, all with 'S' tags, is integrated into a single stop.* In addition to δ for computing Δ_{speed}, we need another parameter τ (*the minimum stop time*) to determine stop episode, i.e., only consecutive points holding low-speed with more than τ time interval can be joined together as a stop. With τ, short term congestions (as "false negative") can be avoided when computing stops. Algorithm 4.4 provides the pseudocode for determining *velocity-based trajectory structure*: firstly, we compute the instant speed if not available from

GPS devices; secondly, we compute the dynamic Δ_{speed} (using Algorithm 4.3) and annotate the GPS point with 'M' or 'S' tag; finally, stops and moves are computed with the consecutive same tags, using another precondition on minimal stop time τ (the candidate stop points with duration less than τ are considered as congestion and changed to the move points).

Algorithm 4.4: ***Velocity-based trajectory structure***

Input: a raw trajectory $\mathcal{T}_{raw} = \{p_1, p_2, \cdots, p_n\}$
Output: a structured trajectory $\mathcal{T}_{str} = \{e_1, e_2, \ldots, e_m\}$ where e_i is a tagged trajectory episode (stop $\mathcal{S}$ or move $\mathcal{M}$)

```
begin
    /* initialize: calculate GPS instant speed if needed */
    ArrayList⟨x, y, t, tag⟩ gpsList ← getGPSList(T_spa);
    if no instant speed from GPS device then
        compute GPS instant speed s_i for all p_i = (x, y, t) ∈ gpsList;
    /* episode annotation: tag each GPS point with 'S' or 'M' */
    forall the p_i = (x, y, t) ∈ gpsList do
        // get dynamic Δ^(i)_speed by Algorithm 4.3
        Δ^(i)_speed ← getDynamicΔ_speed (p, obj_id, δ);
        // tag GPS point as a stop point 'S' or a move point 'M'
        if instant speed s_i < Δ^(i)_speed then
            tag current point p_i(x, y, t) as a stop point 'S';
        else
            tag current point p_i(x, y, t) as a move point 'M';
    /* compute episodes: grouping consecutive same tags*/
    forall the consecutive points with the same tag 'S' do
        // compute stop episode
        get the total time duration t_interval of these points;
        if t_interval > τ the minimal possible stop time then
            stop ← (time_from, time_to, center, boundingRectangle);
            T_str.(stop, 'S'); // add the stop episode
        else
            change the 'S' tag to 'M' for all these points; // as "congestion"
    forall the consecutive points with the same tag 'M' do
        // compute move episode
        move ← (stop_from, stop_to, duration) // create a move episode
        T_str.(move, 'M'); // add the move episode
    return the structured trajectory T_str;
```

Algorithm 4.4 has linear complexity on the size of GPS feed, together with the linear complexity on the size of road segments in the underlying network. It currently performs two data scans while tagging points and grouping consecutive points for the episodes. However, in our experimental implementation, we combine the two scans together to reduce computing time for better performance.

4.5.3 Density based Trajectory Structure

Using only velocity for identifying stops is not enough for some scenarios. For example in bird migration application, we want to discover stops of bird foraging. Some birds, like water-birds, when they are looking for food, can fly at high speed, but keep inside a small area. Another example is in traffic scenarios, the driver can drive the car very fast around a block area and looks for a parking slot. These kinds of stops cannot be detected by the previous velocity-based algorithm as the speed is always reasonably higher than the threshold. Therefore, we design an alternative solution – *density-based* stop discovery, which considers not only the speed but also the maximum diameter that the moving object has traveled during a given time duration. For this algorithm, we need to define density areas for extracting stop or move episodes.

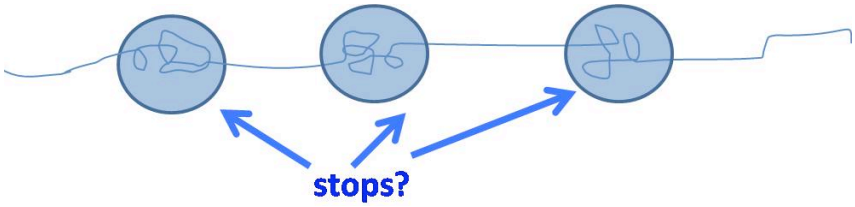


Figure 4.19: Density-based stop identification

Definition 4.4 ($\mathcal{A}_{density}$ - Density Area)**.** Given a cleaned GPS sequence $\{\langle x_i, y_i, t_i\rangle\}$, a maximum distance σ, and a time duration τ, a density area $\mathcal{A}$ is a sub-series of GPS points $\{\langle x_{i1}, y_{i1}, t_{i1}\rangle, \ldots, \langle x_{im}, y_{im}, t_{im}\rangle\}$ with two conditions:

1) For any two different tuples of the density area, if they are temporally distant by less than τ then they are spatially distant by less than σ, i.e. $\forall \langle x_{ia}, y_{ia}, t_{ia}\rangle, \langle x_{ib}, y_{ib}, t_{ib}\rangle \in \mathcal{A}$, $\|t_{ib} - t_{ia}\| \leq \tau \Rightarrow \|\langle x_{ia}, y_{ia}\rangle - \langle x_{ib}, y_{ib}\rangle\| \leq \sigma$

2) For the last (first) tuple of GPS that is just before (after) the density area, say $\langle x_b, y_b, t_b\rangle$ ($\langle x_a, y_a, t_a\rangle$), there exists a point inside the density area, which is temporally distant by less than τ and spatially distant by more than σ, i.e., $\exists \langle x', y', t'\rangle \in \mathcal{A}$ $\|t' - t_b\| \leq \tau$ and $\|\langle x', y'\rangle - \langle x_b, y_b\rangle\| > \sigma$ ($\|t_a - t'\| \leq \tau$ and $\|\langle x_a, y_a\rangle - \langle x', y'\rangle\| > \sigma$)

Thus, this thesis further proposes a robust clustering based stop discovery algorithm, based on the extension of conventional DBSCAN – the well-cited spatial data clustering method, by introducing an additional time duration constrain to discover clusters. We call this density-based stop discovery algorithm "TrajDBSCAN".

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a well-cited spatial clustering algorithm proposed by Ester et al. in 1996 [EKSX96], which looks for core points in order to start a cluster and then expand the clusters by adding density-reachable points. There are a couple of fundamental definitions about such "density-reachable" as follows (see Figure 4.20 for the sketched reference):

- $distance(p_i, p_j)$ – the spatial distance (usually the Euclidean distance) between point p_i and point p_j

- $neighbor_{\varepsilon}(p)$ – the neighborhood points that closed to p (within distance less than ε)
- *core point* p – a point p with at least $minPts$ points within a radius ε, where $minPts$ is the minimum number of points required to form a cluster
- *directly density-reachable* – a point p is directly density-reachable from point q w.r.t. ε, $minPts$ if it holds two conditions: (1) $p \in neighbor_{\varepsilon}(q)$ and (2) $|neighbor_{\varepsilon}(q)| \geq minPts$
- *density-connected* – a point p is density-reachable from a point q w.r.t. ε, $minPts$ if there is a chain of points $p_1, \cdots, p_n$, where $p_1 = q, p_n = p$ such that p_{i+1} is directly density-reachable from p_i
- *density-connected* – a point p is density-connected to a point q w.r.t. ε, $minPts$ if there is a point p_j such that both, p and q are density-reachable from p_j w.r.t. ε, $minPts$

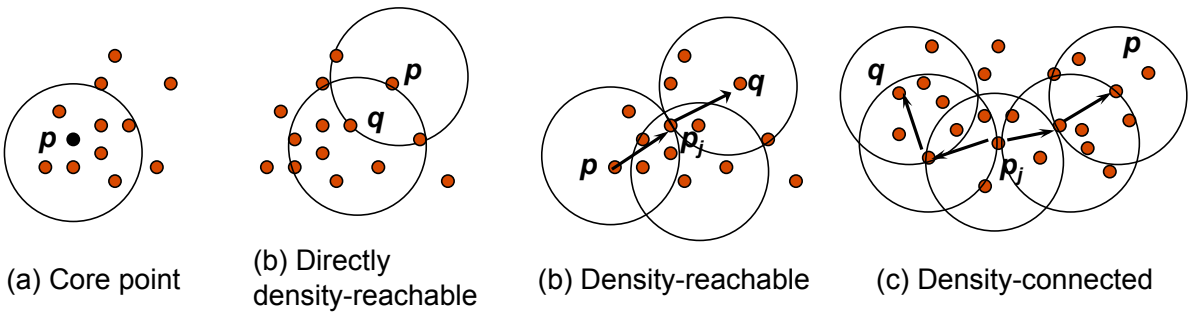


Figure 4.20: The main notions in DBSCAN

Based on these density-based notions, a cluster in DBSCAN is defined as a maximal set of density-connected points. DBSCAN can start with an arbitrary point p that has not been visited and retrieves all points that are density-reachable from p w.r.t. ε and $minPts$. If p is a core point then a cluster formed; otherwise, if no points are density-reachable from p DBSCAN visits the next point in the database. Clusters can be expanded by the notion of density-reachable. The main advantages of DBSCAN are: (1) no prior knowledge is required like the number of clusters (e.g., the k-mean algorithm) needs to be given in advance, thus it is suitable for stop discovery as we do not know the number of stops in advance; (2) it supports arbitrary cluster shapes ("stop" in our case can be any shape; both big and small are acceptable according to the trajectory scenarios); (3) it works efficiently on large spatial databases and is insensitive to the visiting order of the data points in the database.

For extending DBSCAN to discover stops in trajectory data, we design TrajDBSCAN following the DBSCAN principles with certain modifications for trajectories. Since we want to find stops in a single trajectory with respect to both space and time factors, the main concern is that stop must be a continuous sub-sequence of a trajectory; hence, a stop should contain only time consecutive points. We also need to overcome the problems like GPS point absence because of signal loss or low sampling rate. The method looks for core points and then expands them by aggregating other points in the neighborhood. The main distinguishing design principles are:

- The *neighborhood* of a point p is determined not only by the spatial distance ε_{space}, but also the temporal information, i.e., it only considers the temporal linear neighborhood.
- In contrast to using the minimal number of points $minPts$ for fostering clusters in DBSACN, TrajDBSCAN determines whether p is a core point or not by the minimum stop duration $minTime$, not the $minPts$.

Given a trajectory $\mathcal{Q}$, the maximum spatial distance threshold ε_{space} and the minimum time duration threshold ε_{time}, the objective of TrajDBSCAN is to find the sub-trajectory (i.e., the stops) such that any consecutive points in this sub-trajectory with spatial distance less than ε_{space}, the time duration of these points should be greater than ε_{time}; the initial stops can be expanded to generate bigger stops from overlapping small stops. Thus, in order to find stops, we start with a certain point Q_k in $\mathcal{Q}$. After that, we find a sub-trajectory (an initial stop) that is a set of consecutive points $\mathcal{S} = \{Q_m, Q_{m+1}, \cdots, Q_k, \cdots, Q_n\}$ and satisfies two spatio-temporal constraints: (i) $\forall i, m \leq i \leq n, t_m \leq t_k \leq t_n : distance(Q_k, Q_i) < \varepsilon_{space}$, and (ii) $|t_n - t_m| \geq \varepsilon_{time}$.

Therefore, for each core point Q_k in the set $\mathcal{S}$, it has an $esp\_neighbors$ that is denoted as $neighbor_\varepsilon(Q_k) = \{Q_a, Q_{a+1}, \cdots, Q_k, \cdots, Q_b\}$; the spatial distance between Q_k and any point in $neighbor_\varepsilon(Q_k)$ is less than the given spatial distance threshold ε_{space} and the total time duration $t_b - t_a$ is larger than the time duration threshold ε_{time}. Each core point may have a particular $neighbor_\varepsilon(Q_k)$ (considered as the *directly reachable* in DBSCAN) and these neighbors possibly can overlap each other. Therefore, we can apply the expanding idea of DBSCAN (i.e., expanding *the reachability*) to merge the overlap neighbors so as to create a big stop.

Algorithm 4.5 describes the detailed procedure of such density-based stop discovery method. Firstly, we initialize an empty set of stop episodes as a structured trajectory needs to be computed (line 1); then the method iterates through the trajectory and processes the data points that have not yet been processed (from line 2). In line 5, the $neighbor_\varepsilon$ of the point is computed and checked with the ε_{time} constraint to examine if the point is a core-point (line 6). If a core-point is found, a new cluster (stop) is created (line 7). Afterward, the method aggregate other points in the neighborhood to expand the cluster (line 8) by the *expandStop* function, which recursively checks the points in the cluster to find the possible neighborhood cluster to expand the stop. Finally, the stop cluster is added to the output stop set (line 10).

4.5.4 Other Trajectory Structure Methods

In addition to the previous *velocity* and *density* based stop discovery methods, we can apply relevant time-series analysis methods for computing trajectory structure. During this thesis work, we designed a time series method for network-constrained trajectory data modeling [Yan10], i.e., the Traj-ARIMA (*Autoregressive Integrated Moving Average*) model. The initial objective of Traj-ARIMA is to provide the functionality of velocity fitting and prediction. The original procedure of ARIMA modeling provided by Box-Jenkins is an interactive three-stage process, including *model selection*, *parameter estimation*, and *model checking* [BJR94]. For our case, we do two more additional tasks, namely the first stage of *data preparation* and the final stage of *forecasting* [MWH98]. We summarize these five steps in this section, and more details can be found in [Yan10]; the important symbols used in the Traj-ARIMA model is listed and described

Algorithm 4.5: Density-based stop discovery - TrajDBSCAN

```
input  : Q = {Q_1, ..., Q_n} //trajectory
         ε_time //the stop minimum time
         ε_space //the neighborhood distance for fostering a stop
output: T_str structured trajectory as a set of stops w.r.t ε_time and ε_space
T_str = ∅
foreach point Q_i in Q do
    if Q_i is unprocessed then
        mark Q_i as processed;
        N = neighbor_ε(Q_i, ε_space);
        if duration(N) > ε_time then
            S = new stop; // as next stop discovered
            S = expandStop (Q_i, N, S, ε_space, ε_time);
            T_str = T_str ∪ S
return T_str;

function expandStop (Q_i, N, S, ε_space, ε_time)
S = N;
initialize the stop as the first N foreach point Q_j in N do
    if Q_j is unprocessed then
        mark Q_j as processed;
        N' = neighbor_ε(Q_j, ε_space);
        if duration(N') > ε_time then
            S = S ∪ N'
    return S;
```

in Table 4.1. After these five steps of Traj-ARIMA analysis, we can further apply it for stop discovery – *if the forecasted speed is very different from the real speed, there might be a stop or change happening*; therefore the division point is found for trajectory structure.

1) **Data Preparation** – Data preparation focuses on transforming the raw GPS tracking data $\langle x, y, t\rangle$ into trajectory speed time series $\langle speed, t\rangle$. From the plot of the original trajectory speed time series and its autocorrelation function (ACF), shown in Figure 4.21-(a) and (c), we observe it has long lags and needs to be stationarized. The *differencing* operation (i.e., $\delta = x_{i+1} - x_i$) is a key solution to reduce the negative correlation for stationary. After one order of differencing, we get a new time series data, shown in Figure 4.21-(b) and (d), where the ACF lag is much shorter. Thus, we can observe that the new sequence (with 1 step of differencing) is much more stationary than the initial one.

2) **Model Identification** – After a time series has been stationarized by differencing, the next step is model selection that aims to determine the order of AR (p) and MA (q) in fitting an ARIMA(p,d,q) model. From the partial autocorrelation (PACF) plot of the differenced series in Figure 4.21-(f), we observe it "cuts off" at lag 2, which means the correlation is significant at lag 2 and nonsignificant from any higher order lags (> 2), therefore, we can tentatively identify the order of AR (p) is 2. From the differenced ACF plot, we identify the

Symbol	*Description or Definition*
$\langle x, y, t\rangle$	raw mobility data, i.e., the spatio-temporal point
$\langle s, t\rangle$	the speed time series, where s is calculated instant speed
ACF(τ)	autocorrelation of a time series $\{x_t\}$ ACF(τ) = $\frac{E[(x_t-\mu_X)(x_{t+\tau}-\mu_X)]}{\sigma_X^2}$, where x_t is $\mathcal{N}(\mu_X, \sigma_X^2)$
PACF(τ)	partial autocorrelation at lag τ is the autocorrelation between x_t and $x_{t+\tau}$ that is not accounted for by lags 1 through τ-1. PACF(τ) = $corr(x_t - \mathcal{P}(x_t\|x_{t+1}, \ldots, x_{t+\tau-1}), x_{t+\tau} - \mathcal{P}(x_{t+\tau}\|x_{t+1}, \ldots, x_{t+\tau-1}))$, where $\mathcal{P}$ is the best linear projection by minimizing the mean squared error
AR(p)	$x_t = \phi_1 x_{t-1} + \ldots + \phi_p x_{t-p} + \epsilon_t = \sum_{i=1}^{p} \phi_i x_{t-i} + \epsilon_t$
MA(q)	$x_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \ldots + \theta_q \epsilon_{t-q} = \epsilon_t + \sum_{j=1}^{q} \theta_j \epsilon_{t-j}$
ARMA(p,q)	$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q}$ $= \sum_{i=1}^{p} \phi_i x_{t-i} + \epsilon_t + \sum_{j=1}^{q} \theta_j \epsilon_{t-j}$
$\mathcal{N}(\mu_{1\times n}, \sigma^2 V(\phi, \theta))$	the joint Gaussian distribution of $X = \langle x_1, \cdots, x_n\rangle$, where μ is the mean of x, σ^2 is the variance of ϵ, and $\sigma^2 V(\phi, \theta))$ is the variance of X as a function of the ϕ and θ parameters
ARIMA(p,d,q)	$(1 - \sum_{i=1}^{p} \phi_i B^i)(1 - B)^d \; x_t = \epsilon_t (1 + \sum_{j=1}^{q} \theta_j B^j)$, where B is the backshift operation $B(x_t) = x_{t-1}$, and d is the order of deferencing

Table 4.1: Symbol description and definition in TrajARIMA

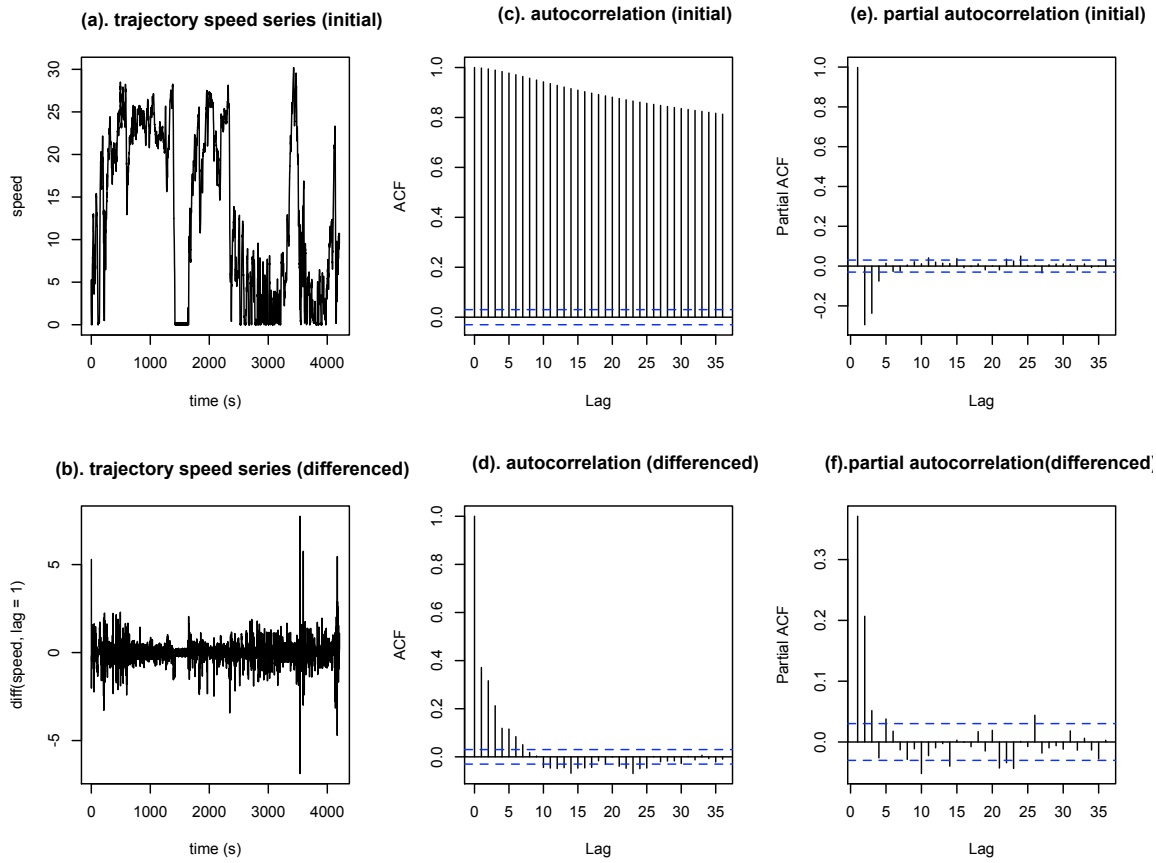


Figure 4.21: Speed time series ACF/PACF (original vs. differenced)

order of MA (q) is 6 as it "tails off"after lag 6. Therefore, a reasonable ARIMA model for the trajectory speed time series in Figure 4.21 is ARIMA(2,1,6).

3) **Parameter Estimation** – After determining the orders (i.e., the values of p, d, q) of ARIMA model, the next step aims at analyzing the detailed sequential values and estimating the model coefficients (i.e., ϕ_i and θ_j in the TrajARIMA definition in Table 4.1) that is able to provide the best fit of the data. We apply the MLE (Maximum Likelihood Estimation) method that is largely applied in parameter estimation in statistics, in particularly for achieving better estimation results in time series data. According to MLE definition, the problem is reformulated as: *given the time series* $\{x_1, \ldots, x_n\}$*, estimate the best parameter values* (*i.e., the coefficients* ϕ, θ *in TrajARIMA,* μ *is the mean of* x*,* σ^2 *is the variance of the white noise* ϵ). According to the ARMA definition (i.e., $x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p}$), $X = \langle x_1, \cdots, x_n \rangle$ has a joint Gaussian distribution $\mathcal{N}(\mu_{1\times n}, \sigma^2 V(\phi, \theta))$ (see Table 4.1). Therefore, the MLE problem for estimating parameters in TrajARIMA is shown in Formula 4.8.

$$\begin{aligned} \ell(\phi, \theta, \mu, \sigma^2; x_1, \ldots, x_n) &= log(f_X(x_1, \cdots, x_n)) \\ &= log(\frac{1}{(2\pi)^{n/2}|V(\phi,\theta)|^{1/2}} \times e^{-\frac{1}{2\sigma^2}(x-\mu_{1\times n})V(\phi,\theta)^{-1}(x-\mu_{1\times n})^T}) \\ &\equiv -\frac{1}{2}\{nlog\sigma^2 + log|V(\phi,\theta)| + \frac{(\mathbf{x}-\mu_{1\times n})V(\phi,\theta)^{-1}(\mathbf{x}-\mu_{1\times n})^T}{\sigma^2}\} \end{aligned} \tag{4.8}$$

Therefore, ϕ and θ are coefficients need to be estimated, together with μ and σ^2, by using the following optimization function,

$$\{\hat{\phi}, \hat{\theta}, \hat{\mu}, \hat{\sigma}\} = \underset{\phi,\theta,\mu,\sigma^2}{argmax}\{\ell(\phi, \theta, \mu, \sigma^2; x_1, \ldots, x_n)\} \tag{4.9}$$

By using the R package for Statistical Computing [6], the estimated result for the ARIMA(2,1,6) model is as follows,

$$\begin{aligned} x_t = \;& \phi_{t-2}x_{t-2} + \phi_{t-1}x_{t-1} + \epsilon_t + \theta_{t-1}\epsilon_{t-1} + \theta_{t-2}\epsilon_{t-2} + \theta_{t-3}\epsilon_{t-3} + \theta_{t-4}\epsilon_{t-4} + \theta_{t-5}\epsilon_{t-5} + \theta_{t-6}\epsilon_{t-6} \\ = \;& 1.5838x_{t-2} - 0.7359x_{t-1} + \epsilon_t - 1.2966\epsilon_{t-1} + 0.5590\epsilon_{t-2} \\ & -0.0446\epsilon_{t-3} - 0.0078\epsilon_{t-4} + 0.1087\epsilon_{t-5} - 0.0115\epsilon_{t-6} \end{aligned}$$

where the standard deviations of those parameters are 0.0860, 0.0641, 0.0873, 0.0525, 0.0307, 0.0295, 0.0264, 0.0254, respectively; σ^2 is estimated as 0.3029.

4) **Diagnosis Checking** – After specifying model and estimating its parameters, diagnose checking is to analyze the *goodness* of the model, i.e., checking whether it fits well the real data set. Residual analysis is a typical method for model diagnostics in statistics, i.e., analyzing the $\{residual = actual - predicted\}$. We compute and plot the diagnostic results in Figure 4.22: sub-figure (a) is the standard residuals, we can see it looks like a typical normal distribution; (b) is the ACF (Autocorrelation of residuals) plot with clearly cut off at lag 1; (c) is the Q-Q (quantile-quantile) plot which is an effective tool for assessing normality; and finally

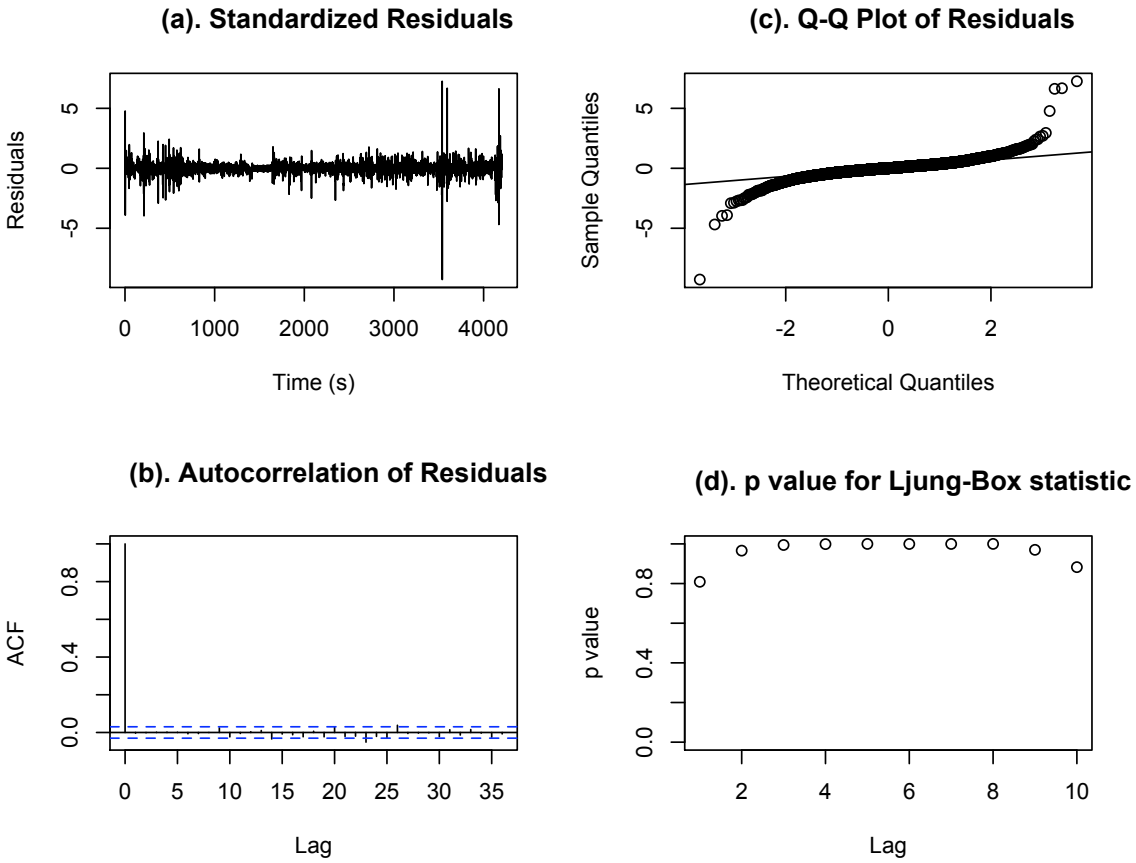


Figure 4.22: Diagnose checking plots of TrajARIMA

(d) shows the p-values that are very close to 1. Those plots validate the good fitness of the model; and the only exception is the Q-Q plot of the residuals which is not so perfect.

5) **Forecasting** – One of the primary objectives of building a ARIMA model for time series is to forecast the values at future time. The following Figure 4.23 shows the forecasting results of the learned ARIMA(2,1,6) model. From the data prediction perspective, this result is not very convincing. There are following possible reasons to explain: (1) up to now, for this data set, it is still using one dimensional ARIMA model for trajectory data, which only focuses on temporal correlations, and there is no consideration about spatial correlations, that is why we need the spatial time series model for trajectory data; (2) building a ARIMA model for a whole trajectory is not so rational, as the focus of trajectory structure is on cutting trajectories into several episodes (stops and moves), therefore we can apply the TrajARIMA model for the individual move parts, which can achieve better forecasting accuracy.

In [RTO+10], Rocha et al. have proposed a direction-based spatio-temporal clustering methods for stop discovery. It is quite similar to our TrajDBSCAN algorithm, but focuses on using the parameter of *direction variation* to determine whether a cluster can be expanded or not. As

[6] `http://www.r-project.org/`

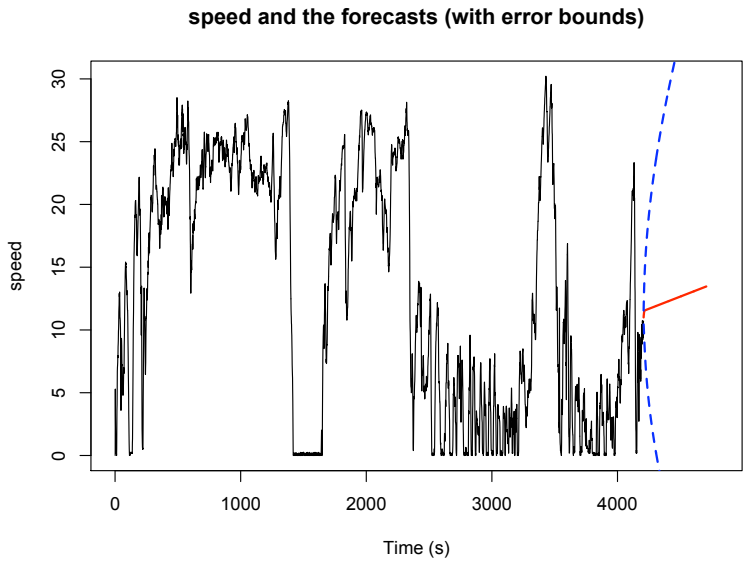


Figure 4.23: Trajectory speed forecasting

only using the direction information, this method is only meaningful to certain applications, like the shipping trajectory as shown in the paper.

Additionally, Buchin et al. present a theoretic framework that compute optimal segmentation using different criteria (e.g., speed, direction, location disk) and some possible combinations [BDKS10]. However, there is no experimental study to validate such segmentation framework.

All of these trajectory structure algorithms are offline solutions. Relevant segmentation algorithms need to tune many thresholds placed on movement features (like *acceleration, direction alteration, stop duration etc.*) so as to find their most suitable parametric values, sometimes in a per object fashion. Therefore, the trajectory structure result is sensitive to the parameter values. In our online computation in Chapter 6, we will focus on designing robust segmentation algorithms that do not rely on many predefined thresholds on certain movement features; the online computation applies pattern alteration under movement similarity estimations, which only allows a similarity threshold on movement patterns.

4.6 Experiment

4.6.1 Trajectory Datasets

We have validated our model and computing platform against different kinds of real-life GPS datasets. During the discussion of modeling requirement in Section 3.2, we have summarized three types of mobility data scenarios, i.e., *vehicle trajectories*, *people trajectories*, and *biological trajectories*. Table 4.2 provides a short list of these datasets from various scenarios that we have studied during our experimental analysis in trajectory computing. The first four datasets are about vehicle trajectories (including car, bus, truck, and taxi); the latter two are for people trajectory and biological trajectory, respectively.

Table 4.2: Trajectory datasets - real-life GPS feeds

	Dataset	*# objects*	*# GPS records*	*Traking time*	*Sampling frequency*
(1)	car (Milan)	17,241	2,075,213	1 week	avg. 40 seconds
(2)	bus (Athens)	2	66,095	108 days	30 seconds
(3)	truck (Athens)	50	112,203	33 days	30 seconds
(4)	taxi (Lausanne)	2	3,347,036	5 months	1 second
(5)	people track	185	7,306,044	1.5 years	10 seconds
(6)	bird migration	19	1,890	8 years	several days to a month

In particular for people trajectories, we additionally focus on studying a concrete subset, i.e., the mobility data of six users that we know their underlying "movement behaviors" as the ground truth data, which is significantly useful for validation. The details of the mobility data of these six users are shown in Table 4.3.

Table 4.3: People trajectory data from mobile phones

All dataset	***Details of a sub-dataset of 6 users***				
summary	*user-id*	*from-date*	*to-date*	*#days-with-gps*	*#GPS*
185 smartphone users	1	2009-02-17	2010-04-27	191	50,274
23,188 daily trajectories	2	2009-02-25	2010-05-16	330	200,418
7,306,044 GPS records	3	2009-09-14	2010-05-16	166	62,272
from date: 2009-02-01	4	2009-11-19	2010-05-16	161	66,304
to date: 2010-08-16	5	2009-12-18	2010-05-16	140	69,467
	6	2010-01-25	2010-05-16	89	45,137

4.6.2 Trajectories at Different Levels – Data Abstraction

The main outcome of trajectory computing is that we construct different levels of trajectory data abstraction, i.e., from raw mobility data, to spatio-temporal trajectories ($\mathcal{T}_{spa}$), to structured trajectories ($\mathcal{T}_{str}$), and finally to the semantic trajectories ($\mathcal{T}_{sem}$). We can achieve a significant decrease in the data size as trajectories are computed and abstracted from the low-level feeds to the higher level models. Therefore, the experiment can show the achievement of data abstraction from trajectory computing.

In this thesis, we firstly design a notion of *abstraction rate* to measure the compression rate at different data abstraction levels resulting from trajectory computing,

$$abstractionRate = log_2(\frac{\#GPS}{\#dataComputed}) \tag{4.10}$$

where $\#GPS$ is the number of the initial GPS records, and $\#dataComputed$ is the number of computed model instances, i.e., the number of trajectories, episodes (stops and moves), and

semantic episodes (e.g., with semantic annotation by geographic landuse cells)[7]. For example in taxi dataset, the initial 3,347,036 GPS records are abstracted to 1,145 trajectories with 1,874 stops and 2,925 moves in structured trajectories, and finally only 816 semantic stops in the semantic trajectories. Trajectory at the higher layer encapsulates multiple concepts from trajectories at the underlying lower layer. Figure 4.24 shows the detailed abstraction results for our four vehicle trajectory datasets. Another interesting (and reasonable) observation is that the abstraction rate is proportional to the GPS sampling frequency. From left to right in Figure 4.24, the GPS recording frequencies of these four datasets are one record per 40 seconds (on average), 30 seconds, 30 seconds, and one second, respectively. We observe that the higher recording frequency (like taxi data), the higher abstraction rate as data compression.

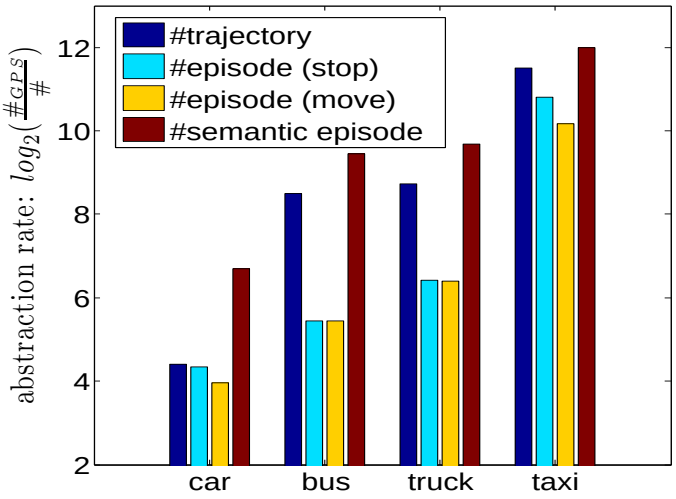


Figure 4.24: Vehicle trajectory computing - data abstraction by $log_2(\frac{\#GPS}{\#dataComputed})$

Regarding the people trajectory scenario, trajectory computing abstracts the 7.3M GPS records to 46,958 moves and 52,497 stops in 23,188 daily trajectories. To better understand this type of knowledge inference, Figure 4.25 shows the loglog plot of the length (i.e., the number of GPS records) of extracted trajectories, stops and moves. It shows that most of *moves/trajectories* have similar patterns, with a large number of GPS records (say more than 10^3), whilst the number of GPS records in *stops* largely stay between 10^2 and 500, decreases from 10^2 to 10^1, and has few unusual cases in $[500, 10^3]$. In addition, Figure 4.26 shows the details of *stops* and *moves* for the selected 1,077 daily trajectories of the six concrete users in Table 4.3. Note that the number of GPS records for each user in Figure 4.26 is divided by 100, for better representation purposes, and to bring out the storage compression achievement.

4.6.3 Trajectories at Different Levels – Visualization

In order to diagrammatically present trajectory computing results, we have implemented a hybrid trajectory visualization tool using Java 2D API. Figure 4.27 provides a snapshot of

[7]The detailed techniques of computing such "semantic episodes" with geographic sources like landuse (for achieving the "semantic trajectories" $\mathcal{T}_{sem}$) will be discussed in the later Chapter 5 about trajectory semantic annotation. In this section, we refer to *semantic episodes* or *semantic trajectories* only for showing the results and visualization of the different levels of mobility data abstraction.

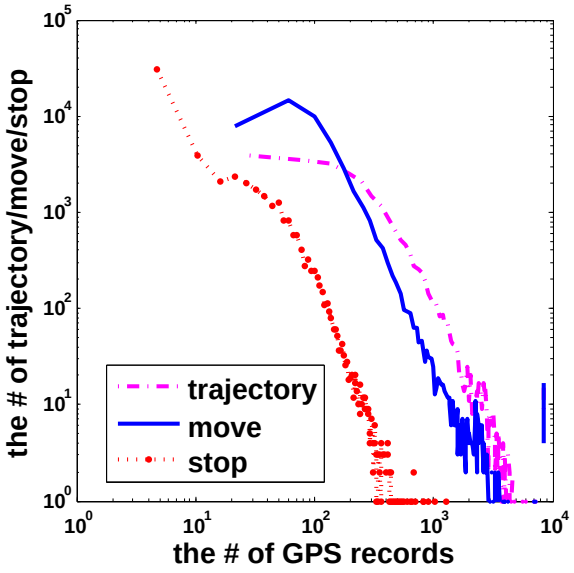


Figure 4.25: People trajectories computing (results distribution)

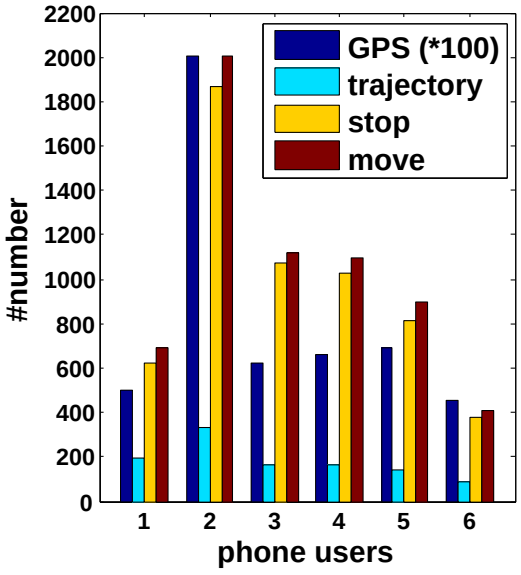


Figure 4.26: People trajectories computing (6 sample users)

the tool that presents four sub-figures corresponding to original *GPS feeds*, *spatio-temporal trajectory*, *structured trajectory*, and *semantic trajectory* computed from the truck dataset. The order of sub-figures (from left to right) follows the progressive procedure of computing different levels of trajectories from the raw feed.

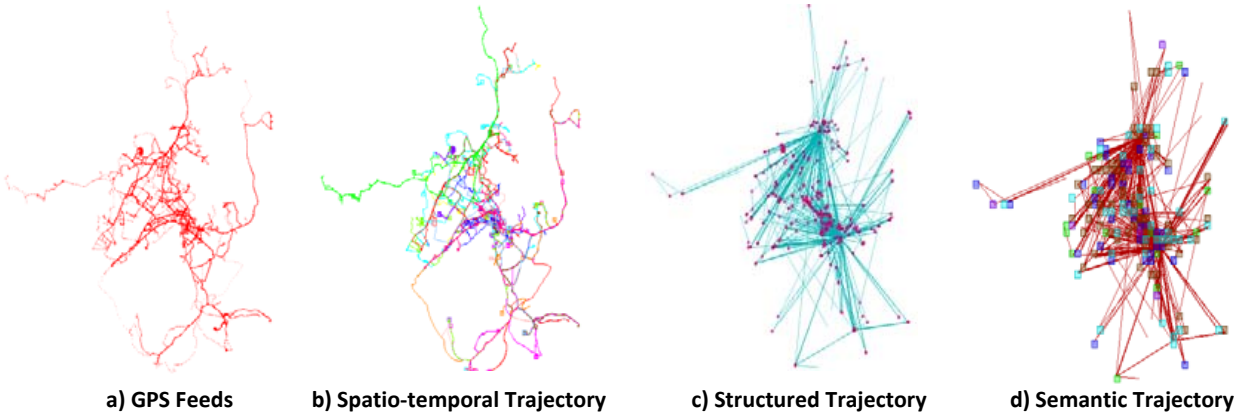


Figure 4.27: Visualization - from GPS feed to semantic trajectories

- Sub-figure (a) visualizes the spatial locations of 112,203 raw GPS records, in terms of their 2D geometric coordinates (x, y), without any further meanings (as the output of the *Data Preprocessing* Layer).
- Sub-figure (b) shows 310 spatio-temporal trajectories obtained from the (x, y, t) cleaned sequence (as the output of the *Trajectory Identification* Layer). For better distinguishing, the neighboring trajectories are shown in different colors.
- Sub-figure (c) displays the trajectory episodes (i.e., stops and moves) and visualizes structured trajectories (output of *Trajectory Structure* Layer). There are 1826 stops (visualized

as *points*) and 1849 moves (as *lines between points*).

- Sub-figure (d) shows the final semantic trajectories from the *Semantic Annotation* in Chapter 5. It displays the semantically enriched stop episodes, where 1826 stops are mapped to 160 landuse grids (visualized as *squares*) in 5 different types. Landuse grids from different types are drawn with distinct colors.

Similarly, Figure 4.28 shows the trajectory episodes that are computed in the Milano car trajectories. In such trajectory visualization, there are four different kind of episodes (i.e., *stop*, *move*, *begin*, *end* . The notions of *begin*, *end* are defined in [SPD+08], and correspond to the *starting* and the *ending* points of an individual trajectory.

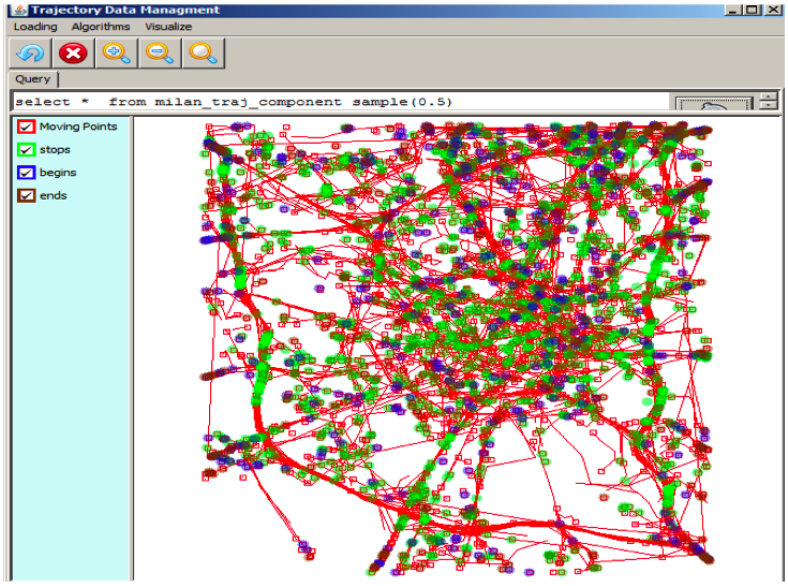


Figure 4.28: Episodes in Milano car trajectories

4.6.4 Sensitivity Analysis of Computing Parameters

As mentioned before, the coefficients for speed thresholds play a role in determining the number of stop and move episodes; and as pointed out before, the speed threshold dynamically depends on several factors (vehicle type, road type etc). Results presented in Figure 4.24 have used the same coefficient of speed thresholds ($\delta_1 = \delta_2 = \delta = 0.3$) and the same minimal stop duration ($\tau = 15$ *mins*) to provide a comparative picture of the abstraction. However, these parameters influence the number of trajectory episodes and need to be calibrated accordingly.

We analyzed the sensitivity of δ and τ in identifying stop episodes. Figure 4.29 shows the number of stops we get with different δ and τ for the Athens truck data. With higher τ (from five minutes to one hour), the number of stops decreases from 2601 to about 633 when given $\delta = 0.15$; whilst with higher δ (from 0.015 to 0.9), the stop number goes up then saturates, because stops computed with higher coefficient δ (i.e., higher Δ_{speed}) usually have longer duration. Therefore the number of stops decrease as some stops join together. Nevertheless, we observe that the total percentage of time duration for stops always increases when the minimal stop time τ becomes

smaller or the speed threshold δ increases (see Figure 4.30). We are investigating better solutions when dynamically determine stop thresholds by calibrating these parameters.

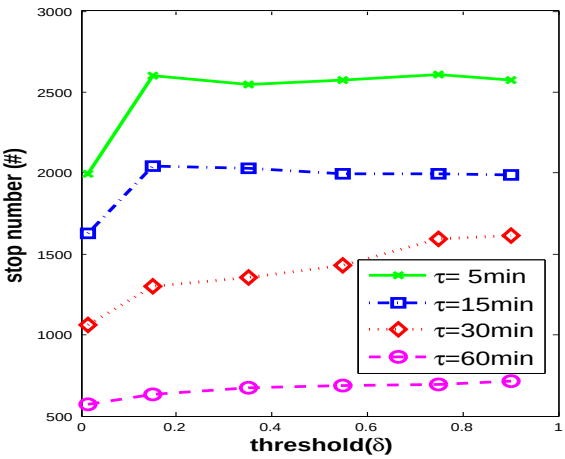


Figure 4.29: Δ_{speed} w.r.t. total stop number

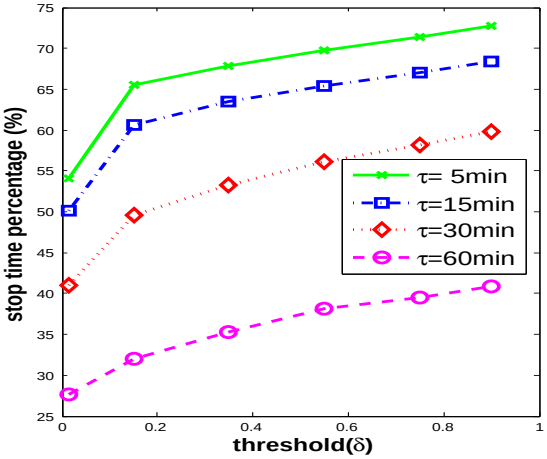


Figure 4.30: Δ_{speed} w.r.t. total stop time

Map-matching is applied for cleaning the network-constrained trajectory data. To measure the efficiency of map matching, we perform a sensitivity analysis of the algorithm using a benchmark dataset – a GPS trace of 2-hour drive of a private car in Seattle with ground truth data (i.e., the real path) provided by Krumm[8] for testing map matching in the network-constrained trajectory data cleaning. We tune the global view radius (R) and the kernel width (σ) for the input data source. Figure 4.31 shows such trajectory with labeled road segments. Figure 4.32 shows the effect of different σ and R on the final matching accuracy. We observe that small values of R (=2) and σ (=0.5R) produce very high matching accuracy, similar to the recent results on this dataset [NK09], confirming the efficiency of the algorithm in fast computation.

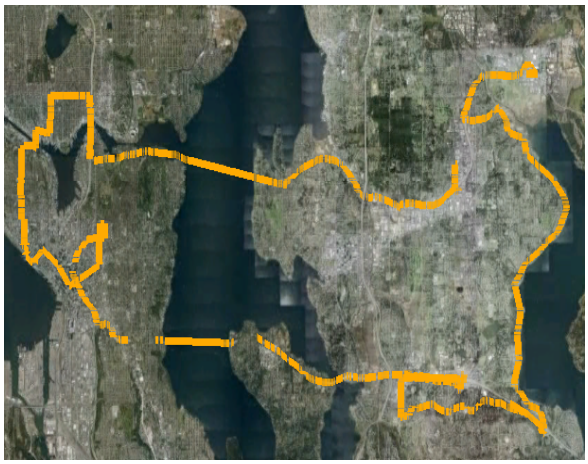


Figure 4.31: Example of map matching data with ground truth

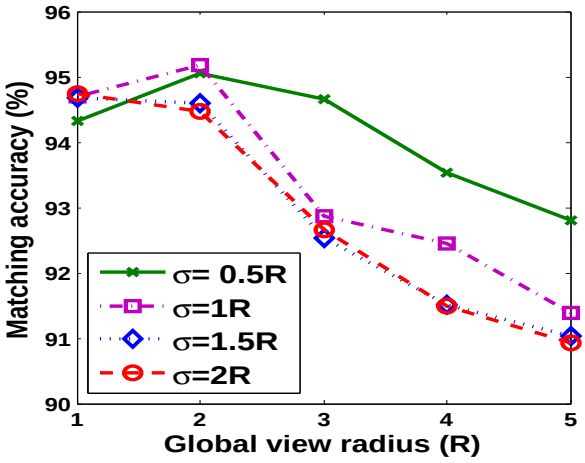


Figure 4.32: Sensitivity of map matching accuracy w.r.t. R/σ

[8] http://research.microsoft.com/en-us/um/people/jckrumm/MapMatchingData/data.htm

4.7 Summary

This chapter presented the second major contribution of this thesis, i.e., the *offline trajectory computing*. There are three main issues in offline trajectory computing including (1) *data preprocessing* with tasks of data cleaning, data compression, map matching etc., (2) *trajectory identification* to find the trajectory's starting point and ending point, and (3) *trajectory structure* to segment a trajectory into several episodes. The initial result of the offline trajectory computing platform is published in [YPSC10]. Regarding the time-series model for network-constrained trajectory, the detailed result is published in [Yan10].

The *trajectory data preprocessing* layer is the inevitable preliminary steps in trajectory computing. In such steps, we build dedicated algorithms for data format transformation, cleaning via smoothing and filtering for freely-movement trajectory datasets, cleaning via map matching for network-constrained trajectory scenarios, data compressing with the SED or STTrace metrics etc. With such layer, the raw trajectory data from GPS alike positioning sensors can be cleaned for better fitting further trajectory computation tasks.

The *trajectory identification* layer has built relevant trajectory identifying policies that can divide the long sequence of cleaned $\langle x, y, t\rangle$ sequence into several non-overlapping spatio-temporal trajectories $\mathcal{T}_{spa}$. The main policies are *raw GPS gaps*, *prior knowledge* like time interval or space extent, *correlation-based information* like applying time series segmentation techniques for automatic trajectory identification without a *priori* knowledge .

The *trajectory structure* layer is the core issue in computing trajectories, which segments each single spatio-temporal trajectory $\mathcal{T}_{spa}$ into a sequence of trajectory episodes and produces a structured trajectory $\mathcal{T}_{str}$. Dedicated algorithms include velocity-based, density-based, time-series based segmentation and stop discovery methods. Most of these types of trajectory segmentation algorithms have a common drawback: *they are sensitive to the computing parameters like speed threshold, minimal stop time, min spatial distance in density etc.* In Chapter 6, we will further discuss methods that can perform online and are less-sensitive to parameters.

Chapter 5

Trajectory Semantic Annotation

Annotation is the act or process of furnishing critical commentary or explanatory notes.

American Heritage Dictionary

5.1 Introduction

In this chapter, we address the third main contribution of this thesis, i.e., *trajectory semantic annotation*. Resulting from the trajectory offline computing in Chapter 4, we have already achieved the structured trajectories $\mathcal{T}_{str}$ (as a sequence of trajectory episodes) from the raw GPS alike tracking data. To further analyze the semantics of these trajectory episodes in $\mathcal{T}_{str}$ and better understand the mobility behaviors, this chapter focuses on designing a comprehensive annotation framework to enrich trajectory semantics with additional third party semantic data sources including "geographic" or "application domain" knowledge. According to the underlying spatial extent, we divide these extra semantic data sources into three sub-categories, i.e., "semantic regions", "semantic lines", and "semantic points". This chapter explains the details of our annotation approach using heterogeneous third party semantic sources, in terms of three dedicated annotation algorithms. To achieve an automatic and heterogeneous semantic annotation, our design principle in semantic annotation is that *the algorithms should exhibit good performance over a wide range of trajectories with varying data quality.*

This chapter is organized as follows: Section 5.2 presents the annotation challenges and our heterogeneous annotation approach; Section 5.3 addresses annotation algorithm by using semantic regions (i.e., Regions of Interest); Section 5.4 addresses annotation algorithm with semantic lines (i.e., Lines of Interest); Section 5.5 addresses annotation algorithm with semantic points (i.e., Points of Interest); Section 5.6 shows some experimental studies; and finally Section 5.7 summarizes this chapter.

5.2 Semantic Annotation Approach

Applications do benefit from semantic enrichment of trajectories. For example, when analyzing people trajectories, rather than using the GPS data (or the episodes without meaningful annotations), we can easily imagine that the application prefers to view a trajectory as the following semantically encoded sequence of triples:

$$(home,\ {\sim}8am,\ \text{-}) \rightarrow (road,\ 8am{\sim}9am,\ on\text{-}bus) \rightarrow (office,\ 9am{\sim}6pm,\ work) \rightarrow$$
$$(road,\ 6pm{\sim}6{:}30pm,\ on\text{-}metro) \rightarrow (market,\ 6{:}30pm{\sim}7{:}30pm,\ shopping) \rightarrow$$
$$(road,\ 7{:}30pm{\sim}8pm,\ on\text{-}foot) \rightarrow (home,\ 8pm{\sim},\ \text{-}),$$

as already shown in Figure 1.2 in Chapter 1, as well as in the top of Figure 5.1.

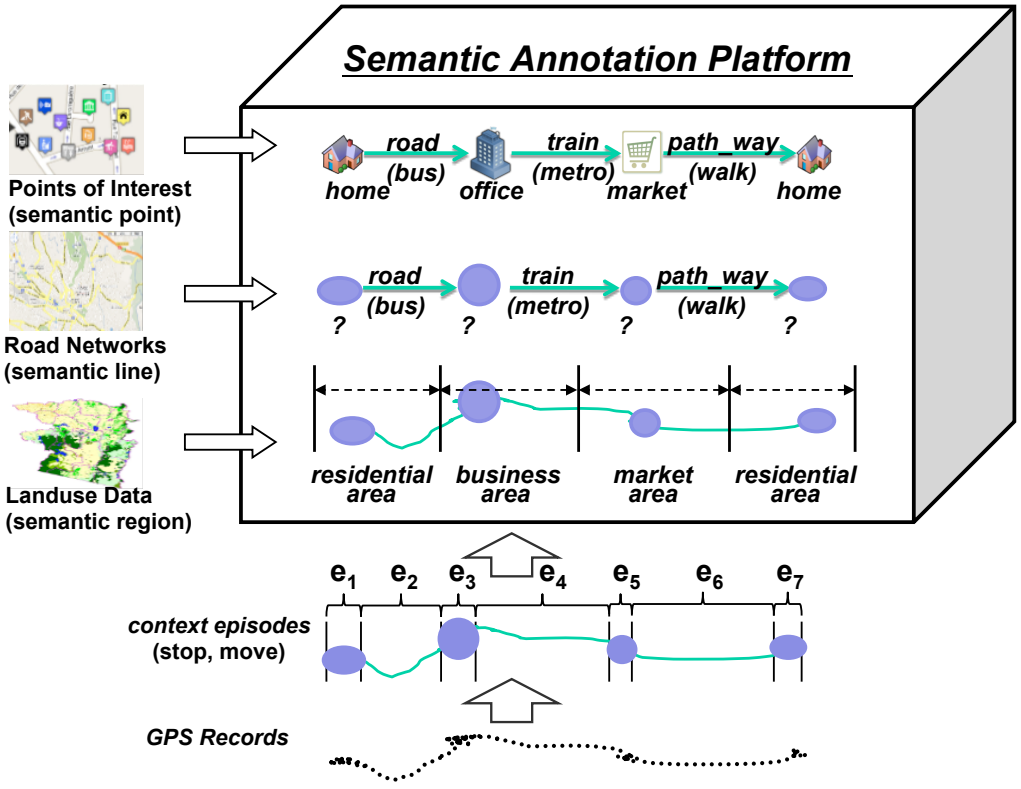


Figure 5.1: Logical view of trajectory annotation

Notice that the first and last triples denote the start (*Begin*) and ending (*End*) spatio-temporal positions delimiting the trajectory, respectively. In all triples, (1) the first element, the spatial ($\langle x, y \rangle$) location, is encoded at the semantic level with labels such as "home", "office", "road", "market", expressing the application's interpretation of the location; (2) the second element denotes the *time period* when the two other elements remain constant (i.e., at the same location, sharing the same annotation); and (3) the third element in the triples conveys additional semantic annotation, in this case related to the activity (e.g., work, shopping for the stop episodes) or to the means of transportation (e.g., on bus, in metro, by foot, etc). Clearly, abstracting trajectory data to such a semantic representation enables a better understanding of the semantic movement behavior. Furthermore, analytics on such semantic trajectories enables contextual and relevant information discovery, which can significantly empower a wide range of

trajectory applications, e.g., finding similarity of movement, semantic pattern mining, mobility analysis and statistical analysis etc.

5.2.1 Annotation Challenges

Designing a generic and efficient annotation framework is non-trivial as many different issues have to be addressed.

(1) The framework should be *application-independent* while being able to support the specific requirements of any potential applications (e.g., traffic monitoring, semantic location analysis). For example, different levels of granularity are required to analyze movement, e.g., cars move between cities or within a city, customers visit different shops in a commercial center, and people can travel across countries. Car movement is constrained by the underlying road network, while people walk follows unplanned paths through places such as parks and buildings. Therefore, no application-specific data should be hard-coded into the framework. Instead, the framework should have the capability of acquiring from third party information whatever geographic or application-specific data sources are needed, and input such data source into the annotation algorithms.

(2) While being generic, the annotation algorithms should exhibit a good performance whatever the characteristics and data qualities of trajectories are. Sampling rates and GPS signal availability influence the quality of raw trajectory data. For example, while vehicles mostly enjoy good GPS coverage, GPS signal may be lost during people's indoor movement. Trajectories might lack enough data to precisely locate which building the person entered. As a result, mapping trajectories to location artifacts in complex nature environments such as dense urban areas is a challenge. The algorithms should be able to handle variations in data quality while annotating trajectory, i.e., supporting *heterogeneous trajectories*.

(3) Providing a *holistic annotation framework* usually calls for integration of several independent information sources. As a *priori* assumption, the amount of candidate sources for annotation data is high and spatially dense. The framework needs to be able to select the most relevant sources and the most relevant kinds of annotation data for each trajectory segment. For example, it does not make sense to annotate a moving car with the list of restaurants or other location artifacts it quickly passes by, unless it stops around one for certain activity (e.g., having dinner in a restaurant). Overwhelming coverage of space is frequently a problem. For example, a major difficulty in choosing the closest points of interest to annotate a given trajectory is not in distance computation but in relevance evaluation. The location where a person stops for shopping in a city center may be associated to many shops in the vicinity. Therefore, some applications need to infer the exact shop the person stopped for.

(4) For *computational efficiency*, annotating each GPS point may result in information overload. The trajectory semantic model must offer generic means of semantically aggregating correlated records and provide their condensed representation at the semantic level. Therefore, annotation on episodes in structured trajectory ($\mathcal{T}_{str}$) would be more efficient than directly annotating on the data points in spatio-temporal trajectory ($\mathcal{T}_{spa}$).

To summarize, the challenges of a comprehensive trajectory annotation approach we need to address can be stated as the following aspects:

(a) To provide a framework that covers the requirements of a wide range of applications. The framework includes both the specification of a generic conceptual model, as well as the specification and implementation of annotation algorithms that exhibit a good performance over a wide range of requirements and data qualities.

(b) To enable determining which kinds of semantic annotation data should be extracted from available sources and how to appropriately filter it to match the moving object at hand.

(c) To design computationally efficient annotation algorithms, since the available datasets are large and quickly growing, and annotation data is even required in real-time.

5.2.2 Annotation Framework

Before presenting the detailed annotation framework proposed in this thesis, we provide relevant definitions as well as the design principle to build a comprehensive annotation framework.

As already presented in previous chapters, the spatio-temporal trajectories and structured trajectories are computed from the raw data stream (generated by GPS-alike mobile positioning sensors) of a moving object. Such trajectory datasets are of varying size, depending on tracking time and location update frequency. There may be gaps in the recording due to several reasons, e.g., signal loss, battery outage, network disconnections, etc. In addition, moving objects or users generating trajectories can use various transportation modes (e.g., walk, metro/bus, bike) which can bring highly varying trajectory data characteristics (e.g., acceleration, velocity). We call all of these diverse trajectories "*heterogeneous trajectories*", as the main focus to deal with in our semantic annotation task.

As a first step towards a semantic representation of trajectories, we introduce *Semantic Places* ($\mathcal{P}$) as the semantic counterpart to annotate the spatio-temporal positions. $\mathcal{P}$ is defined in or inferred from third party semantic sources that contain data about the geographic objects of interest or the application knowledge at hand. The semantic annotation does not make specific difference between third party geographic information (e.g., map, points of interests) and the application domain knowledges (e.g., domain databases like the employee database composing home/office information). Therefore, we consider any general third party semantic source, divide them according to their underlying spatial extents (i.e., region, line and point), and design three dedicated annotation algorithms corresponding to the three extents.

Definition 5.1 (Semantic Places – $\mathcal{P}$)**.** A set of meaningful geographic/application objects used for annotating trajectory data. Each semantic place sp_i has an extent and other attributes describing the place. The set ($\mathcal{P}$) is partitioned into three subsets that are defined according to the geometric shape of their extent[1], i.e., $\mathcal{P} = \mathcal{P}_{region} \bigcup \mathcal{P}_{line} \bigcup \mathcal{P}_{point}$, where

- $\mathcal{P}_{region} = \{r_1, r_2, \cdots, r_{n_1}\}$ is a set of places whose extent is a region;

[1]Region (or area), line and point are standard spatial data types routinely used in GIS (Geographic Information Systems). Their formal definition can be found in e.g., [GS95]. $\mathcal{P}_{region}, \mathcal{P}_{line}, \mathcal{P}_{point}$ are also denoted as *Regions of Interest* (ROI), *Lines of Interest* (LOI), and *Points of Interest* (POI).

- $\mathcal{P}_{line} = \{l_1, l_2, \cdots, l_{n_2}\}$ is a set of places whose extent is a line;
- $\mathcal{P}_{point} = \{p_1, p_2, \cdots, p_{n_3}\}$ is a set of places whose extent is a point.

Due to the spatial extent associated with these third party geographic or application domain objects it is possible to couple a spatio-temporal position in a trajectory with the semantic places whose extent covers this position. Thus we can annotate each spatio-temporal position of a trajectory with links to the semantic place objects that the moving object has (at least we infer so) visited. This is a specific kind of annotation, the geographic reference annotations. Another kind of annotations may also need to be inferred, additional value annotations, which contain extra semantic values, e.g., "work"/"party"/"shopping" as the activities taken during stops, "bus"/"cycling"/"walking" as the transportation modes used in moves.

Definition 5.2 (Semantic Annotation on $\mathcal{T}_{spa}$ – $\mathcal{T}_{sem}^{(spa)}$)**.** A $\mathcal{T}_{sem}^{(spa)}$ is a semantically enriched spatio-temporal trajectory $\mathcal{T}_{spa}$ where spatio-temporal positions are complemented with annotations. i.e. $\mathcal{T}_{sem}^{(spa)} = \{Q'_1, Q'_2, \cdots, Q'_m\}$, where Q'_i = (x,y,t, $\mathcal{A}$) is a tuple defining a spatio-temporal point (x,y,t) and its possibly empty set of associated annotations $\mathcal{A}$.

Instead of annotating semantics on individual spatio-temporal points in $\mathcal{T}_{spa}$, another way of enhancing the knowledge on a trajectory is through structured trajectory $\mathcal{T}_{str}$, which means: *firstly compute the specific trajectory segments that are semantically meaningful for the application, namely the "episodes" via trajectory segmentation in Chapter 4; and then annotate the semantics for each episode.* Annotation on $\mathcal{T}_{str}$ not directly on $\mathcal{T}_{spa}$ is the choice in our annotation framework, and the major reasons are (1) spatio-temporal points in a single episode are more or less homogenous and share similar characteristics (e.g., similar velocity), therefore it should share the same annotation $\mathcal{A}$; (2) annotating episodes is much more efficient than directly annotating spatio-temporal points, as the data has been significantly compressed from the $\langle x, y, t\rangle$ points to the episodes. Therefore, our objective is to apply semantic annotation on $\mathcal{T}_{str}$ and achieve $\mathcal{T}_{sem}^{(str)}$ (see Definition 5.3), not annotation on spatio-temporal trajectory (see Definition 5.2). The notion of $\mathcal{T}_{sem}^{(str)}$, not $\mathcal{T}_{sem}^{(spa)}$, is more consistent with the definition of $\mathcal{T}_{sem}$ for semantic trajectory modeling in Chapter 3.

Definition 5.3 (Semantic Annotation on $\mathcal{T}_{str}$ – $\mathcal{T}_{sem}^{(str)}$)**.** A $\mathcal{T}_{sem}^{(str)}$ is the representation of a semantic trajectory as a sequence of episodes defined by a set of predicates. $\mathcal{T}_{sem}^{(str)} = \{ep_1, ep_2, \cdots, ep_m\}$, such that each episode corresponds to a subsequence of the original trajectory and is represented as a tuple $ep_i = (sp, time_{in}, time_{out}, \mathcal{A})$ where sp is a link to a semantic place ($sp \in \mathcal{P}$), $time_{in}, time_{out}$ are the time the moving object enters and exits sp, and $\mathcal{A}$ is a set of other annotations associated to the whole episode.

Figure 5.1 has already briefly sketched our annotation approach for achieving $\mathcal{T}_{sem}^{(str)}$, in terms of the logic view. This design is based on the following broad design principles:

1) **Exploit Latent Motion Context**: Motion context (e.g., whether the object is moving or stationary) is exploited in various ways. First, it plays a guiding role in choosing relevant annotations (e.g., whether to map a trajectory segment to a road or to the nearest restaurant).

Second, context persistence supports annotating trajectory episodes rather than annotating each individual GPS point. This obviously saves storage space. Such context can be extracted directly from the raw data stream, based on trajectory segmentation using homogeneous (spatio-temporal) correlations (*density, velocity, direction etc.*) present in the offline trajectory computing.

2) **Layered Approach**: The annotation framework should follow a layered approach, carefully designed to support efficient semantic annotation. The framework can progressively annotate trajectory episodes with *semantic places* $\mathcal{P}$ – first provide a coarse-gained annotation with $\mathcal{P}_{region}$; second provide a fine-gained annotation with $\mathcal{P}_{line}$ and $\mathcal{P}_{point}$, enabling e.g., dedicated stop or move annotation used for later decision making. The later sections (Section 5.3, 5.4, and 5.5, respectively) will address the details of these three annotation layers.

3) **Heterogeneity of Semantic Places**: The annotation framework should provide algorithms to map trajectory episodes to three categories of geographic or application domain objects: ROIs (regions of interest) such as *park*, *administrative region* and *landuse grids* (*residential, industrial*); LOIs (lines of interest) such as *jogging path*, *highway* and other *roads*; and POIs (points of interest) such as *bar*, *restaurant* or even a big *shopping mall*.

Figure 5.2 illustrates the system architecture for semantic trajectory computing, showing the various layers and the data flow between the layers. Our system has three main parts, i.e., *Stop/Move Computation*, *Semantic Annotation* and *Application Interface*. Semantic annotation are the main layers for trajectory semantic annotation in this chapter; whilst *Stop/Move Computation* is the offline computing presented in Chapter 4.

1) **Stop/Move Computation** – The raw GPS records are first processed by the *Trajectory Computation Layer*, which performs several data preprocessing operations: (1) remove GPS outliers and smooth the random errors; (2) identify spatio-temporal trajectories from the initial GPS data stream; (3) segment the spatio-temporal trajectory into trajectory episodes, based on several computing policies of spatio-temporal co-relations like *density*, *velocity*, *direction* etc.[2] The output trajectory episodes express the motion context (e.g., stop/move). This context can help trajectory annotation in choosing suitable geographic artifacts from third party sources and applying suitable annotation algorithms. For example, the stop episodes need to be annotated with meaningful landmarks (POIs) while the move episodes are suitable to be integrated with road networks (LOIs).

2) **Semantic Annotation** – We design three annotation layers. The *Semantic Region Annotation Layer* receives the stop/move trajectory and uses a state-of-the-art *spatial join* algorithm to pick up *regions* that the trajectory has passed through, primarily to form a coarse-grained view of the trajectory. The *move* episodes are further processed by the *Semantic Line Annotation Layer*. We have developed a new *line annotation* algorithm that is designed to consider heterogeneous trajectories and road networks. Apart from its basic operation of mapping *move* segments to road networks, this algorithm also infers transportation modes exploiting

[2]Further details of the trajectory computation operations can be found in Chapter 4, though not essential for understanding the semantic annotation algorithms.

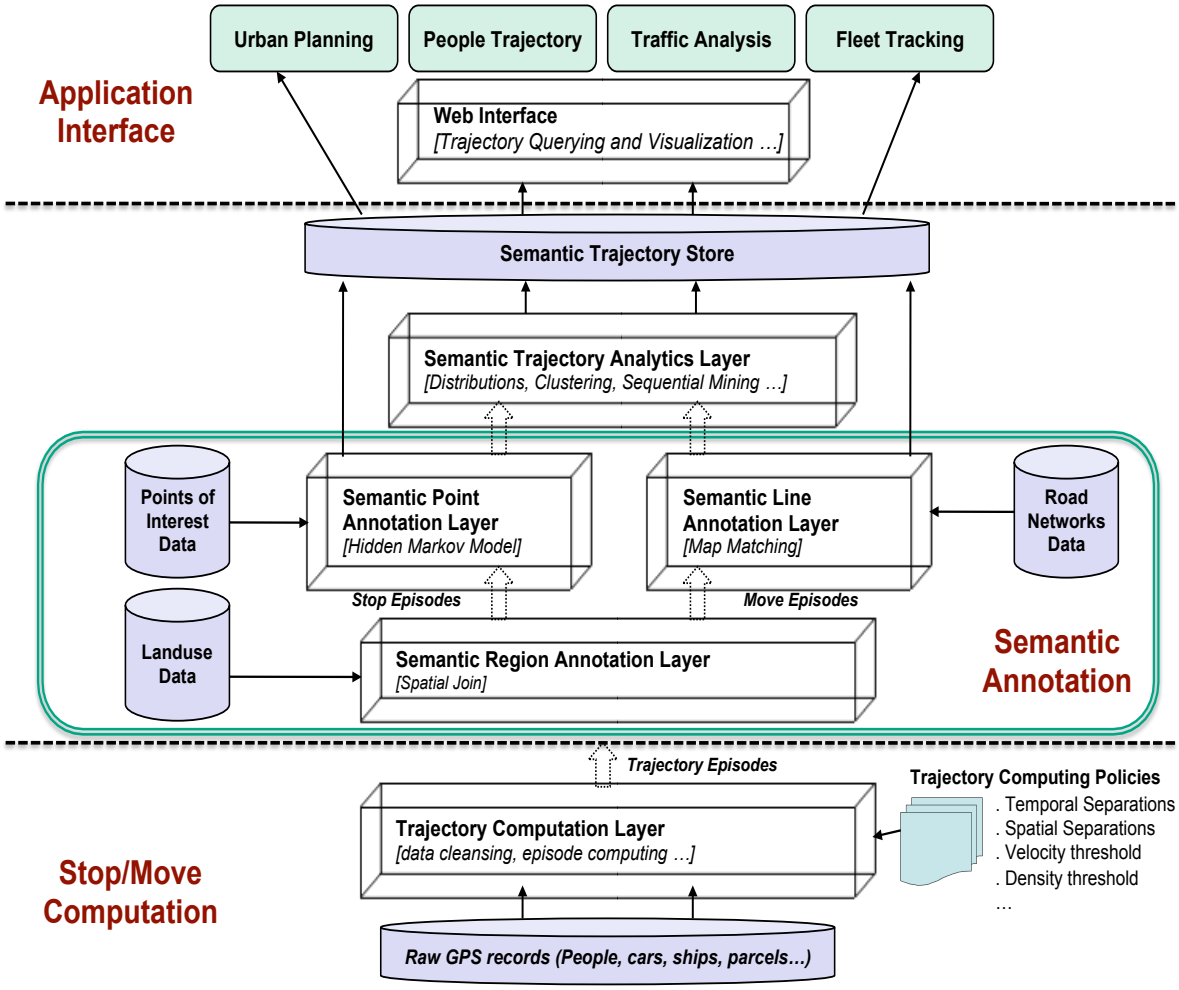


Figure 5.2: System architecture including annotation layers

geometric properties/context of the segment (e.g., velocity, acceleration) along with semantic content (e.g., which type of road). The *stop* episodes are funneled to the *Semantic Point Annotation Layer* that computes activity likelihoods and probabilistic estimates of the purpose behind that stop. This is based on a hidden Markov model algorithm that we designed, considering varying spatial densities of possible POIs for heterogeneous (sparse as well as densely populated) geographic objects. Overlapping ROIs and dense POIs have been traditionally ignored in existing studies of semantic enrichment on trajectory data [ABK+07][XDZ09]. The annotations from the three layers are combined to produce the annotated trajectory $\mathcal{T}_{sem}$. This "Semantic Annotation" part is the focus of this chapter.

3) **Additional Parts with Application Interface** – The computation and annotation result is stored in the *Semantic Trajectory Store*. A *Semantic Trajectory Analytics Layer* encapsulates methodologies that compute statistics about the trajectories (e.g., the distribution of stops/moves, frequent stops, trajectory patterns) and stores them as aggregative information in the store. This data can be accessed and used by applications. Therefore, we have built a *Web Interface* [YSC+10] that enables user to friendly queries and visually observe all kinds of computed trajectories, both at semantic and non-semantic levels.

5.3 Annotation with Semantic Regions

This layer enables annotation of trajectories with meaningful geographic regions. It does so by computing topological correlations of trajectories with third party data sources containing semantic places of spatial regions ($\mathcal{P}_{region}$). There are two types of $\mathcal{P}_{region}$: one is *free-style regions* which have non-restricted shapes, e.g., rectangle, circle, or any other irregular shapes in the real-life geographic areas (e.g., EPFL campus); the other $\mathcal{P}_{region}$ type is *well divided grid areas*, e.g., the landuse data (e.g., composed of $100m \times 100m$ cells) which has been significantly analyzed in the geographic data analysis literatures.

5.3.1 Annotation with Free-Style Regions

In this subsection, we design an efficient annotation algorithm for enriching trajectory semantics with free-style semantic regions $\mathcal{P}_{region}$. As mentioned, free-style regions are very typical in real-life geographic data sources, without any well-defined restrictions on the shape of the underlying regions. Taking the regions inside EPFL campus as a concrete example: we can extract these different types of region data from Openstreetmap[3] - which is a free editable map of the whole world and has been already mentioned in the trajectory modeling chapter. As shown in Figure 5.3, the regions inside EPFL campus area have different shape styles, e.g., regular building areas belonging to different schools, like *BC&IN* for computer science school, *EL* for electronic engineering school, *GC* for civil engineering school; a complicated library building with irregular shapes which is the famous facade at Lausanne, i.e., the *Rolex learning center*. We extract such region data from the Openstreetmap, store in our trajectory databases for the later annotation task; Figure 5.4 shows the extracted region data that visualized with our trajectory web interface via Web Browser with the support of the Google Earth plugin [YSC+10].

Figure 5.3: EPFL area free-style regions in Openstreetmap

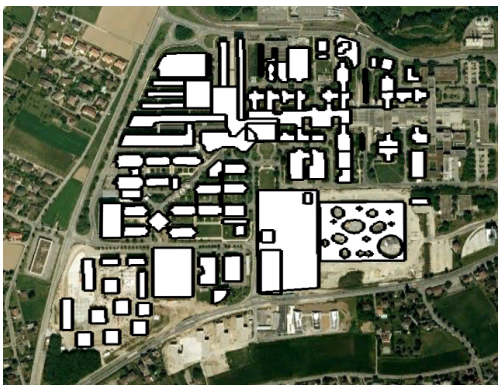


Figure 5.4: EPFL area free-style regions extracted and Google Earth visualization

For annotating trajectories with such free-style semantic regions, the topological correlation is measured using *spatial join* between a trajectory $\mathcal{T}$ and semantic regions $\mathcal{P}_{region}$ (i.e., $\mathcal{T} \bowtie_\theta \mathcal{P}_{region}$). Several forms of spatial predicates are used to compute θ, depending on the type of data. These can be a combination of *directional*, *distance*, and *topological* spatial relations (e.g., *intersection*) [BKS93]. Taking the *stop* episodes for instance, we found spatial subsumption

[3] `http://www.openstreetmap.org/`

(*ObjectA* is *inside ObjectB*) as the most frequently used predicate. For the spatial extent of episodes, we can use either the spatial *bounding rectangle* of the episode (for move or stop) or its *center* (for stop) to perform spatial join. After finding the appropriate regions (r_i), this layer annotates input trajectories with these regions and associated metadata.

Figure 5.5 shows a mobile user's daily trajectory on Sunday, annotated with semantic places of various kinds of free-style regions that are extracted from the Openstreetmap and an application database (e.g., EPFL's employee database). Therefore, the input trajectory (either spatio-temporal trajectory or structured trajectory) can be enriched and represented as the semantic trajectory: *his home* → *EPFL campus* (*staying 4 hours*) → *a swimming pool* (*staying 1 hour*) → *his home*. In this case, three semantic regions have spatial intersections with the four stop episodes from the GPS trace. Figure 5.5 displays such annotation with a Google map visualization; whilst Figure 5.6 displays it in terms of our multi-level trajectory visualization interface in [YSC+10]. From such visualization, we can claim that the annotated semantic trajectory is much more meaningful than the non-annotated ones, neither the spatio-temporal nor the structured trajectory.

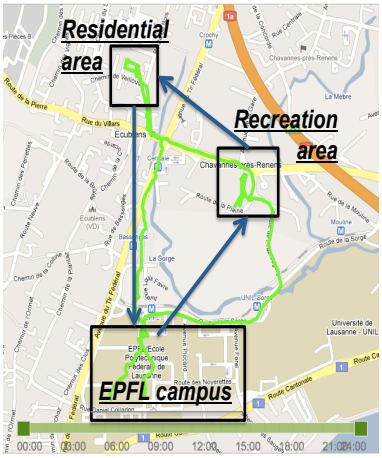


Figure 5.5: A region annotation example (visualized by Google Map)

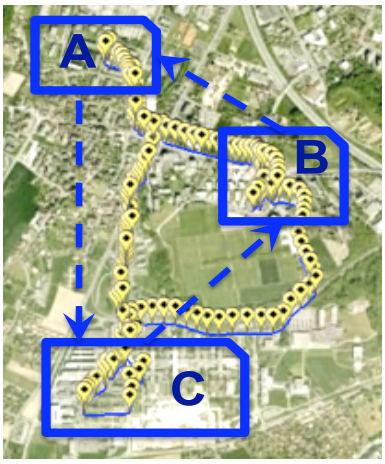


Figure 5.6: A region annotation example (visualized by trajectory Web interface)

Algorithm 5.1 shows the pseudocode of the annotation procedure with regions, which directly annotates GPS records with regions. Note that, depending on requirements, the spatial join can be used to compute only selected episodes. We apply R*-tree index on semantic regions $\mathcal{P}_{region}$ [BKSS90] to improve efficiency of the algorithm. The complexity of the annotation algorithm with region is $O(n * log(m))$, where n is the number of GPS records (or stop episodes) whilst m is the size of $\mathcal{P}_{region}$. For well-divided grid data in the next subsection, the complexity of such region annotation can be even less, namely $O(n)$. This is because for the well-divided grid data, we can directly locate the grids according to its *id* (or the *geometry* values) during the spatial join operation, without the requirement of building the R*-tree indexing for $\mathcal{P}_{region}$.

5.3.2 Annotation with Well-Divided Grids

Apart from free-style regions (no restrictions on the shape or the size, e.g., region data from Openstreemap), we can also apply well-shaped and grid-alike region data for trajectory semantic

Algorithm 5.1: ***Trajectory annotation with ROIs***

Input: (1) a trajectory $\mathcal{T}$ with its sequence of GPS points $\{Q_1, \cdots, Q_n\}$,
(2) a set of semantic regions $\mathcal{P}_{region} = \{region_1, \cdots, region_{n_1}\}$
Output: semantic trajectory $\mathcal{T}_{region}$
begin
  $\mathcal{T}_{region} \leftarrow \emptyset$; //initialize the trajectory
  /* compute *intersections* between $\mathcal{T}$ and $\mathcal{P}_{region}$; */
  do spatial join $\mathcal{T} \bowtie_{intersect} \mathcal{P}_{region}$;
  /* process each *intersection* and compute trajectory tuple */
  forall the *intersected regions* **do**
    group continuous GPS point $Q_i \in \mathcal{T}$ in the *intersection*;
    approximate entering time t_{in} and leaving time t_{out};
    create a trajectory *tuple* $\leftarrow (region_j, t_{in}, t_{out}, reg_{type})$;
    if *current* reg_{type} = *previous* reg_{type} **then**
      merge the two tuples into a single tuple;
    else
      $\mathcal{T}_{region}$.add(*tuple*); //add the previous tuple to $\mathcal{T}_{region}$;
  $\mathcal{T}_{region}$.add(*tuple*); //add the last tuple to $\mathcal{T}_{region}$;
  return trajectory $\mathcal{T}_{region}$

region annotation. For example, in our experimental studies we have significantly applied the Switzerland landuse provided by Swisstopo[4], where the landuse data is well-divided geographic grids that describe the usage details of land resource. Our Swiss landuse data contains 1,936,439 grids, and each grid is a 100m×100m square. Figure 5.7 shows an abstracted ontology of such landuse data, which has 4 top-level concepts (L_1 to L_4, i.e., *industrial*, *public buildings*, *transportation*, *wooded* areas) and 17 second-level concepts (from 1.1 to 4.17). The detailed and complete terminology can be found in the original classification in Figure 5.8.

L1 Settlement and urban areas
1.1 industrial and commercial area
1.2 building areas
1.3 transportation areas
1.4 special urban areas
1.5 recreational areas and cemeteries
L2 Agricultural areas
2.6 orchard, vineyard and horticulture areas
2.7 arable land
2.8 meadows, farm pastures
2.9 alpine agricultural areas
L3 Wooded areas
3.10 forest (except brush forest)
3.11 brush forest
3.12 woods
L4 Unproductive areas
4.13 lakes
4.14 rivers
4.15 unproductive vegetation
4.16 bare land
4.17 glaciers, perpetual snow

Figure 5.7: Landuse ontology

Figure 5.9 visualizes a small part of such Swiss landuse data in the area of Lausanne downtown by using our visualization implementation via Java 2D API; in this figure different colors

[4] http://www.swisstopo.admin.ch/

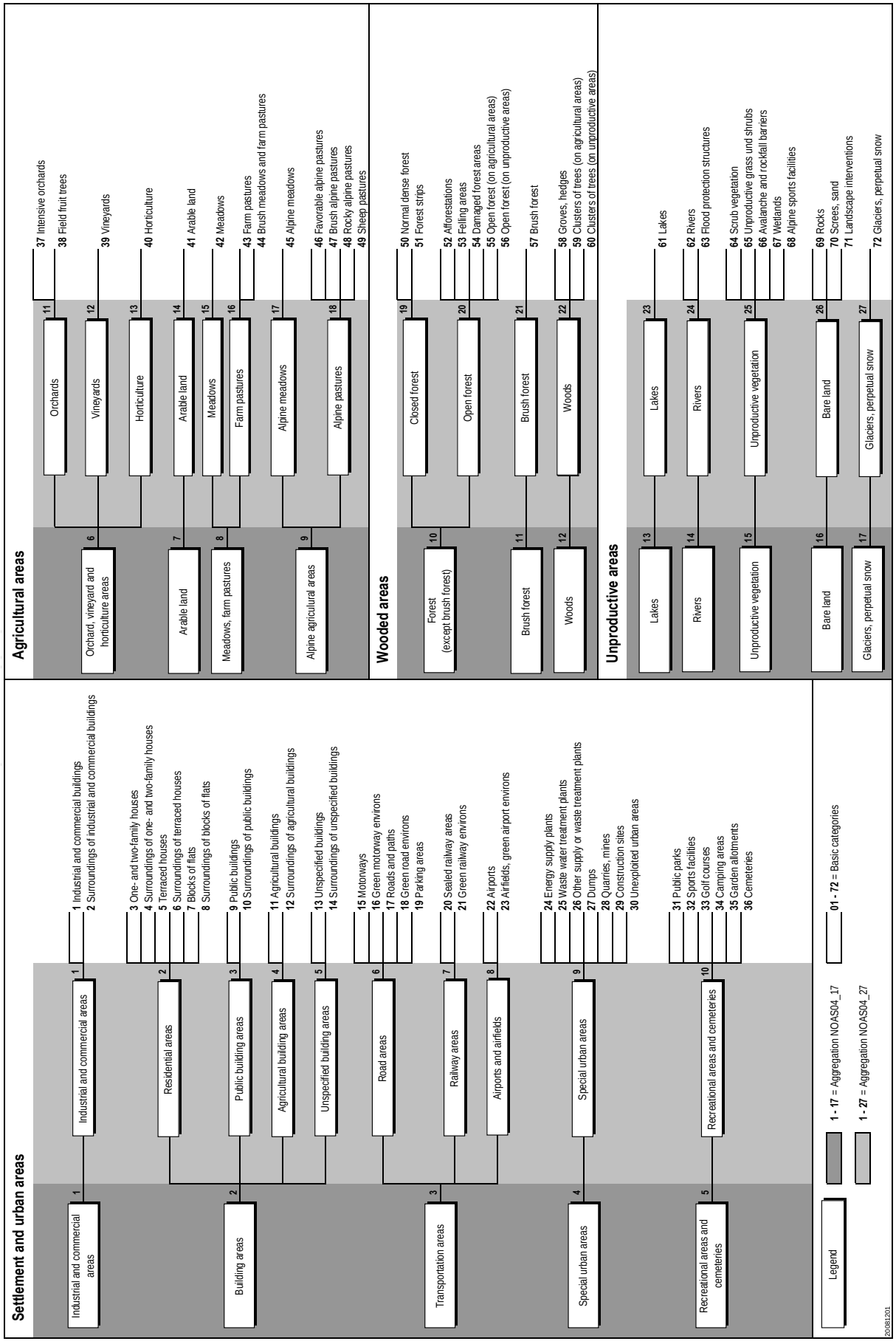


Figure 5.8: The detailed landuse ontology: categories and aggregations (from Swisstopo)

indicate different landuse categories. At the early stage of this thesis study on semantic region annotation, when real landuse data is not available for some trajectory scenarios (e.g., Milan trajectory data we used), we additionally simulate Gaussian distributed synthetic cells (i.e., the space is divided into uniform cells; each cell is randomly belonging to a certain landuse category; and the number of cells belonging to different categories follow a Gaussian distribution). Figure 5.10 shows a sample trajectory with five stop episodes. S_1 and S_5 share the same landuse cell; S_3 and S_4 belong to the same landuse category (drawn in the same color). In this case, semantic trajectory $\mathcal{T}_{sem}$ is a sequence of four landuse cells with a temporal duration $(cell_{id}, time_{in}, time_{out})$, where S_3 and S_4 share the same landuse category, and S_1 and S_5 are exactly in the same landuse cell.

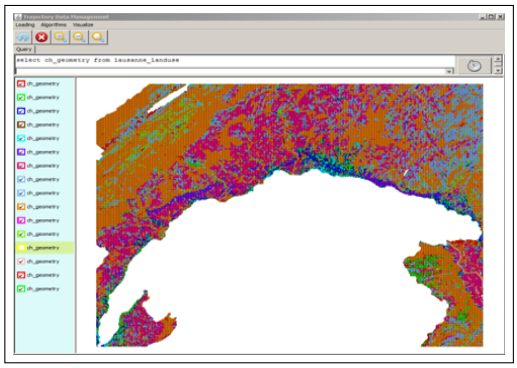


Figure 5.9: Real landuse in Lausanne downtown area

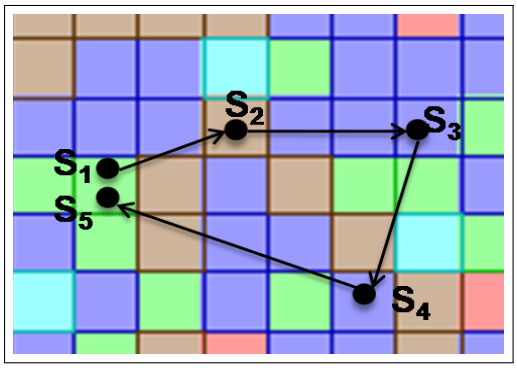


Figure 5.10: Synthetic landuse with Gaussian distribution

Regarding annotating trajectories with such well-divided landuse grids, there are two types of technical solutions:

(1) We can consider each grid as a separate semantic region, and apply the annotation procedure in Algorithm 5.1 directly on the grids. As the grid size in our landuse data is uniform (i.e., $100m \times 100m$) and the grids are linking with each other neighborhoods; therefore, we can efficiently identify whether a GPS point (x,y,t) is geographically inside a grid G_{ij} or not. In such case, without the needs of R*-tree indexing, the time complexity of Algorithm 5.1 can be reduced to *linear* on the size of trajectory data points.

(2) We can merge the neighboring grids belonging to the same landuse category into a single region, namely compute the regions from the grids; and then apply the region-based annotation algorithm in Algorithm 5.1.

For the first solution, Figure 5.11 displays a trajectory that passed an area of landuse grids. We can annotate the trajectory directly using these grids, as shown in Figure 5.12. For the second solution, we apply the *Grid2Region* algorithm in advance to merge the grids into regions, and then apply the region based semantic annotation algorithm. As shown in Figure 5.13, all of the landuse grids (17^2 in total) are merged into 8 regions, from P_1 to P_8.

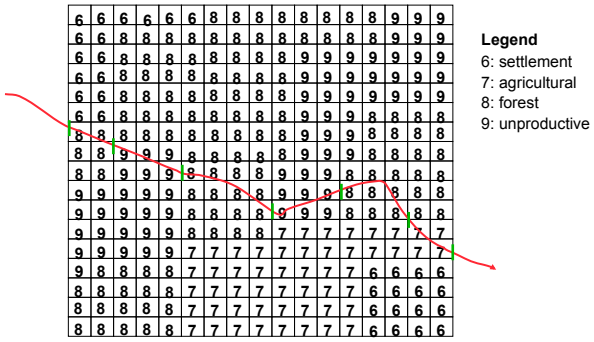


Figure 5.11: A trajectory on landuse grids

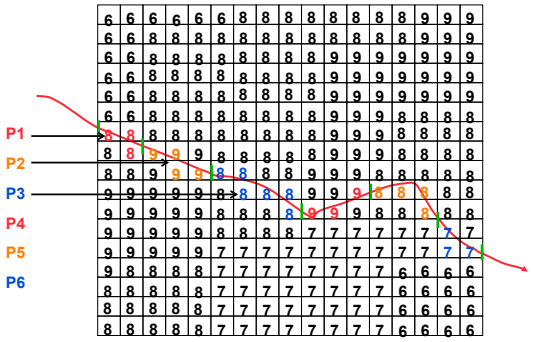


Figure 5.12: Annotation trajectory with grids

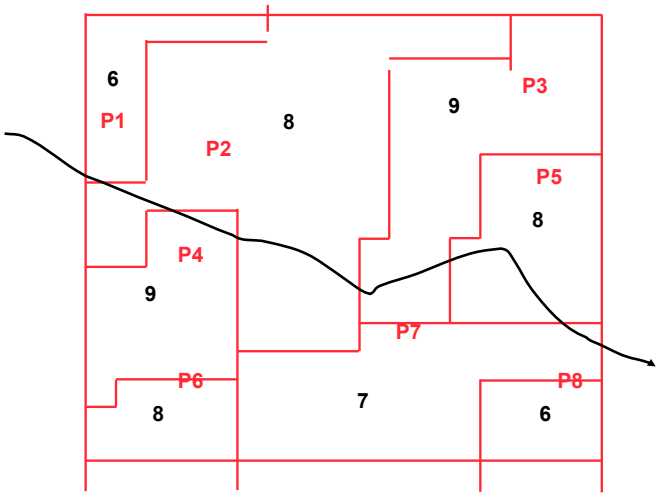


Figure 5.13: Annotation trajectory with regions that merged by grids

5.4 Annotation with Semantic Lines

This layer annotates trajectories with semantic lines (LOIs) and considers variations present in heterogeneous trajectories (e.g., vehicles run on road networks, human trajectories use a combination of transport networks and walk-ways etc). Given data sources of different form of road networks, the purpose is to identify *correct* road segments as well as infer *transportation modes* such as *walking*, *cycling*, *public transportation like metro*. Thus, the algorithms in this layer supports two main functionalities: the first one is designing a global map matching algorithm to identify the correct road segments for the move episodes of the trajectory $\mathcal{T}$, and the second one is inferring the transportation mode that the moving object used. For the first part, the map matching algorithm should support both homogeneous and heterogeneous road networks.

5.4.1 Map Matching based Annotation

In the offline trajectory computing, we have designed a global map matching algorithm for cleaning the network-constrained trajectory data in Section 4.3.3. Such map matching algorithm can be significantly redesigned for semantic annotation on trajectories by using road network data sources, i.e., the LOI data. Different from Algorithm 4.1 about data cleaning via map matching, we do not need to estimate the new location (x', y') for the input spatio-temporal data point

(x, y, t) in this section for the semantic line annotation. The objective is to annotate trajectory (precisely speaking the move episodes) with road segments, i.e., replacing the sequence of spatio-temporal points $(x_1, y_1, t_1), (x_2, y_2, t_2), \cdots, (x_n, y_n, t_n)$ with a sequence of semantic lines $(l_1, l_2, \cdots, l_m)$, and achieve certain levels of *semantic trajectory compression* like using lines to replace the data points in mobility data understanding. In [SRL09], the authors propose the similar idea of applying map matching to achieve semantic trajectory annotation and compression. Figure 5.14 sketches such annotation procedure, from the raw positioning data to the semantic trajectory that composed of meaningful lines.

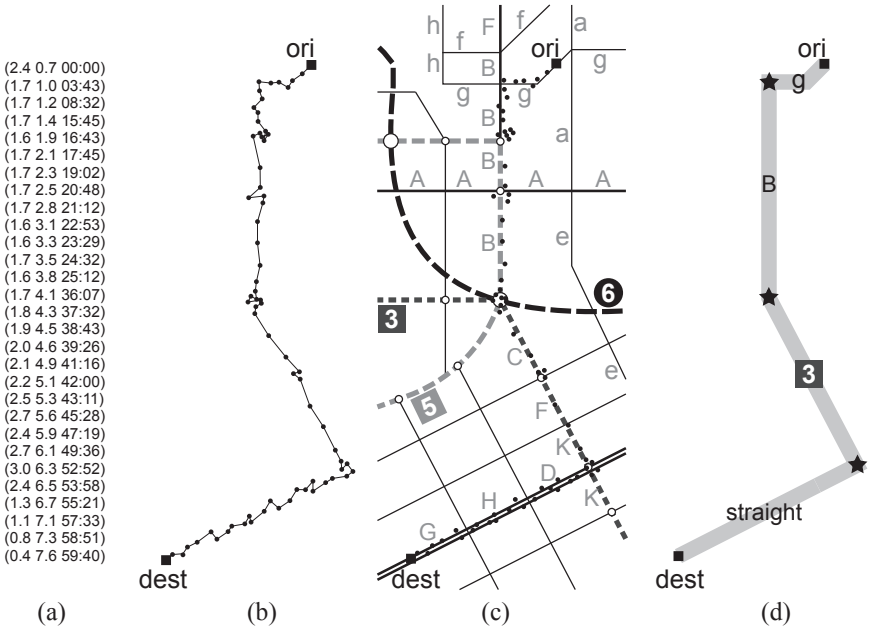


Figure 5.14: Schmid's semantic trajectory compression via map matching: (a) raw positioning data; (b) a trajectory (actually spatio-temporal trajectory) in 2D space; (c) trajectory embedded in geographic contexts (actually LOIs - Lines of interest), such as road network and metro lines (#3,#5, #6); (4) the finial semantically compressed trajectory via LOIs. [SRL09]

For trajectory semantic line annotation with road networks, the semantic lines can have two types: (1) *homogeneous lines* and (2) *heterogeneous lines*. Homogeneous lines are uniform network infrastructures like high-way, metro-line that usually are taken during vehicle movement; in such case, a single vehicle trajectory does not have irregular network change, e.g., from high-way to a metro-line. However, in people trajectories, changing between different types of road networks during movement is quite common. As shown in Figure 1.2 in the thesis introduction, the trajectory can take different types of road networks, like *bus* on the planed road, *metro* on the metro route, *walk* on the path-way. Therefore, we need additional annotation techniques for using heterogeneous road networks to better understand the movement, which comes to inferring transportation mode in the next subsection.

5.4.2 Inferring Transportation Mode

After the first step of the global map matching method (the main part of Algorithm 4.1), each move episode is annotated in terms of a list of road segments, i.e., $ep = \{r_1, r_2, \ldots, r_l\}$. We

need to further infer the annotation of transportation mode on each segment (or route), getting the pairs of $\langle r_i, mode_i \rangle$. In our study, we infer several different types of transportation modes, such as *walking*. *bicycle*, *bus* and *metro*. Such annotation is determined by the characteristics, in terms of the statistical *features* of the movement (e.g., average speed, direction of the move episode) and the type of the matched road segments (e.g., high-way, path-way, metro line, bus line etc.). Based on these road segments or network features, the state-of-the-art learning methods (like Decision tree, LibSVM, Bayesian Network, AdaBoost etc.) can be applied to build classifier on some selected training data and further test it for trajectory datasets with unknown transportation modes of episodes. Figure 5.15 briefly shows the procedure for inferring such transportation modes. This inference procedure includes *feature calculations* of different types of episode features, and *inference* including both training and testing for the classifiers of transportation modes of trajectory move episodes.

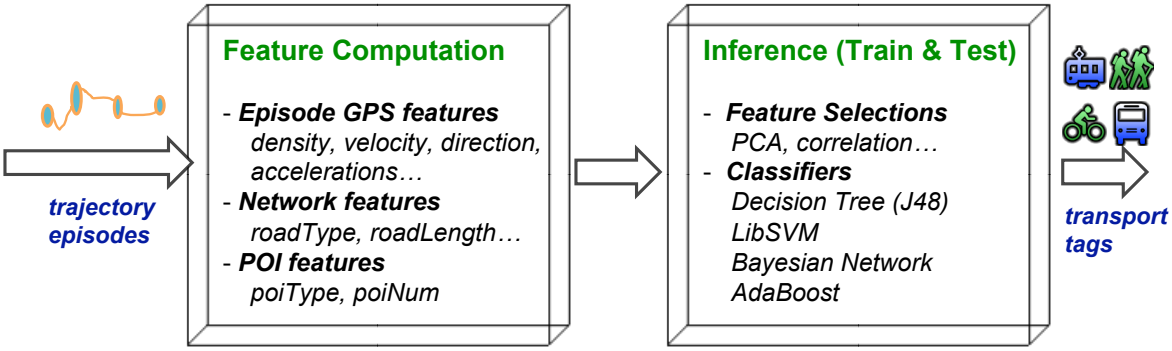


Figure 5.15: The procedure for inferring transportation mode

Table 5.1 expresses the main features that calculated and applied for training and testing a classifier to infer the possible transportation mode on the move episodes in trajectories. We can use three different types of features: (1) *GPS-based episode features*, i.e., the basic statistic information calculated by the GPS data points; (2) *network-based features*, i.e., the information about road segment, like road type; and (3) *POI-based features*, i.e., the information related to certain POIs/landmarks the episode has started or ended, passed and stayed, e.g., the number of bus-stops/shops in this move episode.

Algorithm 5.2 shows the complete procedure of *semantic line annotation*: (1) apply the global map matching algorithm that is previously designed for network-constrained trajectory data cleaning (without the needs of the final step of $\langle x', y' \rangle$ estimation); (2) compute the three types of episode features; and (3) train and test the classifiers for the inference of the transportation mode by using these episode features.

5.5 Annotation with Semantic Points

This layer annotates the *stop* episodes of a trajectory with information about suitable *Points of Interest* (POIs). Examples of POI are *restaurant*, *bar*, *shops*, *movie theater* etc. For scarcely populated landscapes, it is relatively trivial to identify the objective (in terms of POI) of a stop (e.g., petrol pump on a high-way or user's home in a residential area). However, densely

Feature Type	***Feature Name***	***Definition***
episode GPS-based features	distance	$\sum_{i=1}^{n} \lvert Q_iQ_{i+1}\rvert = \sum_{i=1}^{n} \sqrt{(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2}$
	duration	$\sum_{i=1}^{n} \lvert t_{i+1}-t_i\rvert$
	$speed_{Q_i}$	$\frac{\sqrt{(x_{i+1}-x_{i-1})^2+(y_{i+1}-y_i)^2}}{t_{i+1}-t_{i-1}}$
	$acceleration_{Q_i}$	$\frac{speed_{Q_i}-speed_{Q_{i-1}}}{t_i-t_{i-1}}$
	$direction_{Q_i}$	$\frac{direction_{Q_i}-direction_{Q_{i-1}}}{t_i-t_{i-1}}$
	$directionChange_{Q_i}$	$arctan(\frac{y_{i+1}-y_i)}{x_{i+1}-x_i})$
	speed_mean	$mean(speed_{Q_i})$
	speed_variance	$var(speed_{Q_i})$
	acceleration_mean	$mean(acceleration_{Q_i})$
	acceleration_variance	$var(acceleration_{Q_i})$
	direction_mean	$mean(direction_{Q_i})$
	direction_variance	$var(direction_{Q_i})$
	directionChange_mean	$mean(directionChange_{Q_i})$
	directionChange_variance	$var(directionChange_{Q_i})$
Network-based features	episdoeRoadType	$\{highWay, busLine, metroLine, pathWay, \cdots\}$
	additionalRoadInfo	$totalLength, avgSpeed, \cdots$
POI-based features	episodeStartPOIType	$\{busStop, metroStop, restaurant, home, \cdots\}$
	episodeEndPOIType	$\{busStop, metroStop, restaurant, home, \cdots\}$

Table 5.1: Features for inferring transportation mode

populated urban areas have several candidate POIs for a stop. Further, low GPS sampling rate due to battery outage and signal losses makes the problem more intricate.

In this thesis, we design a HMM (*Hidden Markov Model*) based technique for semantic annotation of *stops*. Unlike most other algorithms to identify POIs [ABK+07][XDZ09], an unique novelty of our approach is that it works for densely populated area with many possible POI candidates for annotation, thus catering to heterogeneous people and vehicle trajectories. It also enables identifying the activity (behavior) behind the stop, thus semantically annotating the trajectory with such information.

5.5.1 HMM-based Stop Sequence Modeling

HMM is a largely-used and classical statistical signal model in which the system being modeled is assumed to be a Markov process with unobserved state [Rab90]. We consider the *temporal sequence* of GPS stops: $\mathcal{S} = (S_1, S_2, \cdots, S_n)$ as the observed values (actually the real observation is the raw GPS point sequence, while the stop sequence is the observation we computed through trajectory structure – see the tops of Figure 5.16). Dense urban areas can have several different POIs. For example, the Milan dataset in our experiments has 39,772 POIs with largely varying density. Such large number makes it probabilistically intractable to infer the exact POI from imprecise location records. However, the number of *types* (or categories) of POIs usually is tractable. The Milan POI dataset is divided into five top-categories, i.e., *services*, *feedings*, *item*

Algorithm 5.2: ***Trajectory annotation with LOIs***

Input: (1) a move episode of raw trajectory $\mathcal{T}$ of GPS points $\{Q_i(x_i, y_i, t_i)\}$
(2) a set of road segments $P_{line} = \{r_1, r_2, \cdots, r_m\}$
(3) some training data with tags $\langle x, y, t, tag\rangle$

Output: semantic trajectory as a sequence of episodes with tags
$\mathcal{T}_{line} = \{ep_1, ep_2, \cdots, ep_m\}$

begin
- $\mathcal{T}_{line} \leftarrow \emptyset$; //initialize the trajectory
- $preMode \leftarrow null$, $preSeg \leftarrow null$;
- $move \leftarrow \{r_a, r_{a+1}, \cdots, r_b\}$; // get the map matched road segments by Algorithm 4.1;
- **forall the** $r_i \in move$ **do**
  - $r_i \leftarrow \langle f_1, \cdots, f_k\rangle$; //calculate the feature vector
  - **if** *classifier not exist* **then**
    - calculate features for episodes with known tags;
    - $transportModeLeaner \leftarrow$ train the classifier;
  - $currMode \leftarrow transportModeLeaner(r_i)$; // test classifier on r_i;
  - **if** $preMode \neq null$ *and* $preMode = currMode$ **then**
    - $preSeg \leftarrow preSeg \bigcup r_i$; // merge the two segments;
  - **else**
    - $\mathcal{T}_{line} \leftarrow \mathcal{T}_{line} \bigcup \langle preSeg, preMode\rangle$; // add the previous segment
    - $preMode \leftarrow null$, $preSeg \leftarrow null$; // renew a null episode
  - $preMode \leftarrow currMode$;
- **return** a semantic trajectory with move episode tags $\mathcal{T}_{line}$

sale, *person life*, and *unknown*. POI categories add significant semantic content to the stop for activity inference (e.g., Sally stopped for *lunch*), which becomes a tractable problem.

Figure 5.16 expresses the resultant HMM problem. The initial input is the raw trajectory $\mathcal{Q}$, i.e., the sequence of (x,y,t) points; a *sequence of stops* is computed and forms the HMM observation (O); the *exact* POI data are the superficial hidden states, whilst the POI *categories* are the real hidden states that we are interested in. Our goal is to identify the real hidden states and use them to annotate the stops.

To learn the HMM model for such stop sequences in trajectory, this problem can be formalized as follows: Let there be m POI categories $C_1 \ldots C_m$, we need to learn the three major components of a HMM problem (λ), i.e., $\lambda = (\pi, \mathcal{A}, \mathcal{B})$; π is the probability of the initial states, i.e., $Pr(C_i)$, $\mathcal{A}$ is the state transition probability matrix ($[Pr(C_j|C_i)]_{m\times m}$), and $\mathcal{B}$ is the observation probability for each state $Pr(o|C_i)$.

- **Initial Probabilities (π).** Based on the sound category of our POI data, we approximate the probability of initial states π as the percentage of POI samples belonging to each category from the information source. Therefore, for the Milan POI dataset, the initial probability vector is shown in Table 5.2.

Table 5.2: An example of initial probability vector

$\pi = \{\frac{4339}{39772}, \frac{7036}{39772}, \frac{12510}{39772}, \frac{15371}{39772}, \frac{516}{39772}\}$

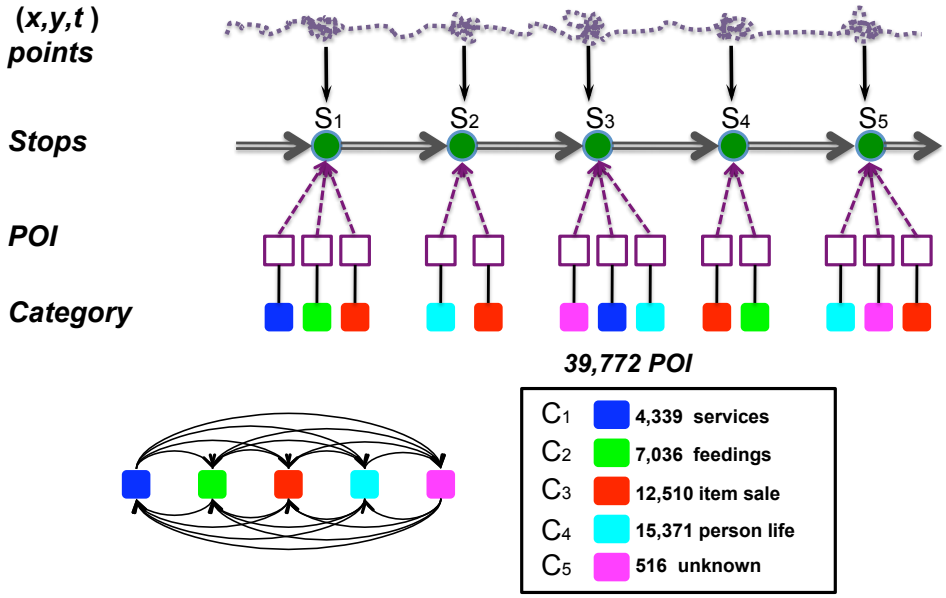


Figure 5.16: HMM formalism for inferring POI category

- **State Transition ($\mathcal{A}$).** State transition probability $Pr(C_j|C_i)$ in our formulation represents the possible stop (activity) sequence of user; i.e., probability to perform activity in places belonging to category C_j given his prior activity in places belonging to category C_i. Wherever available, activity sequences (e.g., *home*→ *work* → *shop or swim*→ *home*) are obtained through other information sources (e.g., from *region* transitions). For trajectories having insufficient history, we initialize the state transition matrix following nomenclatures of the POI categories and object type (e.g., associate high probability for meaningful state transitions and low probabilities for non-meaningful state transitions in Table 5.3). *Learning* dynamic and personalized transition matrix $\mathcal{A}$ is interesting but also challenging as a future study beyond the current result in this thesis.

Table 5.3: An example of state transition matrix

$$\mathcal{A}=\begin{pmatrix} 0.8 & 0.05 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.8 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.8 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.8 & 0.05 \\ 0.15 & 0.15 & 0.15 & 0.15 & 0.4 \end{pmatrix}$$

- **Observation Probabilities ($\mathcal{B}$).** $Pr(o|C_i)$ intuitively represents the probability of seeing a *stop* o (as the observation) in $\mathcal{T}$ *caused by* user's interest in places belonging to category C_i. $Pr(o|C_i)$ can be approximated by the center of the *stop* $Pr(center_{xy}|C_i)$ or the bounding rectangle $Pr(boundRectangle|C_i)$.

Computing $\mathcal{B}$ for areas having high POI density is not easy. Our solution is based on the intuition that influence of a POI category on a *stop* is proportional to the number of exact POIs of that category in the stop area. We model the influence of a POI as a two-dimensional Gaussian

distribution – the mean is the POI's physical position (x, y) and the variance is $[\sigma_c^2, 0; 0, \sigma_c^2]$, where σ_c is category specific. Figure 5.17 displays an example of 12 POIs' Gaussian distributions with the corresponding densities in Figure 5.18. By Bayesian rule, we deduce the lemma to determine $Pr(o|C_i)$ in $\mathcal{B}$.

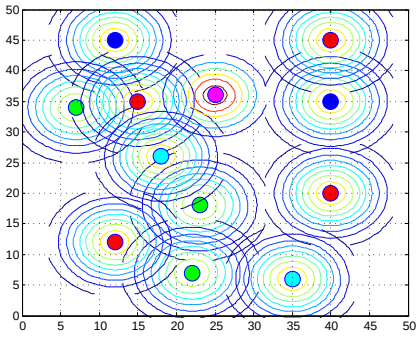


Figure 5.17: POI examples

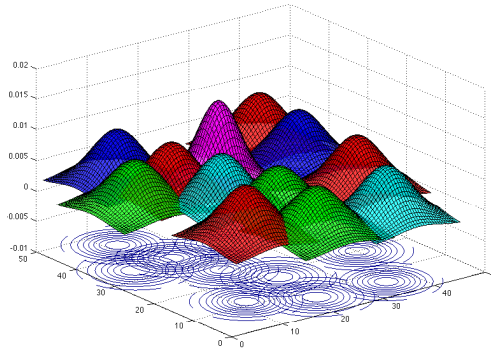


Figure 5.18: POI densities

Lemma 1. $Pr(o|C_i)$ is proportional to the sum of the probability of each POI that belongs to this category C_i, namely $Pr(o|C_i) \propto \Sigma_j Pr(o|poi_j^{(C_i)})$.

Proof. of Lemma 1

$$\begin{aligned} Pr(o|C_i) &= \frac{Pr(o, C_i)}{Pr(C_i)} = \frac{\Sigma_j Pr(o, poi_j^{(C_i)})}{\Sigma_j Pr(poi_j^{(C_i)})} \\ &= \frac{\Sigma_j Pr(o|poi_j^{(C_i)}) Pr(poi_j^{(C_i)})}{\Sigma_j Pr(poi_j^{(C_i)})} \\ &\propto \Sigma_j Pr(o|poi_j^{(C_i)}) Pr(poi_j^{(C_i)}) \\ &\propto \Sigma_j Pr(o|poi_j^{(C_i)}) \end{aligned}$$

□

For the sake of the computing efficiency, we have two strategies to enhance the calculation of $Pr(o|C_i)$: *discretization* and *neighboring*. In our current case study, we choose the center (x, y) as our stop position. Instead of directly computing the on-line $Pr(\langle x, y\rangle|C_i)$ values, we discretize it in terms of dividing space into grids, and calculate the discretized value for each grid $Pr(grid_{jk}|C_i)$ in advance. Therefore, we can approximate $Pr(\langle x, y\rangle|C_i)$ by the pre-computed $Pr(grid_{jk}|C_i)$ when point $\langle x, y\rangle \in grid_{jk}$ (see Figure 5.19). The second strategy is *neighboring*, which means we do not consider the complete POI set but only the neighboring POI for a given position $grid_{jk}$, namely the black searching rectangle for the triangle position displayed shown in Figure 5.19. With these two strategies, we can efficiently get the histogram table of $\mathcal{B}$, which is a conditional probability table with the probability element $[j, k, i] = Pr(grid_{jk}|C_i)$.

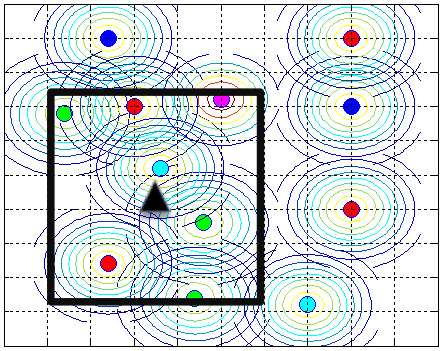


Figure 5.19: POI discretization and neighboring

5.5.2 Inferring Stop Activities

Using the above defined complete form HMM $\lambda = (\pi, \mathcal{A}, \mathcal{B})$, we infer their hidden states (the intrinsic purposes or behaviors behind the stops) $HS = \{pc_1, pc_2, \cdots, pc_n\}$ from the stop sequence $OV = \{stop_1, stop_2, \cdots, stop_n\}$ (observed / pre-computed), available through the stop/move computation; where pc_t is the POI category $pc_t \in \{C_1, \cdots, C_m\}$. This problem can be reformulated as an optimization problem with an objective to maximize the likelihood $\mathcal{L}(HS|OV, \lambda)$ (see Formula 5.1).

$$\begin{aligned} \mathcal{L}(HS|OV,\lambda) \quad where \quad & \lambda = (\pi, \mathcal{A}, \mathcal{B}) \\ & OV = \{Stop_1, Stop_2, \cdots, Stop_n\} \\ & HS = \{pc_1, pc_2, \cdots, pc_n\}, pc_i \in \{C_1, \cdots, C_5\} \end{aligned} \tag{5.1}$$

We redefine this problem as a *dynamic programming* problem, defining $\delta_t(i)$ as the highest probability of the t^{th} *stop* caused due to POI category C_i (see Formula 5.2). Afterward, Formula 5.3 gives the corresponding induced form of the highest probability at the $(t+1)^{th}$ stop for category C_j, considering the HMM state transition probabilities (A_{ij} in $\mathcal{A}$) and the observation support ($B_j(o_{t+1})$ in $\mathcal{B}$). We record the previous state C_i that gives the highest probability to current state C_j by $\psi_{t+1}(j)$ (see Formula 5.4).

$$\delta_t(i) = \max_i Pr(pc_1, \cdots, pc_t = C_i, o_1, \cdots, o_t|\lambda) \tag{5.2}$$

$$\delta_{t+1}(j) = \max_i \{\delta_t(i) A_{ij}\} \times B_j(o_{t+1}) \tag{5.3}$$

$$\psi_{t+1}(j) = \operatorname*{argmax}_i \quad \delta_t(i) A_{ij} \tag{5.4}$$

Finally, we employ the Viterbi algorithm [For73] to solve this dynamic programming problem for inferring the hidden state (stop category) sequence. We first recursively compute $\delta_t(i)$, and deduce the final stop state with the highest probability in the last stop, then backtrack to the previous stop state by $pc^*_{t-1} = \psi_t(pc^*_t)$. An example of such "*forward computing*" and "*backward tracking*" is shown in Figure 5.20, where the stop is annotated with POI category. The details of the algorithm for inferring hidden stop category sequence is described in Algorithm 5.3. The output of this layer is a sequence of semantic episodes describing the stops.

Algorithm 5.3: ***Trajectory annotation with POIs***

Input: (1) an observation sequence of stops
$O = \{Stop_1, Stop_2, \cdots, Stop_n\}$; (2) points of interest
$POIs = \{\langle p_1, q_1\rangle, \cdots, \langle p_k, q_k\rangle\}$ where $q_i \in \{C_1, \cdots, C_5\}$
Output: a hidden state sequence about stop behaviors (in terms of POI categories), i.e.,
$S = \{q_1, q_2, \cdots, q_n\}, q_i \in \{C_1, \cdots, C_5\}$
begin
  /* learn the model from POIs */
  $\lambda = (\pi, \mathcal{A}, \mathcal{B})$
  /* initialization */
  forall the *POI category* C_i **do**
    $\delta_1(i) = \pi_i B_i(o_1), 1 \leq i \leq N$; $\psi_1(i) = 0$
  /* recursion */
  forall the *t: 2 to n* **do**
    forall the *categories* C_j **do**
      $\delta_t(j) = \max_i[\delta_{t-1}(i)A_{ij}] \times B_j(o_t)$
      $\psi_t(j) = \operatorname{argmax}_i[\delta_{t-1}(i)A_{ij}]$
  /* termination */
  $P^* = \max_i[\delta_T(i)]$; $q_n^* = \operatorname{argmax}_i[\delta_T(i)]$
  /* state sequence backtracking */
  forall the *t: n to 2* **do**
    $q_{t-1}^* = \psi_t(q_t^*)$
  /* get the semantic trajectory with POI tags */
  $S = \{\langle stop_1, q_1\rangle, \cdots, \langle stop_n, q_n\rangle\}$
  summarize $\mathcal{T}_{point}$ from extracted POI sequence ($\langle stop, t_{in}, t_{out}, tagList\rangle$).
  return semantic trajectory $\mathcal{T}_{point}$

To put forward, given a variety of trajectories (e.g., mobility data belonging to different people, tracked in different time duration), we want to simply classify the trajectory based on the computed semantic stop sequence, where each stop is associated to a hidden POI category, such as *services*, *feedings*, *item sales*, *person life* or *unknown*. The intuitional method is based on the sum of the total stop time for each category, as the following unconstrained optimization problem, namely *the trajectory type is the POI category that has the maximum stop time*.

$$traj\_type = \operatorname*{argmax}_{C_i} \sum_{stop.POIType=C_i} (stop.time_{out} - stop.time_{in}) \tag{5.5}$$

To summarize, the complete procedure of annotating *stop* episodes is the following: (1) initialize the HMM problem, compute the three major components, namely π, $\mathcal{A}$, and $\mathcal{B}$; (2) given a mobility trace (a original trajectory), apply relevant stop identification algorithms (e.g., velocity or density method) to discover stop sequence; (3) take the stop sequence as the observation, and deduce the hidden POI category sequence by using the Viterbi algorithm on the transformed form of the problem; and (4) finally assign a POI category for this trajectory, according to the maximum total stop time defined in Formula 5.5.

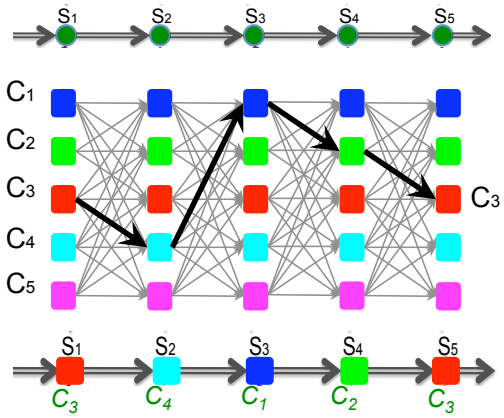


Figure 5.20: POI category annotation result

5.6 Experiment

To validate our semantic annotation approach, this thesis has implemented the annotation framework and carried out extensive experiments to annotate large GPS trajectories of heterogeneous moving objects with varying data qualities, from tracks of private cars, taxis, to GPS embedded smartphones carried by people. In [YCP+11], we have published some initial results of trajectory semantic annotation, where we built a system framework called "SeMiTri" (*Semantic Middleware for Trajectories*, see Figure 5.2) including the dedicated annotation layers.

5.6.1 Implementation Setup

The SeMiTri system with annotation functionality is implemented and deployed on a Linux operating system - Ubuntu 9.10, with the Intel(R) 2×3.00GHz CPU and 7.9GiB memory. The context computation and semantic annotation algorithms are implemented in Java 6; PostgreSQL 8.4 with spatial extension PostGIS 1.5.1 is used for implementing the different database stores. The raw GPS records and geographic information from third party sources are loaded into the databases in different tables and queried by different layers during execution time, previously shown in Figure 5.2.

As the main output of SeMiTri, the annotated *semantic trajectories* are stored in the *Semantic Trajectory Store*. Dedicated database storing tables are designed for *GPS records*, *trajectories*, *stops/moves*, and *annotations with geographic data*, using some new datatypes we defined for trajectories in PostGIS. This is expected to be queried by several trajectory applications. In our experiments, since the datasets are huge, we highlight large-scale aggregated results through the *Semantic Trajectory Analytics Layer*. This layer computes additional statistical information on the trajectories at all of the different abstraction levels. In addition, we have developed a *Web Interface* as a pilot application using Apache 2.2.12 Web Server and Tomcat 6.0.26 Application Server. This provides trajectory querying and visualization services to end-users in trajectory applications [YSC+10].

Our experiments focus on vehicle (taxi, private cars) and people trajectories. While transportation mode of vehicles is trivial (depending on the vehicle type), people can take different choices in several or even one trajectory. Note that trajectories might have varying semantic

annotations due to varying amount of third party sources available. For some scenarios, SeMitri produces partial annotations, while exploiting the available information sources maximally. Our objective here is to bring out the *individual* performance of each layer, which results in the collective annotation of the overall trajectory. To do this, we present annotation performance of each layer with different third party sources tuned to test the layers individually and all together.

5.6.2 Semantic Place Data Sources

Recall the experimental study of offline trajectory computing in Section 4.6.1, we have presented the trajectory datasets that largely used in this thesis, including the homogeneous vehicle trajectories as well as the heterogeneous people trajectories. Based on the previous trajectory computing results in Chapter 4, this section focuses on semantically annotating such datasets and the computed trajectories, with additional third party semantic data sources. It is worth noting that different vehicle trajectory data might have varying third party information contents. SeMitri would produce partial annotations, but exploit available information sources maximally.

For vehicle trajectories, we validate: (1) spatial join based *Semantic Region Annotation* on Lausanne taxi data and the people trajectory data with the available of landuse sources; (2) HMM based *Semantic Point Annotation* on Milan private cars data with the available of POIs; (3) map matching based *Semantic Line Annotation* on a benchmark dataset about a 2-hour drive in Seattle provided by Newson and Krumm[5] from Microsoft, as this data set has the ground truth path. The summary of such data of vehicle trajectories has already been shown in the first part of Table 4.2. The *Semantic Place Data Sources* include: (1) the landuse data of Switzerland on the taxi data, (2) a large POI dataset of Milan on the Milan private car data, and (3) the benchmark dataset containing the road network of Seattle as well as the ground truth path.

Type	***Name***	***Details***
annotation for vehicle trajectories	(1) Lausanne (Switzerland)	1,936,439 grids
	(2) Milan: points of interest	39,772 POIs
	(3) Seattle network (Krumm's benchmark)	158,167 road lines
annotation for people trajectories	(1) Lausanne (Switzerland)	1,936,439 grids
	(2) Swiss map from OSM	109,954 points, 344,975 lines, 233,896 regions

Table 5.4: Datasets of semantic places (third party sources for annotation)

People trajectories are far more non-homogeneous than vehicle trajectories: (1) Many phenomena can result in GPS data loss, such as the limited power of smartphones, battery outage, and indoor signal loss. (2) Non-stationary sampling rates due to on-chip power saving software modules that monitor the sensor; (3) Compared with vehicles, users in people trajectory can take complicated on-road/off-road routes, and choose diverse transportation modes (e.g., *walk*,

[5] `http://research.microsoft.com/en-us/um/people/jckrumm/MapMatchingData/data.htm`

bicycle, *bus*, *metro*) during their daily movements. Therefore, capabilities of SeMiTri are well established through systematic semantic enrichment of such trajectories. Our experiment validates the people trajectory dataset provided by Nokia Research Center, Lausanne [KBD+10]. They distributed smartphones (Nokia N95) to students/researchers in Lausanne, collecting several *people sensing* data including GPS feeds. We analyzed 185 users, with 23,188 daily trajectories with 7.3M GPS records from this data. Previously, Table 4.3 also describes the details of 1,077 daily trajectories of 6 specific users we know. For *Semantic Place Data Sources* used in annotating people trajectories, we use the swiss landuse data, and extract additional geographic data (through OSM files) from Openstreetmap that includes regions, POIs, road networks of several types, and load them into our PostGIS data store (using Osm2pgsql [6]). The details of these third party data sources for semantic annotation are listed in Table 5.4, including both the sources for vehicle trajectories and people trajectories.

5.6.3 Trajectory Annotation with Landuse

As the result of offline trajectory computing, we produce 172 daily trajectories with 1,824 *moves* and 1,786 *stops* over the Lausanne taxi data. Based on this, the *Semantic Region Annotation Layer* annotates the raw trajectories and the trajectory episodes (stops/moves) with the landuse data. Figure 5.21 shows the detailed landuse category distribution over taxi trajectories. Landuse has 4 large categories and 17 sub-categories (from 1.1 to 4.17, see Figure 5.7). We observe that most of the taxi GPS records are in *building areas (1.2)* (46.6%) and *transportation areas (1.3)* (36.1%), nearly 83% GPS points belonging to these two categories (see the trajectory column in Figure 5.21). The *move* part covers 79.25% of the taxi landuse area, whilst the *stop* part only covers 20.75%. Due to the high-level abstraction into region-based movements, the resultant semantic trajectory $\mathcal{T}_{region}$ representation achieves almost 99.7% storage compression (e.g., 3M GPS records can be annotated with only 8,385 Landuse grids).

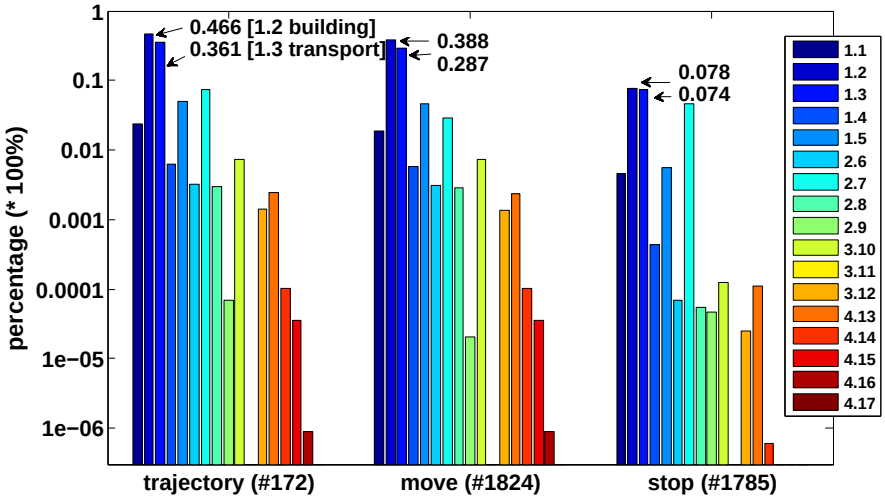


Figure 5.21: Landuse distribution for annotating taxi data

Figure 5.22 displays these landuse grids within the Google earth plugin on the *Web interface layer* of SeMiTri. This is done in terms of a dynamic KML file that generated using the query

[6] http://wiki.openstreetmap.org/wiki/Osm2pgsql

results from the database. The color (from total white to total black) of the grid indicates the density of GPS records - the deeper color, the more GPS records inside.

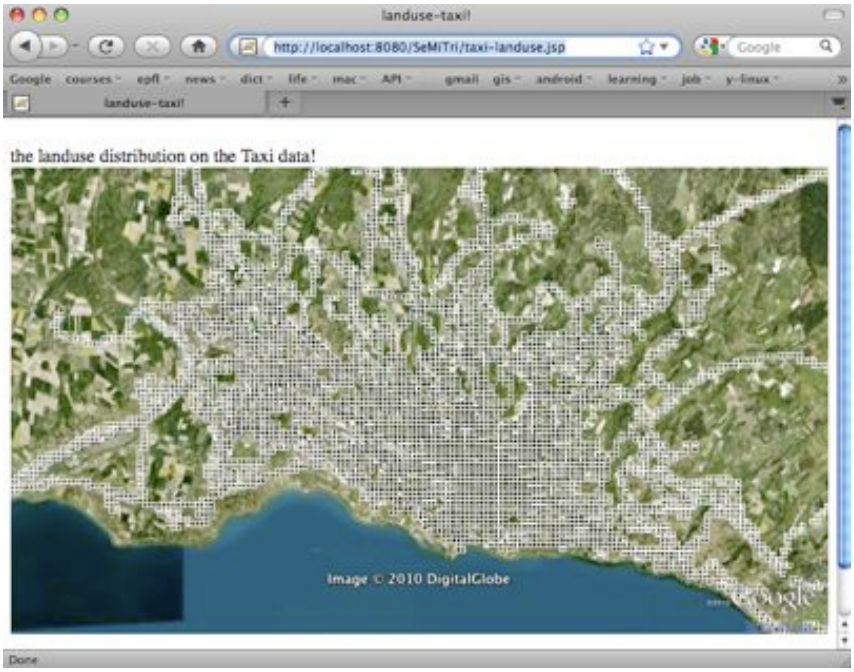


Figure 5.22: Landuse distribution for taxi data on SeMiTri's Web Interface

Similar to the semantic region annotation on the taxi trajectories with the Landuse data shown in Figure 5.21, we observe similar results in general for the people trajectories: many of the stops/moves/trajectories are staying in the *building areas (1.2)* (33.3%) and the *transportation areas (1.3)* (28.6%). However, while 83% of stops/moves in taxi trajectories (Figure 5.21) are in these two types of areas, only 61% of people trajectories fall in them. This is likely and intuitive, showing people trajectories have much more variations in their movements and areas covered, compared with the taxi trajectories. To discover further insights, Figure 5.23 shows the precise distribution of the six people selected (with the list of top-5 categories for each user). We observe, *user*3 has much higher percentage of location records in *woods areas (3.12)* because his accommodation is in such a wood place close to the Geneva lake; *user*4's higher percentage in *industrial and commercial area (1.1)* because of her house in a commercial center; and *user*2's higher percentage in *forest (3.10)* because he does a lot of hiking and skiing, etc., automatically identifying these kinds of specific and individual location information and behaviors that are different from other users.

5.6.4 Trajectory Annotation with Road Networks

Regarding purely map matching for network-constrained trajectories, Section 4.6 has already shown some results in the experiment of offline trajectory computing, namely the network-constrained trajectory data cleaning via map matching. Therefore, this section is more interested in further annotating with heterogeneous road networks for people trajectories, in terms of the inference of the underlying transportation modes that are used during moves.

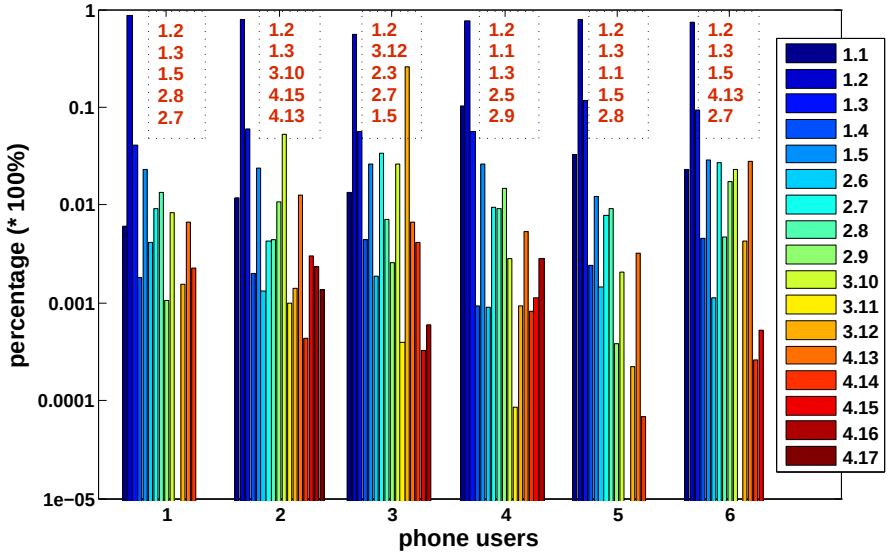


Figure 5.23: Landuse category distribution and top-5 categories in people trajectories

Apart from performing map-matching, people trajectories in *Semantic Line Annotation* are enriched to determine the transportation mode (e.g., *metro, bus, walk etc*). Our *Semantic Line Annotation Layer* considers the underlying network information along with the velocity/acceleration distribution for each road segment from the initial map-matching results to determine the transportation mode. For example, Figure 5.24 shows a typical *home-office* trip of *user*4, who walked a few blocks from home, then took the Metro line, and finally walked from the Metro stop to his office: sub-figure (a) shows the original GPS points; (b) displays the initial map-matched road segments for these GPS points; (c) further infers the corresponding different transportation modes such as *metro* or *walk*; and finally (d) summarizes the trajectory in terms of meaningful road sequences stored in the semantic trajectory store.

Besides taking the *metro* as shown in Figure 5.24, *user*4 has taken three other transportation modes: *bus*, *bike* or *walk*. In Figure 5.25, the left subfigure (a) shows an example of using *bike* for going home to office; whilst in subfigure (b) the user took the *bus*, with *walking* as well during the beginning and ending parts of the home-office move, for getting on/off the buses.

From his 161 daily trajectories, we extracted 186 direct trips between home and office (including both *office*→*home* and *home*→*office*), and inferred 66 *bike*-, 39 *metro*-, 49 *bus*-, and 32 *walk*- moves respectively, as shown in Figure 5.26.

5.6.5 Annotation with Points of Interest

We analyze the performance of the HMM-based *Semantic Point Annotation* algorithm using the POI data in Milan. The 39,772 POIs are divided into 5 categories: 4,339 *services*, 7,036 *feedings*, 12,510 *item sale*, 15,371 *person life* and 516 *unknown* (Figure 5.27 - first column). The third party sources have a high density of POIs in this area. Traditional one-to-one match methods like [XDZ09] are not suitable here. Our *Semantic Point Annotation* layer enriches the *stop*s computed from the GPS tracks of the private cars and extracts the most probable POI category for each stop. In Figure 5.27 (second column), we observe that most of the stops (about 56.3%)

Figure 5.24: Move annotation - a home-office example (via Metro)

Figure 5.25: Home-office (via Bike and via Bus)

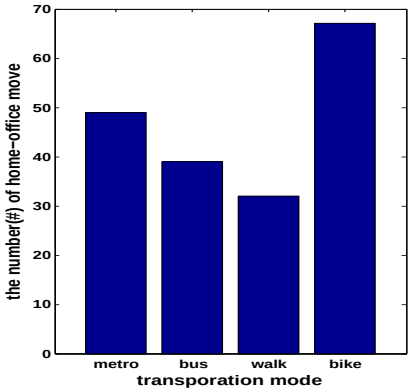


Figure 5.26: Transport modes of *Home-office* moves (phone user 4)

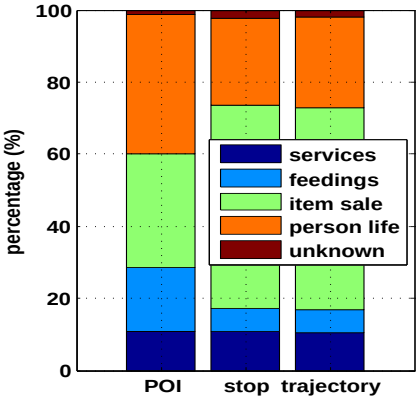


Figure 5.27: Semantic stops/trajectories w.r.t. POIs by point annotation

belong to *item sale* (shopping, groceries etc.), with the next one being *person life* (e.g., sport) (about 24.2%); this result makes intuitive sense for private car trajectories.

Through well-defined rules, SeMiTri is able to perform analytics over the extracted semantic trajectories. For example, Figure 5.27 (third column) also shows the *trajectory category*, which is defined as: *the category of* $\mathcal{T}$ *is the category which has the maximum stop time* (Formula 5.5). This can be considered as a *semantic classification* over the complete trajectories. Note that the distribution of *trajectory* categories is statistically similar to the distribution of *stop* categories (see Figure 5.27). This is because the dataset has only 1.7 stops per trajectory on the average – 2M GPS records of 77,694 trajectories are computed into 133,556 stops, thereby resulting in a similar distribution. This is co-incidental and depends largely on the trajectory data in applications.

Our POI dataset of Lausanne area is sparse at the moment, and does not reflect the real-life POI density of the area (compared with the much more complete POI data in the Milan city we got). We will apply the semantic point annotation method when more POIs with well-defined categories are available. We are currently building such dataset for EPFL campus.

5.6.6 Web Interface and Performance

The above experiments validate the semantic annotation part of SeMiTri, the capability of annotating *heterogeneous* trajectories comprehensively and creating structured semantic trajectories, by maximally exploiting available third party geographic information sources that contain various types of semantic information to enrich trajectories.

Additionally, we develop a *Web Interface* for query and visualization of the annotation results from SeMiTri. This enables users to easily query their raw GPS traces, episodes in structured trajectories, as well as semantic trajectories through a web browser with Google Earth Plugins. The visualization capabilities of such web interface can support several aspects of trajectories,

- *Spatio-Semantic Trajectories* - Demonstrates multiple levels of trajectory data abstraction, showing *raw GPS tracks*, *spatio-temporal trajectories* (exploiting space/time gaps), *structured trajectories* (e.g., stops/moves), and *semantic trajectories* (e.g., the semantic trajectory $home \rightarrow office \rightarrow supermarket \rightarrow home$ in the thesis introduction).

- *Semantic Places* - Demonstrates diverse third party semantic data resources (in particular the geographic data) that are associated to trajectories - *Landuse*, *Road network*, and *Point of interest* (POI) data.
- *User Interactions* - Provides a friendly Web interface to query and visualize trajectories (e.g., daily tracks) at different abstraction levels.
- *Analytics Results* - Highlights *statistical analytics* results of semantic trajectories, e.g., the average speed when user is moving, Landuse distribution where user has stopped, most frequent transportation modes, etc.

Figure 5.28 shows SeMiTri's Web interface with a schematic example of a daily trajectory (at 30/3/2010) for a given phone user (id 205). The interface displays four parts: (a) the query input, (b) the visualization with Google Earth plugin, (c) the checkboxes that allow user to incrementally and selectively visualize different parts/levels of trajectories as well as semantic places they traverse, and (d) the aggregated statistics associated with the trajectories from different perspectives.

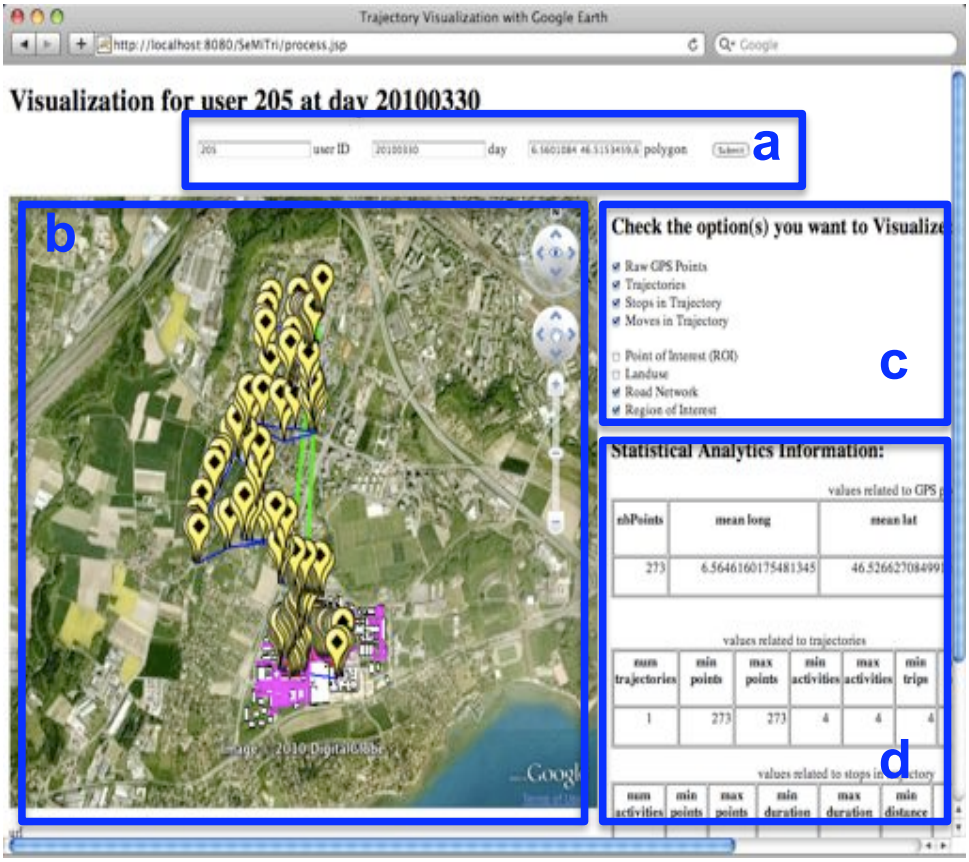


Figure 5.28: Web Interface of SeMiTri

Finally, we analyze the run-time performance of SeMiTri. Figure 5.29 summarizes the latency distribution of SeMiTri for processing phone trajectories. We observe that computation and annotation latencies are much lower (both *map-matching* and *landuse*) compared with the storing time (i.e., write the results into databases – the semantic trajectory store). For all of the six

users, the average time for *computing episodes*, *storing episodes*, *map matching annotation*, *storing matched results*, *landuse annotation* for a daily trajectory are 0.008, 3.959, 0.162, 0.292, and 0.088 seconds, respectively. Latency distributions for vehicle trajectories are also similar.

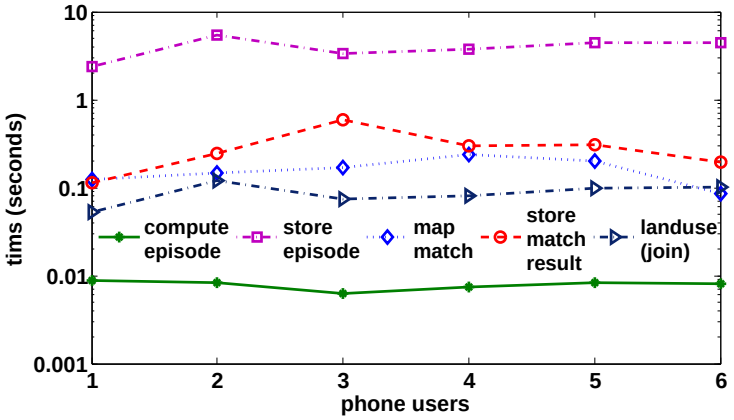


Figure 5.29: Trajectory computing and annotation latency measurement

5.7 Summary

This chapter presented the third major contribution of this thesis, i.e., the *trajectory semantic annotation*. The main outcome is a complete trajectory framework ("SeMiTri") with dedicated annotation layers to support semantically enriching heterogeneous trajectories (e.g., both vehicle and people trajectories) with heterogeneous third party geographic or application domain sources. We categorized such third party semantic data sources according to their underlying spatial extents (i.e., region, line, and point), and designed three dedicated annotation algorithms. The initial result of such heterogeneous annotation framework has already been published in [YCP+11]. Additionally, this thesis has built a prototype with a Web interface for querying and visualizing the annotated semantic trajectories as well as the collected raw mobility data, the computed spatio-temporal and structured trajectories. This prototype has been published and demonstrated in [YSC+10].

Regarding the heterogeneous annotation functionality, the objective is to maximally enrich trajectory semantics with available third party semantic data sources, including both public data like Openstreetmap and private data like Swiss Landuse resource. The three dedicated annotation algorithms are: (1) annotation with semantic regions that can support well-divided and grid-based regions like the landuse data, as well as the free-style and irregular regions; (2) annotation with semantic lines that firstly applies a global map-matching algorithm and then infers the transportation modes during trajectory's move episodes; (3) annotation with semantic points that significantly reformulates the problem as a HMM problem, infers the underlying purposes or behaviors of the stops in a single trajectory, and additionally provides semantic classification of the complete trajectories.

Chapter 6

Online Trajectory Computing

In online computation a computer algorithm must decide how to act on incoming items of information without any knowledge of future inputs.

Allan Borodin, Ran El-Yaniv, 1998

6.1 Introduction

In this chapter, we address the fourth main contribution of this thesis, i.e., *online trajectory computing*. As in Chapter 4, we have discussed the offline trajectory computing to constructing and understanding mobility data from different levels of abstraction. However, location data generated from GPS equipped moving objects are typically collected as streams of spatio-temporal (x,y,t) points that then put together form corresponding raw trajectory data. Therefore, the offline techniques in the current literature as well as our methods in Chapter 4 and Chapter 5, are not sufficient for real-time trajectory applications that rely on timely delivery of computed trajectories to serve real-time query answers. Filling this gap, this thesis additionally proposes an online platform, namely "SeTraStream", for real-time semantic trajectory construction. This online framework is capable of providing real-time trajectory data *cleaning*, *compression*, *segmentation* over streaming movement data. In addition, we explore *tagging* functionality to evaluate the online computation and achieve fully semantic-aware trajectories.

This chapter is organized as follows: Section 6.2 presents the challenges of real-time trajectory applications and our online approach; Section 6.3 addresses preprocessing tasks of the streaming movement data like online data cleaning, compression etc.; Section 6.4 addresses the online segmentation algorithms to identify the trajectory episodes; Section 6.5 exploits online tagging functionality to annotate relevant semantics; Section 6.6 shows some experimental studies; finally Section 6.7 summarizes this chapter.

6.2 Online Computing Approach

In real-time trajectory applications, the setting is even a distributed environment. At the lowest level of the system architecture, location-aware objects continuously transmit their status updates towards middle-layer sites-servers equipped with communication antennas. Sites cover non-overlapping regions and are capable of communicating with their peers via wired infrastructure. The realm of a region R_i assigned to site S_i is defined by the communication range of its antenna and can be of arbitrary shape though we henceforth refer to rectangular areas for simplicity. Objects' status updates include tuples apart from movement attribute data values and will be described in the upcoming subsection. At the upper-tier, a central coordinator receives semantic-aware trajectory representations so as to utilize them during ad-hoc or continuous query procedures. The whole architectural framework is depicted in Figure 6.1.

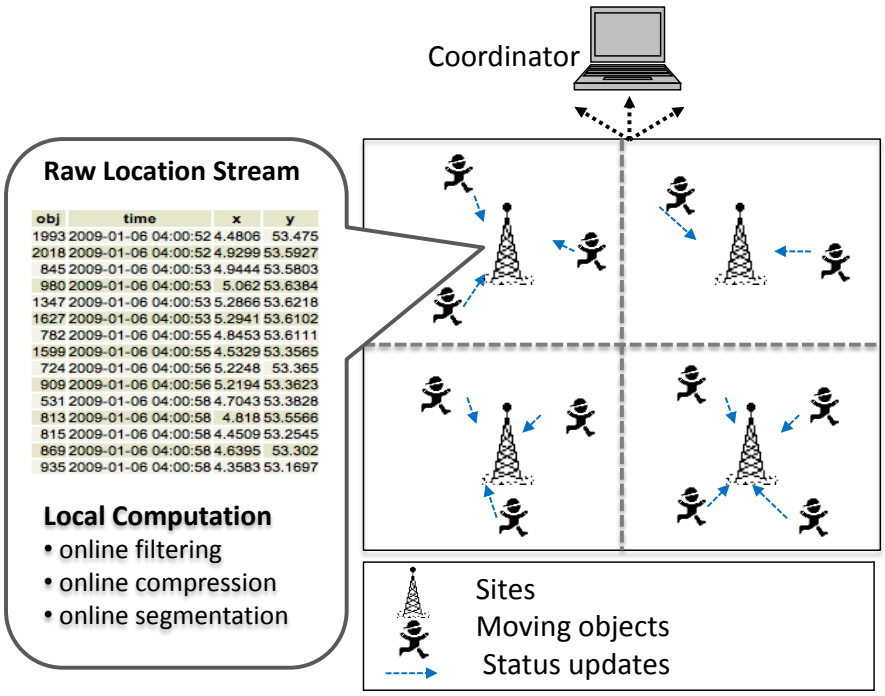


Figure 6.1: Real-life distributed setting for online trajectory computing

6.2.1 Real-time Settings and Challenges

Real-time semantic trajectory construction can be useful in many traffic monitoring scenarios where authorities are interested in identifying apart from recent (i.e., within a restricted time window) objects' trajectory representation, the behavior of the drivers by posing queries of the form: *"Report every τ secs the movement and driving behavior of the objects within area A during the last T minutes"*. In that, authorities are able to continuously diagnosing streets where the density of vehicles whose drivers tend to have aggressive (speeding, tailgating, driving at the edges of the lanes etc.) behavior has recently become high, thus enabling suitable placement

and periodic rearrangement of traffic wardens and patrol cars. As another example, state-of-the-art navigation services like Waze [1] provide the potential for combining traditional routing functionality with social networking facilities. Online semantic trajectory construction allows users to acquire a compact picture of the movement and the social activities of interconnected friends around their moving area.

Different from the offline setting in Chapter 4 where all of the mobility data is collected in advance, the online and distributed setting has servers that continuously collect the status updates of moving objects that move inside an area of interest, i.e. the monitoring area of moving objects belonging to that site (or Antenna). Firstly, such updates involving an object O_i contain spatio-temporal $\langle x, y, t \rangle$ points forming its "*Raw Location Stream*", and are transmitted to the closest site. Based on such setting, a couple of local online computing tasks for each site are required, like online cleaning of GPS errors, online data compression, online segmentation, and even online tagging. Afterward, the local results from each site need to be refined globally according to the neighborhood sites. Such refinement could be like splitting the big trajectory episodes cross two sites into small ones, or merge the small episodes into big one; or even change the status between different trajectory episodes (e.g., from stop to move). In this thesis, the current research focus is on the online computing part, not supporting the complete functionality of the distributed context.

It is non-trivial to establish a real-time semantic trajectory computation platform, even only focusing on the online case in a local site, without the consideration of many global sites in the distributed context. Compared with the previous offline solutions in Chapter 4, the online computing has several new technical challenges, as follows:

- *Efficient Computation* – Large amounts of movement data are generated continuously, therefore we need to come up with more efficient algorithms that can handle different levels of trajectories in an acceptable time – including all data processing aspects like data cleaning, compression, segmentation, and semantic tagging. Such kind of requirement on efficient computing is an almost must-have condition for all of the streaming data systems and applications.

- *Suitable Trajectory Segmentation Decision Making* – In offline trajectory construction, many existing algorithms need to tune a lot of thresholds placed on movement features (like *acceleration, direction alteration, stop duration etc.*), so as to find their most suitable values, sometimes in a per object fashion. However, in the real-time context the movement attribute distribution may tremendously vary over time and continuous parameter tuning is prohibitive for the online delivery of semantic trajectory outcomes. Thus, suitable techniques should not rely on many predefined thresholds on certain movement features but instead consider pattern alterations under movement pattern similarity estimations throughout the online computation stages. This only allows a similarity threshold on comparing movement patterns.

[1] `http://world.waze.com/`

- *Semantic Trajectory Tagging* – After trajectory segmentation (i.e., identifying the suitable episode division points), additional semantic tags need to be explored for episodes, e.g., characterization of the activity (shopping, work) of a stop episode, or transportation modes (e.g., car, metro, bus) that are taken during a move episode. Figure 6.2 shows an exemplary sketch of such a semantic-aware tagged trajectory, which is quite similar to the semantic trajectory example in the previous chapters, but generated from the streaming movement data in a real-time setting.

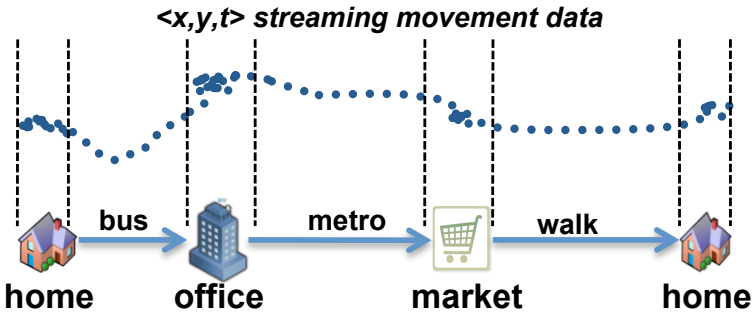


Figure 6.2: From streaming movement data to semantic trajectory

6.2.2 Window Specifications for Online Computing

The window-based sequential data analysis is a fundamental concept in streaming data processing [GHP+03]. Before providing the detailed window specification in our online computing approach, we present some preliminary definitions that are intensively used in this online computing chapter.

In our online setting, each local server continuously collects the status updates of moving objects that move inside an area of interest, i.e., the monitoring area of moving objects. As the initial input, such updates involving an object O_i contain spatio-temporal $\langle x, y, t\rangle$ points which form the "*Raw Location Stream*".

Definition 6.1 (Raw Location Stream)**.** The continuous recording of spatio-temporal points that update the status of a moving object O_i, i.e., $\langle Q_1^{\ell s}, Q_2^{\ell s}, \ldots, Q_n^{\ell s}\rangle$, where $Q_i^{\ell s} = \langle x, y, t\rangle$ is a tuple including moving object's O_i, position $\langle x, y\rangle$ and timestamp t.

By means of the raw location streams, we can derive information of movement features such as *acceleration*, *speed*, *direction* etc., which make up a "*Location Stream Feature Vector*" ($Q^{\ell f}$). Moreover, depending on the application, updates include additional attributes such as heading, steering wheel activity, lane position, distance to headaway vehicle (e.g to assess tailgating), displacement and so on. These features formulate a "*Complementary Feature Vector*" (Q^{cf}). Consequently, the two types of feature vectors combined together are forming the "*Movement Feature Vector*" ($\Omega = \langle Q^{\ell f}, Q^{cf}\rangle$) of d dimension describing d attributes of O_i movement at a specific timestamp.

Definition 6.2 (Movement Feature Vector)**.** The movement attributes of object O_i at timestamp t can be described by a d-dimensional vector that is the concatenation of the location stream feature vector and the complementary feature vector $\mathcal{Q} = \langle Q^{\ell f}, Q^{cf} \rangle$.

- Location Stream Feature Vector ($Q^{\ell f}$) – The movement features of object O_i that can be derived from the raw location stream tuple $Q^{\ell s}$.
- Complementary Feature Vector (Q^{cf}) – The movement features of an object that cannot be derived from the location stream but are explicitly included in O_i's status updates.

To provide better understanding and mobility data abstraction, we have already defined the concept of "semantic trajectory" in Chapter 3, where the trajectory is thought of as a sequence of meaningful episodes (e.g., stop, move, and other self-contained and self-correlated trajectory portions). Similarly, we adopt this definition in the real-time setting, that needs to be computed from streaming movement data.

Definition 6.3 (Semantic Movement)**.** A semantic movement or trajectory consists of a sequence of meaningful trajectory units, called "episodes", i.e., $\mathcal{T}_{str} = \{e_{first}, \ldots, e_{last}\}$

- An episode *(e)* groups a subsequence of the location streams (a number of consecutive $\langle x, y, t \rangle$ points) having similar movement features.
- From the semantic data compression point of view, an episode is a tuple that stores only the subsequence's temporal duration and spatial extent. $e_i = (time_{from}, time_{to}, geometry_{bound}, tag)$.

The $geometry_{bound}$ is the geometric abstraction of the episode, e.g., the bounding box of a stop area or the shape trace of roads that the moving object has followed. The term *tag* in the last part of the previous definition refers to the semantics of the episode, i.e., characterization of the activity or means of movement that is taking place in an episode.

Based on these definitions, we design a sliding window paradigm to find episode "division points" in trajectories according to the comparison between movement patterns in different time intervals of trajectory's lifespan. We apply a modified version of the logarithmic tilted time window [GHP+03], for the detection of both short term and long term alterations in the pattern of an O_i's movement. Figure 6.3 briefly sketches the data flow: the initial input is the "*Raw location streams*"; we compute their "*Movement feature vectors*" on the fly; based on the "*Fragmented time window*", we can additionally compute the movement patterns in windows, and finally identify the episode division point for computing such "*semantic-aware trajectory*". The more detailed systematic procedure will be shown in a later section about system framework and algorithms.

In our context, the time window size T expresses the most recent portion of semantic trajectories the server needs to be informed about. An additional parameter τ specifies a time interval in which client side devices, installed on moving objects, are required to collect and report batches of their time ordered status updates [GHP+03]. Thus, $\frac{T}{\tau}$ batches are included in the window. Obviously, posed prerequisites are: (1) $\tau \ll T$ and (2) $T \mod \tau = 0$. As the window slides, for each monitored object O_i, the most aged batch expires and a newly received one is appended in it. The size of τ may vary from a few seconds to minutes depending on the

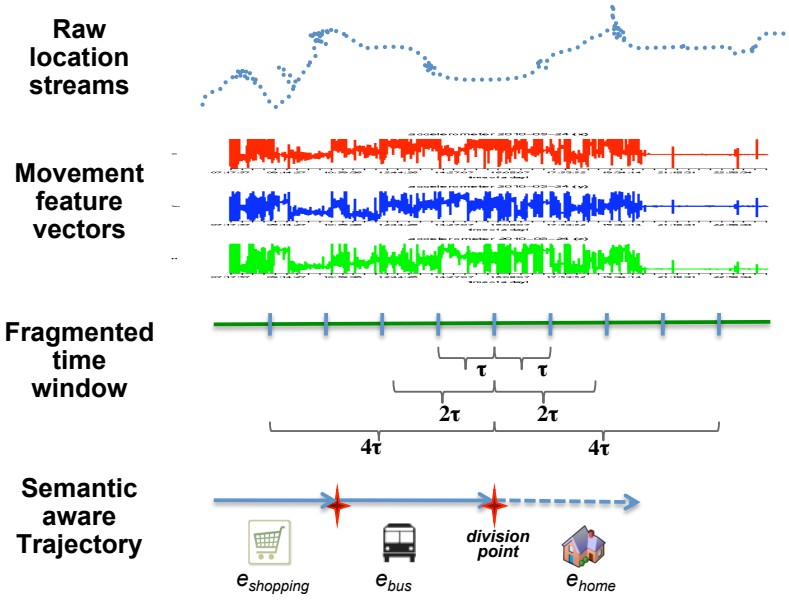


Figure 6.3: Data and semantic information model

application's sampling frequency. Small τ values enable fine-tuned episode extend determination with the make-weight of increased processing costs, while larger τ values reduce the processing load by increasing the granules that are assigned to episodes.

6.2.3 SeTraStream Overview

Having present the primitive concepts, we design an online computing framework, called "SeTraStream" (*Semantic-aware Trajectory Construction over Streaming Movement Data*), as shown in Figure 6.4. Upon the receipt of a batch containing the status updates including $\langle Q^{\ell s}, Q^{cf} \rangle$ vectors at different timestamps in τ, a cleaning and smoothing technique is applied on it (see *Step 1* on the right part of the figure). Consequently, a novel compression method (*Step 2*) is applied on the batch considering both $Q^{\ell s}$ and Q^{cf} characteristics while performing the load shedding. Finally, *Step 3* extracts the cleaned and compressed $\langle Q^{\ell s}, Q^{cf} \rangle$ feature vectors, a corresponding matrix is formed and the batch is buffered until it is processed at the SeTraStream's segmentation stage (see the left part of Figure 6.4). During the segmentation stage , a previously buffered batch is dequeued and compared with other batches' feature matrices in O_i's window. SeTraStream seeks both for short and long term changes in O_i's movement pattern, and identifies an episode whenever feature matrices are found to be dissimilar based on the RV-Coefficient (to be defined later) and a specified division threshold σ.

Towards the objective of real-time semantic trajectory construction, the core contributions of the SeTraStream framework are:

- *Online Trajectory Preprocessing.* As a *priori* step for constructing semantic trajectories, we significantly redesign trajectory data preprocessing in the real-time context, including *online cleaning* and *online compression*. Our cleaning includes an one-loop procedure

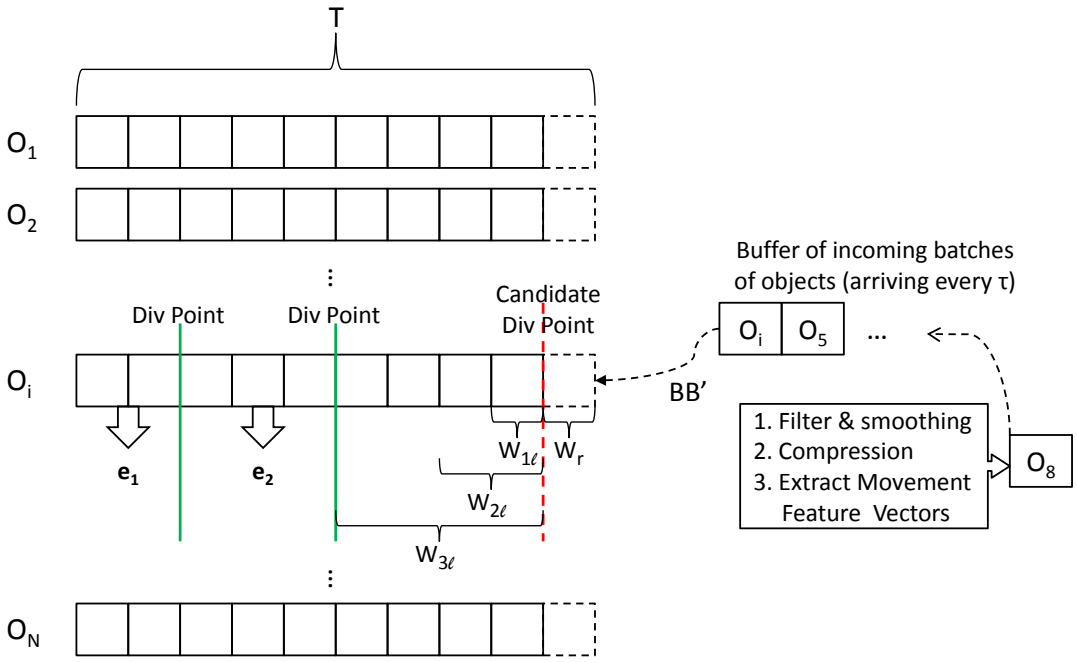


Figure 6.4: Online trajectory computing framework

for removing outliers and alleviating errors based on a *Kernel smoothing* method. SeTraStream's compression scheme uses a combination of the *Synchronized Euclidean Distance* (*sed*) and the novel definition of a *Synchronized Correlation Coefficient* (*scc*).

- *Online Trajectory Segmentation.* We design techniques for finding division points which infer trajectory episodes during online trajectory segmentation. This is the tilted time window based method in terms of the comparison on movement patterns between two adjacent windows by using the RV-coefficients.

- *Online semantic tagging* – Based on the result of online trajectory computation, we can further assign semantic tag to the episode, indicating the transportation mode that the moving object has taken during the episode, e.g., "bike", "walking", "jogging", "car"/"driving", "dwelling for shopping" etc.

- *Implementation Platform & Evaluation.* We implement SeTraStream's multi-layer procedure for semantic trajectory construction and evaluate it, considering different real-life vehicle and people positioning trajectory datasets. The results demonstrate the ability of SeTraStream to accurately provide computed semantic-aware trajectories in real-time, readily available for applications' querying purposes.

6.3 Online Data Preparation

As already described, arriving batches involving monitored objects contain their raw location stream, as well as complementary feature vectors. In this section we discuss the initial steps of data preparation before proceeding to episode determination (i.e., trajectory segmentation).

The task regards three steps depicted in the right part of Figure 6.4: (1) an *online cleaning* step that deals with noisy tuples, (2) an *online compression* stage that manages to reduce both the available memory usage and the processing cost in computing trajectory, and (3) extracting movement feature vectors, including both the location stream features and complementary features. Note beforehand that data filtering and compression take place on a par with one another as soon as a data point in a newly received batch is examined. Table 6.1 summarizes the symbology utilized in the current and the upcoming sections as well. Note beforehand that data cleaning and compression take place on a par with one another as soon as a data point in a newly received batch is examined.

Symbol	Description
N	Number of monitored objects
T, τ	Window size and batch interval
d	Number of movement features
O_i	The i-th monitored object id
B_i	The i-th batch from a candidate div. point
$Q^{\ell s}$	Tuple including $\langle x, y, t \rangle$ triplet of an objects' raw location stream
$Q^{\ell f}$	Feature vector derived from the raw location stream at t
Q^{cf}	Complementary feature vector at timestamp t
$\delta_{outlier}, \delta_{smooth}, \sigma$	Filtering, smoothing and segmentation thresholds respectively
res	The residual between the smoothed and the true value
sed, scc	Synchronous Euclidean Distance and Correlation Coefficient
W_ℓ, W_r	A left and right workpiece respectively
e_i	The i-th episode in an objects window

Table 6.1: Notations of symbols in SeTraStream

6.3.1 Online Cleaning

Traditional spatio-temporal and trajectory databases hold a common assumption that the positioning of moving objects can be precisely provided. Most efficient trajectory querying and indexing techniques are built upon this assumption. The real-life movement streams however, are far more imprecise than expected, usually unreliable and incorrect, as already discussed in Section 4.3. The reason lies on the limitations of positioning techniques (e.g., GPS measurement and sampling errors, indoor signal loss, smartphone battery runs out) or privacy concerns (e.g., people do not want to disclose their precise or private locations). Therefore, trajectory cleaning cannot be overlooked when reconstructing meaningful (semantic) trajectories from the raw locomotion data.

The main focus of trajectory data cleaning is to remove GPS errors. Jun et al. [JGO06] summarize two types of GPS errors: *systematic errors* (i.e., the totally different GPS positioning from the actual location which is caused by low number of satellites in view, Horizontal Dilution Of Position HDOP etc.) and *random errors* (i.e., the small errors up to ± 15 meters which can be caused by the satellite orbit, clock or receiver issues). These systematic errors are also named "outliers", where researchers usually design *filtering* methods to remove them;

whilst random errors are small distortions from the true values and their influences can be decreased by *smoothing* methods. Many offline GPS data cleaning works can be found such as [JGO06][SA09b][YPSC10], as already discussed in Section 4.3

In the context of streaming data, *online filtering* and *online smoothing* of streaming tuples has become a hot topic [DKV+09][GKD+10][KD08][KVD+07]. Different from the focus of prior works on data accuracy and distribution estimation, our primary concern of cleaning such streaming movement data is refining the data points that have substantial distortion of movement features for computing semantic trajectories. In current study on online data cleaning, we only apply the location features $Q^{\ell f}$ not the complementary movement features Q^{cf}.

To gain an efficient data cleaning in real-time context, we need to combine *online filtering* and *online smoothing* in a single loop. When a new batch B regarding object O_i arrives, we do the following cleaning steps:

(1) Build a kernel based smoothing model, which is similar to the offline cleaning:

$$(\widehat{x}, \widehat{y}) = \frac{\sum_i k(t_i)(x_{t_i}, y_{t_i})}{\sum_i k(t_i)} \tag{6.1}$$

where $k(t)$ is a function with the property $\int_0^{|B|} k(t)dt = 1$. The kernel function describes the weight distribution, with most of the weight in the area near the point. In our experiments, similar to [SA09b], we apply the Gaussian kernel $k(t_i) = e^{-\frac{(t_i - t)^2}{2ß^2}}$, where ß refers to the bandwidth of the kernel.

(2) Calculate the residual between the model prediction $\langle \widehat{x}, \widehat{y} \rangle$ and the true value $\langle x, y \rangle$ of the examined point $Q_p^{\ell s}$, i.e., $res = \sqrt{(\widehat{x} - x)^2 + (\widehat{y} - y)^2}$.

(3) By using a speed limit v_{limit} and the speed $v_{Q_{p-1}^{\ell s}}$ at the previous point $Q_{p-1}^{\ell s}$, respectively compute the outlier bound $\delta_{outlier}$ and the smooth bound δ_{smooth}[2]:

$$\begin{aligned} \delta_{outlier} &= v_{limit} \times (t_{Q_p^{\ell s}} - t_{Q_{p-1}^{\ell s}}) \\ \delta_{smooth} &= v_{Q_{p-1}^{\ell s}} \times (t_{Q_p^{\ell s}} - t_{Q_{p-1}^{\ell s}}) \times 120\% \end{aligned} \tag{6.2}$$

(4) Filter out the point if the residual is more than the outlier bound, i.e., $res > \delta_{outlier}$, or replace the location of the point $\langle x, y \rangle$ with the smoothed value $\langle \widehat{x}, \widehat{y} \rangle$ if the residual is between the outlier bound and the smooth bound, i.e., $\delta_{smooth} < res < \delta_{outlier}$. Otherwise, we keep the original $\langle x, y \rangle$ of the point.

This cleaning method has the advantages of the distance based outlier removal as well as the local-weighted kernel smoothing method with linear memory requirements of $O(|B|)$, where $|B|$ is the size of a batch.

[2]Here, we increase the smooth bound by using the 20% of the location prediction provided by the speed of the previous point.

6.3.2 Online Compression

A primary concern when operating in a streaming setting regards the load shedding with respect to incoming tuples. In the context of semantic trajectory computation, this happens both for limiting the available buffer usage as well as to reduce the processing cost [CMC06, KPT09, MdB04, PPS06]. In our approach, as both Definition 6.2 and Definition 6.3 imply, the approximation quality of the mere spatio-temporal trajectories is not our only concern. Semantic trajectories will be extracted based on additional features other than those derived from spatio-temporal $\langle x, y, t\rangle$ points. On the other hand, if we ignore the spatio-temporal trajectory approximation quality, the portion of the movement features that rely on the pure location stream will later be uncontrollably distorted. To cope with the previous requirements, we propose a method and define a *significance score* suitable to serve our purposes.

Assume that a batch regarding object O_i is processed (step. 2 at right part of Figure6.4) and $(Q^{\ell s}_{p-1}, Q^{\ell s}_{p})$ is the last examined pair of points in it. When a new point $Q^{\ell s}_{p+1}$ is inspected, we first obtain the significance of $Q^{\ell s}_{p}$ from a spatio-temporal viewpoint by fostering the *Synchronous Euclidean Distance* (SED) that is applied in [MdB04][PPS06]. The calculation of such SED metrics is as follows:

$$
\begin{aligned}
sed(Q^{\ell s}_{p}, Q^{\ell s}_{p-1}, Q^{\ell s}_{p+1}) &= \sqrt{(x_{Q'^{\ell s}_{p}} - x_{Q^{\ell s}_{p}})^2 + (y_{Q'^{\ell s}_{p}} - y_{Q^{\ell s}_{p}})^2} \qquad (6.3)\\
where,\ the\ projection\ x_{Q'^{\ell s}_{p}} &= x_{Q^{\ell s}_{p-1}} + v^{x}_{Q^{\ell s}_{p-1}Q^{\ell s}_{p+1}} \cdot (t_{Q^{\ell s}_{p}} - t_{Q^{\ell s}_{p-1}})\\
y_{Q'^{\ell s}_{p}} &= y_{Q^{\ell s}_{p-1}} + v^{y}_{Q^{\ell s}_{p-1}Q^{\ell s}_{p+1}} \cdot (t_{Q^{\ell s}_{p}} - t_{Q^{\ell s}_{p-1}})\\
where,\ the\ velocity\ vector:\ v^{x}_{Q^{\ell s}_{p-1}Q^{\ell s}_{p+1}} &= \frac{x_{Q^{\ell s}_{p+1}} - x_{Q^{\ell s}_{p-1}}}{t_{Q^{\ell s}_{p+1}} - t_{Q^{\ell s}_{p-1}}}\\
v^{y}_{Q^{\ell s}_{p-1}Q^{\ell s}_{p+1}} &= \frac{y_{Q^{\ell s}_{p+1}} - y_{Q^{\ell s}_{p-1}}}{t_{Q^{\ell s}_{p+1}} - t_{Q^{\ell s}_{p-1}}}
\end{aligned}
$$

The above measure is also employed in the sampling based approach of [PPS06]. Nevertheless, *sed* constitutes an absolute number that lacks the ability to quantify the particular significance of a point with respect to other spatio-temporal points within the current batch. In order to appropriately derive the aforementioned significance quantification, in SeTraStream's compression scheme, we normalize the *sed* measurement and define the relative spatio-temporal significance Sig^{SP} of a point as:

$$
Sig^{SP}(Q^{\ell s}_{p}) = \frac{sed(Q^{\ell s}_{p}, Q^{\ell s}_{p-1}, Q^{\ell s}_{p+1})}{max_{sed}} \qquad (6.4)
$$

with $0 \leq Sig^{SP}(Q^{\ell s}_{p}) \leq 1$. The denominator max_{sed} denotes the current maximum *sed* of points in the batch. Obviously, increased $Sig^{SP}(Q^{\ell s}_{p})$ estimations represent points of higher spatio-temporal significance.

Carefully inspecting *sed*'s formula, we can conceive that the intuition behind its definition is to measure the amount of distortion that can be caused by pruning the spatio-temporal point $Q^{\ell s}_{p}$. That is, having omitted $Q^{\ell s}_{p}$ we could virtually infer the respective data point at the

timestamp $t_{Q_p^{\ell s}}$ using the proceeding and succeeding ones $(Q_{p-1}^{\ell s}, Q_{p+1}^{\ell s})$. And calculating $Q_p'^{\ell s}$, $sed(Q_p^{\ell s}, Q_{p-1}^{\ell s}, Q_{p+1}^{\ell s})$ measures the incorporated distortion.

Thus, as regards the complementary feature vectors of O_i we choose to base the measure of their significance on the *Correlation Coefficient* (*corr*) metric. First, fostering an attitude similar to that in *sed*'s calculation as explained in the previous paragraph, we estimate the value at the i-th position of vector $Q_p'^{cf}$ as:

$$[Q_p'^{cf}]_i = [Q_{p-1}^{cf}]_i + \frac{[Q_{p+1}^{cf}]_i - [Q_{p-1}^{cf}]_i}{t_{Q_{p+1}^{cf}} - t_{Q_{p-1}^{cf}}}(t_{Q_p^{cf}} - t_{Q_{p-1}^{cf}}) \tag{6.5}$$

Then, based on *corr* in statistics we define the *Synchronized Correlation Coefficient* (*scc*) between $(Q_p'^{cf}, Q_p^{cf})$ of complementary feature vectors:

$$scc(Q_p'^{cf}, Q_p^{cf}) = \frac{E(Q_p'^{cf}Q_p^{cf}) - E(Q_p'^{cf})E(Q_p^{cf})}{\sqrt{(E((Q_p'^{cf})^2) - E^2(Q_p'^{cf}))(E((Q_p^{cf})^2) - E^2(Q_p^{cf}))}} \tag{6.6}$$

where $E()$ refers to the mean and $-1 \leq scc(Q_p'^{cf}, Q_p^{cf}) \leq 1$.

The choice of *scc* is motivated by the fact that its stem, *corr*, possesses the ability to indicate the similarity of the trends that are profound in the examined vectors rather than relying on their absolute values [DKV+09][GKD+10][KVD+07]. Hence, it provides an appropriate way to identify (dis)similar patterns in the complementary vectors and can be generalized in order to detect similar patterns between movement feature vectors in their entirety. Values of *scc* that are close to -1 exhibit high dissimilarity between $(Q_p'^{cf}, Q_p^{cf})$, indicating that omitting Q_p^{cf} results in higher pattern distortion. Calculating "$1 - scc$" enables higher measurements to account for more dissimilar patterns and taking one step further, min-max normalization on $1 - scc$ allows (dis)similarity values lie within $[0, 1]$. Thus, we eventually compute the relative significance of the complementary feature vector as:

$$Sig^C(Q_p^{cf}) = \frac{1 - scc(Q_p'^{cf}, Q_p^{cf})}{2max\{(1 - scc)\}} \tag{6.7}$$

In the context of our compression scheme, the more dissimilar $(Q_p'^{cf}, Q_p^{cf})$ are, the higher the probability to be included in the window should be. As a result, the overall significance $Sig(Q_p)$ of Q_p can be estimated by the combination of both the location stream feature $Sig^{SP}(Q_p^{\ell s})$ and the complementary feature $Sig^C(Q_p^{cf})$. The weight balance between them is application dependent, though we choose to treat them equally important [LHW07]:

$$Sig(Q_p) = \frac{1}{2}(Sig^{SP}(Q_p^{\ell s}) + Sig^C(Q_p^{cf})) \tag{6.8}$$

Eventually, for a threshold $0 \leq Sig_{thres} \leq 1$, Q_p remains in the batch when $Sig(Q_p) \geq Sig_{thres}$, or it is removed for compression purposes otherwise.

6.4 Online Trajectory Segmentation

Similar to the importance of trajectory segmentation in offline computing, the online trajectory segmentation is the fundamental stage in the online trajectory computing. This stage comes right after the initial data preparation utilizing the resulted feature vectors that are chosen to remain in a newly received batch after data cleaning and compression (see Figure 6.4).

6.4.1 Sliding Window Method

Upon deciding the data points of a batch that are to be included in the window as devised in the previous subsection, SeTraStream proceeds by examining episode existence in T. To start with, we assume the simple case of the current window consisting of a couple of τ-sized batches (i.e., $T = 2\tau$). We will henceforth refer to each part of the window composed of a number of compressed batches as *workpiece*. Intuitively, distinguishing episodes is equivalent to finding a *division point*, where the movement feature vectors on its left and right sides are uncorrelated and thus correspond to different movement patterns. In our simple scenario, a *candidate division point* is placed in the middle of the available workpieces.

Hence, we subsequently need to dictate a suitable measure in order to determine movement pattern change existence. We already noted the particular utility of the correlation coefficient on the discovery of trends [DKV$^+$09][GKD$^+$10][KVD$^+$07], and thus (in our context) patterns in the movement data. In this processing phase movement feature vectors composing each workpiece essentially form a pair of matrices for which correlation computation needs to be conducted. As a result, we will reside to the *RV-coefficient* which constitutes a generalization of the correlation coefficient for matrix data. We organize left window features (W_ℓ) into a $d \times m$ matrix, where d is the number of movement features and m represents a number of vectors (at different timestamps) that are the columns of the matrix . Similarly, right window features (W_r) are organized in a $d \times n$ matrix i.e., n columns exist. The *RV-Coefficient* between $\langle W_\ell, W_r \rangle$ is defined as:

$$RV(W_\ell, W_r) = \frac{Tr(W_\ell W'_\ell W_r W'_r)}{\sqrt{Tr([W_\ell W'_\ell]^2) Tr([W_r W'_r]^2)}} \tag{6.9}$$

where W'_ℓ, W'_r refer to the transpose matrices, $Tr()$ denotes the trace of a matrix and $0 \leq RV \leq 1$. RV values closer to zero are indicative of uncorrelated movement patterns. Based on a division point threshold σ workpieces W_ℓ, W_r can be assigned to a pair of different episodes $e_\ell = (0, T - \tau, geometry_{bound})$, $e_r = (T - \tau + 1, T, geometry_{bound})$ when:

$$RV(W_\ell, W_r) \leq \sigma \tag{6.10}$$

or to a single episode $e = (0, T, geometry_{bound})$ otherwise.

Now, consider the general case of T covering an arbitrary number of batches. It can easily be conceived that in a larger time window an alteration in the movement pattern may happen:

- instantly as a sharp change, or
- in a more smooth manner as time passes

As a result, upon the arrival of a new workpiece W_r, we initially check for short-term changes in the patterns of movement. We thus place a candidate division point between the newly received workpiece and the last of the existing ones. Then the correlation between the movement feature vectors present in $\langle W_{1\ell}, W_r \rangle$ is computed. Notice that $W_{1\ell}$ this time possesses an additional subscript which denotes the step of the procedure, as will be shortly explained. Similarly to our discussion in the previous paragraphs, when $RV_1(W_{1\ell}, W_r)$ is lower than the specified division threshold, a division point exists and signals the end of the previous episode e_ℓ and starts a new one e_r.

No short-term change existence triggers our algorithm to proceed by seeking long-term discorrelations. For this purpose, we first examine $RV_2(W_{2\ell}, W_r)$ doubling the time scale of the left workpiece by going 2τ units back in the window from the candidate division point. In case RV_2 does not satisfy Inequality 6.10, this procedure continuous by *exponentially expanding* the time scale of the left workpiece in a way such that at the i-th step of the algorithm the size of $W_{i\ell}$ is $2^{(i-1)}\tau$ units and $RV_i(W_{i\ell}, W_r)$ is calculated. When Inequality 6.10 is satisfied the candidate division point is a true division point which bounds the previous episode $e_i = (time_{from}, time_{to}, geometry_{bound})$ and constitutes the onset of a new. Otherwise, W_r is rendered the current bound of the last episode by being appended to it. If no long-term change is detected, the aforementioned expansion ceases when either the beginning of the last episode or the start of T (in case all previous batches have been attributed to the same episode) is reached, i.e., no data points of the penultimate episode are considered since its extend has already been determined.

The exponential workpiece expansion fostered here is inspired by the *tilted time window* definition [GHP+03] as a general and rational way to seek movement pattern changes in different time granularities. Other expansion choices can also be applied. All of these options are orthogonal to our approaches and do not affect the generic function of SeTraStream. Our approach manages to effectively handle sliding windows as a slide of τ time units results in:

(1) the expiration of the initial batch of the first episode e_{first} of O_i which affects its ($time_{from}$, $geometry_{bound}$) attributes;

(2) the appendage of a newly received batch that either extends the last episode e_{last} (when no division point is detected) or starts a new episode. The outcome of the online segmentation consists of tuples $\mathfrak{T}_{O_i} = \{e_{first}, \ldots, e_{last}\}$ representing objects semantic trajectories.

6.4.2 Time and Space Complexity

The introduced trajectory segmentation procedure, premises that a newly appended batch will be compared with left workpieces that may be (depending on whether a division point is detected) exponentially expanded until either the previous episode end or the start of the window is reached. Based on this observation, the lemma below elaborates on the complexity of the checks required during candidate division point examination.

Lemma 1. The time complexity of SeTraStream's online segmentation procedure, for N monitored objects, under exponential $W_{i\ell}$ expansion is $O(Nlog_2(\frac{T}{\tau}))$ per candidate division point.

Proof. For a single monitored object, the current window is composed of $\frac{T}{\tau}-1$ batches (excluding the one belonging to W_r). The worst case scenario appears when no previous episode exists in the window and the candidate division point is not proven to be an actual division point. By considering the exponential workpiece expansion, comparisons (i.e., σ checks) may reach a number of $k = min\{i \in \mathbb{N}^* : \frac{\frac{T}{\tau}-1}{2^{(i-1)}} \geq 1\}$ at most. Adopting logarithms on the previous expression and summing for N objects completes the proof. □

Now, recalling the definition of the *RV-Coefficient* measure, it can easily be observed that its computation relies on the multiplication of the bipartite matrices with their transpose. Assume that the number of d-dimensional movement feature vectors in a cleaned and compressed batch are n. Based on the above observation we can see that instead of maintaining the original form of the vectors which requires $O(d \cdot n)$ memory space, we can reduce the space requirements during episode determination by computing the product of the $d \times n$ matrix of the batch with its transpose. This reduces the space requirements to $O(d^2)$ per batch since in practice $d \ll n$. So, to check a short-term change in the movement patterns we do not need to store the full matrices of $W_{1\ell}, W_r$ which in this case are composed of one batch each, but only the matrix products as described above.

However, this point may not be of particular utility since left workpieces are expanded during the long-term pattern alteration checks. A natural question that arises regards whether or not the product $W_{i\ell}W'_{i\ell}$ can be expressed by means of the multiplication of single batch matrices, with their transposes.

Lemma 2. $W_{i\ell}W'_{i\ell}$ is the sum of batch matrix products with their transposes: $W_{i\ell}W'_{i\ell} = \sum_{j=1}^{2^{(i-1)}} B_jB'_j$, where B_j is used to notate the matrix formed by the vectors in the j-th batch (from a candidate division point to the end of $W_{i\ell}$).

Proof. Let $W_{i\ell} = [B_1|B_2|\cdots|B_{2^{(i-1)}}]$ the matrix of the (i-th) left workpiece during the current division point check. B_js are used to denote sub-matrices belonging to individual batches that were appended to the workpiece. It is easy to see that the transpose matrix can be produced by transposing these submatrices: $W'_{i\ell} = [B'_1|B'_2|\cdots|B'_{2^{(i-1)}}]$. And then $W_{i\ell}W'_{i\ell}$ can be decomposed into $B_jB'_j$ products: $W_{i\ell}W'_{i\ell} = B_1B'_1 + B_2B'_2 + \cdots + B_{2^{(i-1)}}B'_{2^{(i-1)}} = \sum_{j=1}^{2^{(i-1)}} B_jB'_j$. □

Thus, for each batch we only need to store a square $d \times d$ matrix[3], which determines the space complexity of online segmentation leading to Lemma 3.

Lemma 3. During the online episode determination stage of SeTraStream, the memory requirements per object O_i are $O(d^2\frac{T}{\tau})$ and assuming N objects are being monitored the total space utilization is $O(d^2N\frac{T}{\tau})$.

[3]We also keep the geometry bound of the batch that is utilized in the final episode geometry bound determination as well as some additional aggregate statistics, of minor storage cost, for classification and tag assignment in the next step.

6.5 Trajectory Episode Tagging

After the step of online segmentation that divides the movement data into separate trajectory episodes, we need to further identify the semantic tagging information for each episode to better understand the real-time streaming movement data. Similar to the trajectory annotation in Chapter 5 for achieving the complete semantic-aware trajectories, but in a real-time context. Such real-time semantic tagging is a very challenging topic. In this thesis, we also address this problem and provide an early study towards this direction.

Having detected an episode e_i, SeTraStream manages to specify in an online fashion the triplet $(time_{from}, time_{to}, geometry_{bound})$ describing its spatio-temporal extend. Based on such information, the online tagging algorithm needs to infer the episode *tag*, as the final piece of information associated with an episode in a complete semantic movement in Definition 6.3. Given application's context, possible *tag* instances form a set of movement pattern classes and notice that the instances of the classes are predetermined for the applications we consider, e.g., the annotation of stops with activities like *work in office*, *relax at home*, the annotation of moves with transportation modes like *walking, cycling, jogging, on bus*, as discussed in Chapter 5.

The problem of episode tag assignment can be formulated to a classification task, where the classifier can be trained in advance based on the collected episodes and the detected episode e_i can be timely classified based on the trained model and the episode features. The procedure of such online tagging can be briefly described as follows:

- *Offline training* – Based on some tagged mobility data, we train a model offline with well-known techniques like decision trees, boosting, SVM, neural or Bayesian networks [FPSSU96]. Firstly, the episode features are computed like segment *distance*, *duration*, *density*, *avg. speed*, *avg. acceleration*, *avg. heading* etc. In the offline annotation in Section 5.4.2, we have discussed the three types of episode features, i.e. the GPS-based features, network features and POI features. In online case, the algorithm relies on the location features and complementary features.

- *Online testing* – After detecting an episode, we compute the features (similar to the offline feature calculation during the training stage), and test the features in the offline trained model to infer the possible tags.

6.6 Experiment

In this section, we present our experimental results and performance analysis in computing semantic trajectories from streaming movement data.

6.6.1 Experimental Setup

We utilize two different datasets: *Taxi Data* - this dataset includes taxi trajectory data for 5 months with more than 3M GPS records, which do not have any complementary features. We mainly use taxi data to validate compression. It is non-trivial to get real-life on-hand dataset with both complementary features and the underlying segment ground-truth tags. Therefore, we

collect our own trajectory data by developing Python S60 scripts deployed in a Nokia N95 smartphone, which can generate both GPS data and accelerometer data from the embedded sensors. We calculate GPS features (e.g., *transformed longitude, latitude, speed, direction*) as the location stream vectors ($Q^{\ell f}$) and accelerometer features (e.g., *mean, variance, magnitude, covariance of the 3 accelerometer axis*) as the complementary feature vectors (Q^{cf}). We term the latter dataset as *Phone Data* within which, we also provide our own real segment tags (e.g., *standing, jogging, walking*) to validate the online segmentation accuracy. For *Phone Data*, we also work on the GPS data from the data campaign organized by Nokia Research Center - Lausanne, which has collected 185 users' phone data with about 7M records in total [KBD+10][YCP+11].

6.6.2 Online Data Preparation

As described previously, our online data cleaning needs to consider two types of GPS data errors, i.e., filtering outliers as systematic errors and smoothing the random errors. The experimental cleaning results are shown in Figure 6.5: (a) sketches the original trajectory data; (b) identifies the outliers during the online cleaning process; (c) and (d) present the original movement sequences together with the final smoothed trajectories, where (c) includes the outliers in the original sequences, whilst (d) removes them for better visualization.

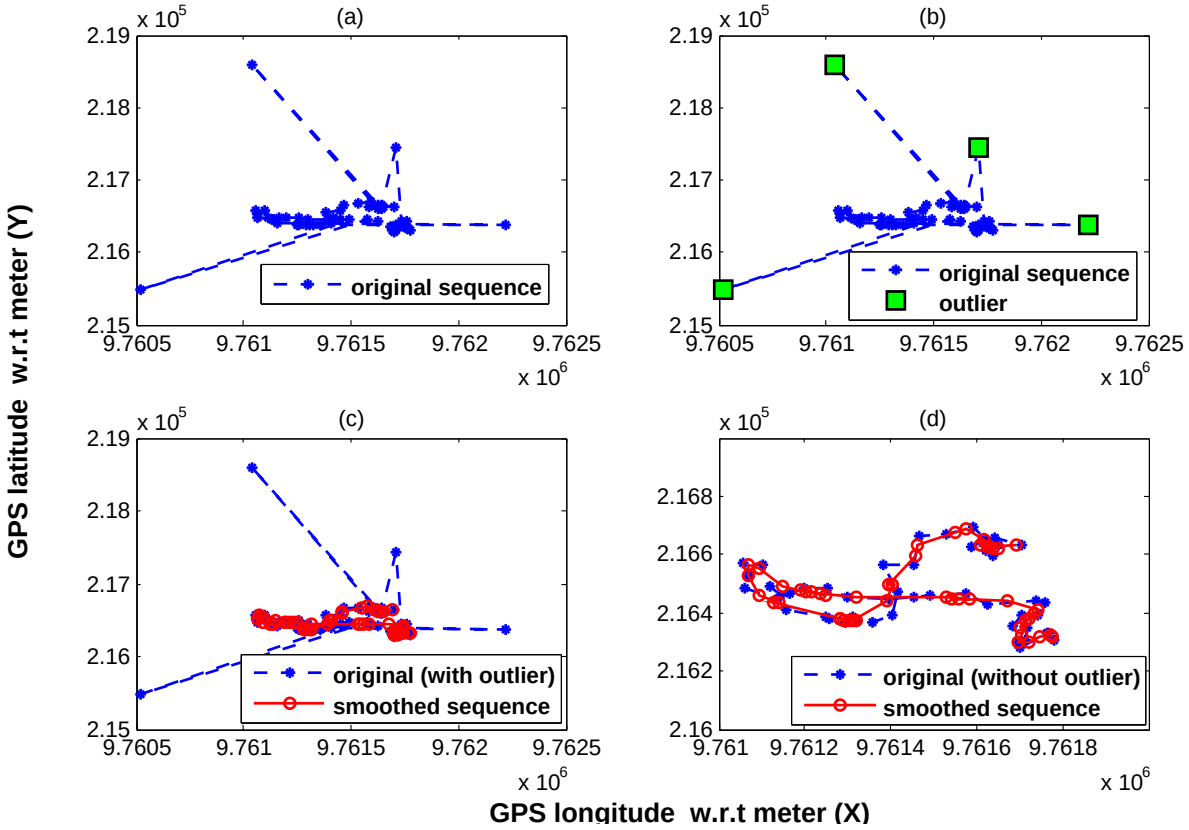


Figure 6.5: Online data cleaning (outlier removal and smoothing)

Technically, compression makes sense when dealing with large data sets, however both *Taxi Data* and the big part of *Phone Data* have no complementary features (Q_p^{cf}) available but only the GPS features ($Q_p^{\ell s}$). Thus, our current experiment validates the sensitivity of data compression

rate with respect to the spatio-temporal significance $Sig^{SP}(Q_p^{\ell s})$ on location streams, without considering the significance of the complementary features $Sig^{C}(Q_p^{cf})$. As shown in Figure 6.6, we plot the compression rate sensitivity when applying different thresholds on $Sig^{SP}(Q_p^{\ell s})$. The results are proportional when using the *Phone Data* with respective $Sig(Q_p)$ thresholds.

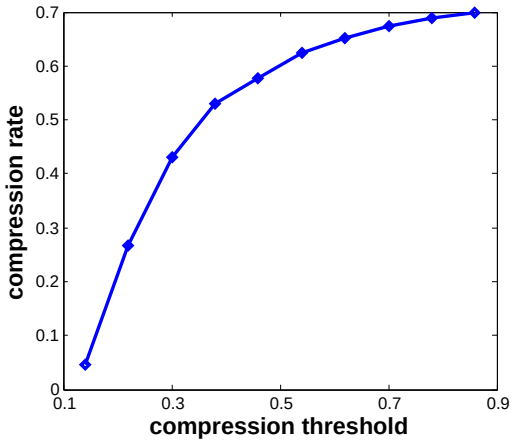


Figure 6.6: Data compression rate w.r.t. different thresholds

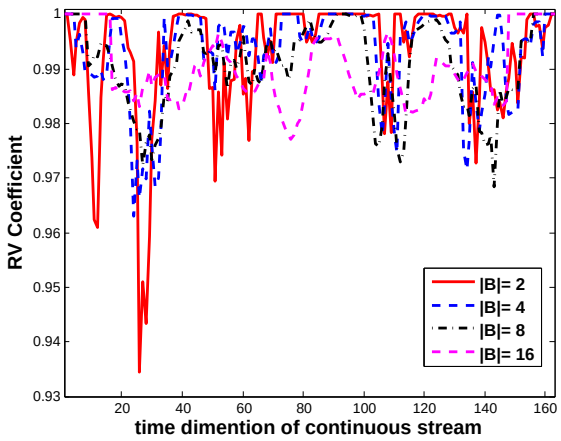


Figure 6.7: $RV(B_l, B_r)$ in the two neighboring batches (without div_{thres})

6.6.3 Online Segmentation

SeTraStream's procedure in online trajectory segmentation relates to (1) initially computing the RV-coefficient between two workpieces $RV(W_\ell, W_r)$ and (2) expanding W_ℓ if $RV(W_\ell, W_r)$ is bigger than the given threshold σ (otherwise, we identify a division point between two episodes). Results are shown in Figure 6.8, where for $T = 60s$ we can discover two main division points (with RV-coefficient< 0.6 and batch size $\tau < 16$), which is consistent with the underlying ground-truth tags. The stars in the figures are the real division points in the streaming data, which indicate when user changes their movement behaviors e.g., from jogging to walking and finally to standing.

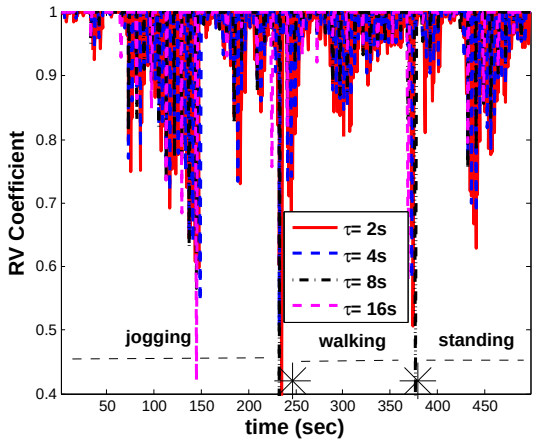


Figure 6.8: Episode identification varying batch size, for $\sigma = 0.6$

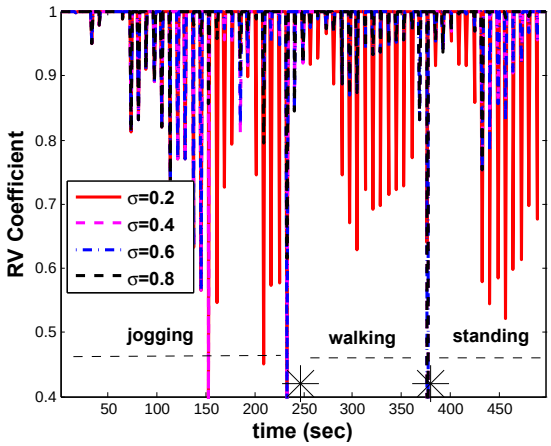


Figure 6.9: Sensitivity of RV w.r.t. different σ at $\tau = 8s$

Figure 6.8 analyzes the sensitivity of using different batch sizes, where the best outcome (i.e accurate episode extend determination) is $\tau = 8s$; when $\tau = 16s$, we actually identify three division points, which is partially correct, since as we can see there are only two real division points in the stream. Similarly, we also investigate the segmentation sensitivity regarding to different division thresholds σ in Figure 6.9. The best segmentation result is achieved when $\sigma = 0.6$.

Finally, we evaluate the time performance of SeTraStream's trajectory segmentation module. We measure the segmentation latency with 25 users in the *Phone Data*. In the experiments, we used a laptop with 2.2 Ghz CPU and 4 Gb of memory. From Figure6.10 and Figure 6.11, we can see the segmentation time is almost linear, in both situations with different batch sizes (τ) and different division thresholds (σ), which is quite consistent with Lemma 1.

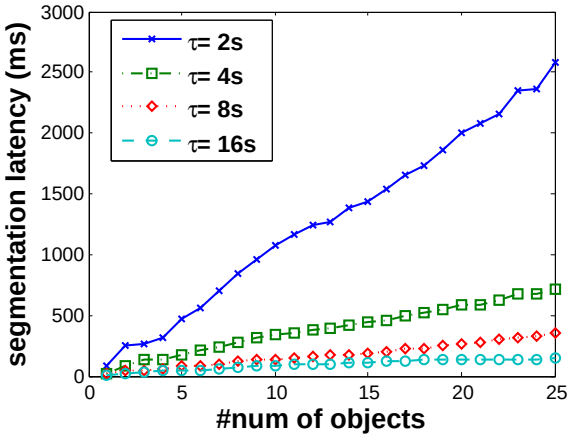


Figure 6.10: Segmentation latency with different τ sizes (σ=0.6)

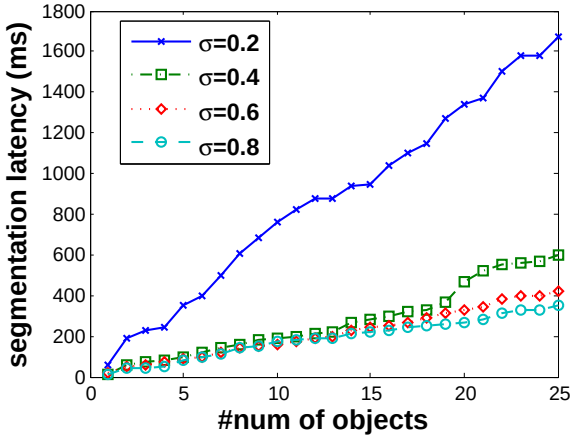


Figure 6.11: Segmentation latency with different σ thresholds ($\tau = 8s$)

6.7 Summary

This chapter presented the fourth major contribution of this thesis, i.e., the *online trajectory computing*. The main outcome is an online framework (called "SeTraStream") that enables semantic trajectory construction over streaming movement data. To the best of our knowledge, this is the first method proposed in the literature dealing with this problem in real-time streaming environments. Moreover, we considered challenges occurring in real-world applications including data cleaning (with filtering and smoothing methods) and load shedding procedures (with online compression) before accurately identifying trajectory episodes in objects' streaming movement data. The initial result of such online computing framework is published in [YGK+11].

In SeTraStream, the online cleaning with combined filtering and smoothing in one loop is capable of efficiently preprocessing streaming movement data. The online segmentation applies the *tilted* window based methods for identifying episode division points using the RV-coefficients. SeTraStream additionally exploits online tagging functionality to achieve complete semantic-aware trajectories, in terms of offline training and online testing. The future focus of this real-time study is on further investigating different kinds of window techniques for online segmentation, as well as on online (or incremental) training of semantic tagging.

Chapter 7

Semantic Trajectory from Multi-Sensors

A smartphone is a mobile phone that offers more advanced computing ability and connectivity than a contemporary feature phone.

Andrew Nusca, 2009

7.1 Introduction

In the previous chapters, both online and offline approaches of computing trajectories are dedicated to GPS-alike mobility positioning data, focusing on data feed from one single GPS sensor. As a complementary study, this chapter discusses the benefits and challenges of computing semantic trajectories by using multiple sensory data streams from smartphones, which establishes the fifth major contribution of this thesis, i.e., *semantic trajectories from multiple sensors in smartphones*. In particular, the focus is on the combination of GPS and accelerometer sensors, to extract meaningful semantics from this temporally co-related streams of two independent sensors. For example, we want to infer the semantic indoor activities such as cooking & dining (at home) and working & having lunch (at the workplace). Towards such goal, this thesis builds a two-tier framework: the first tier is to identify the micro-activities (denoted as MA, e.g., sitting, standing, walking) from accelerometer data; the second tier uses the GPS location data (e.g., home and office) together with the sequences of micro-activities to infer the high-level semantic indoor activities (notated as HA, e.g., cooking at home, working in office).

This chapter is organized as follows: Section 7.2 presents the challenges of inferring semantic activities via multiple sensors in smartphones and our two-tier approach; Section 7.3 addresses the techniques for inferring the micro-activities as locomotion from accelerometer data; Section 7.4 applies the locations using GPS data and infers the high-level semantic activities; Section 7.5 shows some experimental studies; and finally Section 7.6 summarizes this chapter.

7.2 Learning from Multi-Sensors

There is a lot of studies on mobility behavior analysis of moving objects, including this thesis about offline and online trajectory computing. Such research has traditionally focused on moving objects such as vehicles, ships etc. With the advent of smartphone techniques, the focus has shifted to "people sensing". People-sensing [CEL+08] is an active area of research today. The primary objective is to use the sensors on the phone and sense different aspects of the environment, effectively bringing people in the sensing loop. Driven from the GPS-alike vehicle and people trajectory study, this thesis further investigates semantic knowledge discovery of people trajectories from multi-sensor data feeds coming from smartphones.

Phones are one of the most ubiquitous sensors worn by people today. People trajectories created by phones are quite different in nature than vehicles, ships and other moving objects, in terms of (1) *interest* - they do not enjoy long GPS exposure; (2) *dynamic* - people perform several activities using their phones while carrying it around; (3) *multiple* - Phones contain multiple sensors and data (GPS, accelerometer, microphone, call records, gyroscope). Multi-sensor (GPS + accelerometer, sound, nearby devices) feeds from phones contain rich information about a trajectory (activity being performed, neighbors, acoustic characteristics of environment). This increases richness of data inferred through such knowledge discovery and provides better understanding of people movement patterns. In this chapter, we consider GPS and accelerometers, found in most high-end phones today. Other sensors (e.g., microphones) are not the focus of this thesis work, but for future research directions.

7.2.1 Activities from GPS+Accelerometer

GPS and accelerometers individually capture a partially overlapping set of semantics. GPS traces have been studied widely to capture object's movements in areas, capture GIS resources (roads, parks) where object passes. Analysis of GPS traces have focused on pattern mining [GNPP07, HLGL08] and extraction of semantic knowledge [SPD+08, YPSC10, YCP+11, YGK+11]. Accelerometers on the other hand have been used typically for detecting activities (Activities of Daily Life ADLs [RDML05, BGC09] – walk, stand, sit, jump, run, cook etc.). GPS traces have been used to infer outdoor movements like walking, jogging as well. However, GPS alone is not enough to have a *complete* representation of a user's movements and activities for the whole day since a large part of people trajectories are in indoor environments. Moreover, GPS alone does not have enough discriminatory capability, e.g., distinguishing between "cooking at home" and "sitting at home" as GPS signal is missing at home. Accelerometers can provide vital cue towards precise detection of the activity.

In our study, we record both GPS and accelerometer data of subjects continuously. Thereafter, our goal is to analyze this data to extract meaningful semantics from this temporally co-related stream of independent sensors. At the first stage, we focus on accelerometer streams. In this regard, our goal is to characterize dominant motion patterns of accelerometer data streams from mobile phones under completely naturalistic conditions. Related work that extracts activities from accelerometers have largely focused in artificial or semi-naturalized settings, where *multiple* accelerometer sensors are placed on different body parts (elbow, back etc.) and motion patterns are classified based on training data. Real-life people trajectories cannot impose such

restrictions on placement of the phone. This thesis additionally investigates activity recognition considering such completely naturalized settings. In the second stage, our goal is to infer activities being performed by people, thereby building a richer, spatio-semantic model of user activities over a span of a day. We do this using a mutual reinforcement learning strategy using cues from GPS data. The complete process is unsupervised and requires no manual intervention. This model helps us achieve a wider cover of a user's activities, covering both indoor and outdoor environments. Therefore, the objective is to infer the semantic people trajectory. To better describe such semantic people trajectory, we define different levels of activities, by using accelerometer, GPS or both of them, as shown in Figure 7.1.

- **Micro Activity (MA)**: Low-level semantic activities such as *jogging*, *walking*, *standing*, *sitting*, *cycling*, *driving*.
- **Location Inference (GPS)**: different locations that can be inferred by GPS, e.g., *home*, *office*, *shopping-center*, *transportation*.
- **High-level semantic Activity (HA)**: High-level semantic activities such as *home*, *office*, *shopping*, *transportation*, *sport*.

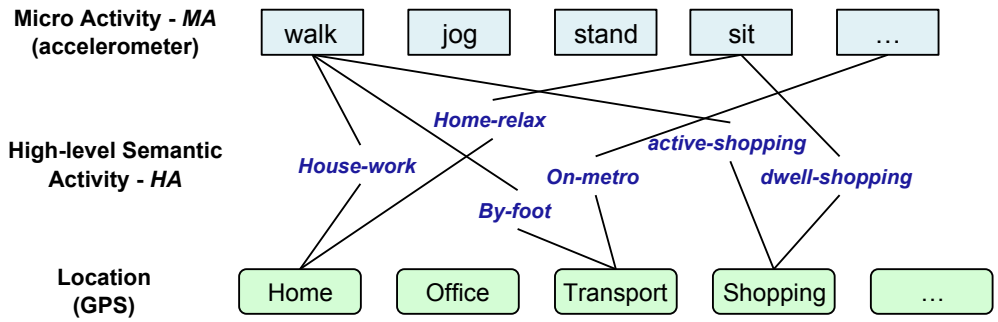


Figure 7.1: Different levels of activities

7.2.2 Two-Tier Computing Framework

We explore the use of mobile phone-generated sensor feeds to determine the higher-level (i.e., at the semantic level), indoor, lifestyle activities of individuals. In particular, we investigate whether human locomotive behavior, derived from phone-embedded accelerometer streams collected under naturalistic settings, can be used to identify activities such as cooking & dining (at home) and working & having lunch (at the workplace). We propose and evaluate a two-tier activity extraction framework (called SAMMPLE[1]) where features of the low-level sensor data are first used to identify individual locomotive micro-activities (e.g., sitting or standing), with the micro-activity sequence subsequently being used to identify the discriminatory characteristics of individual higher-level activities. At the first-tier, we use empirical data to establish the set of features that provide most accurate classification under naturalistic conditions, when the on-body position and orientation of the phone change unpredictably. At the second-tier, we

[1] Semantic Activity Mining via Mobile Phone-based Locomotive Estimation

explore feature extraction techniques utilizing the duration and sequence of micro-activities of an individual to identify higher-level activities.

Figure 7.2 illustrates the hierarchy of steps associated with our semantic activity extraction approach; roughly speaking, the hierarchy enables the association of different natural timescales with different 'levels' of activities.

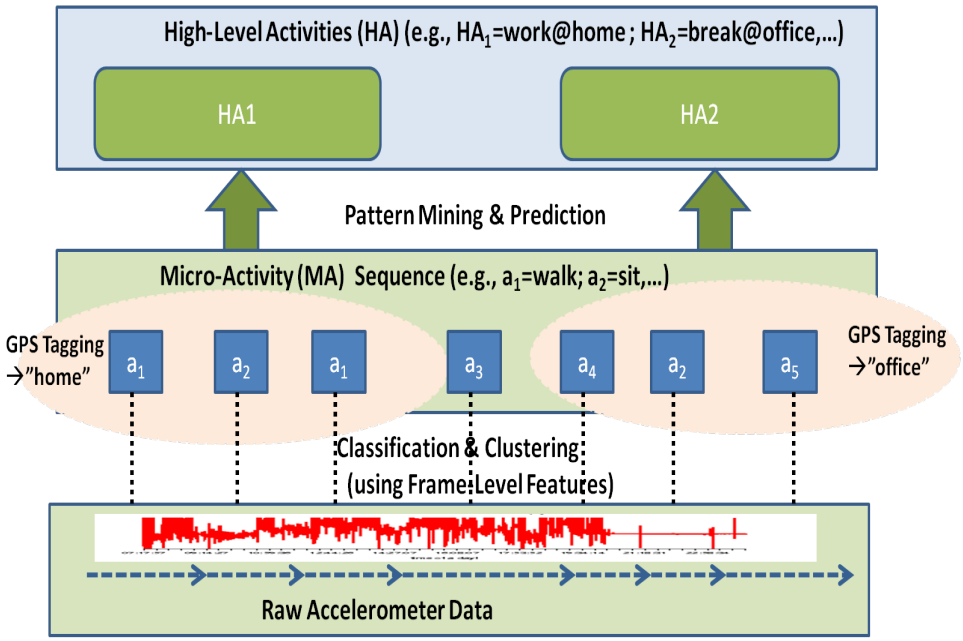


Figure 7.2: Our Two-Tier Semantic Activity Inferencing Framework

At the lowest-level, we collect the "3-axis" accelerometer stream from the phone, and group the continuous stream of accelerometer readings into a sequence of non-overlapping *"frames"*; typically, each frame duration is relatively small (e.g., 30 seconds, 1 minute) and is used to identify a specific locomotive state of the individual. From each frame, we extract appropriate classification *features* associated with the frame, and then employ a classification algorithm to map each frame into a corresponding micro-activity. This stream of micro-activities is then subjected to a *segmentation* process, which transforms the stream of frame-based micro-activities into a sequence of, potentially longer-duration (e.g., lasting a few minutes), micro-activity *"episodes"*. This segmentation step is necessary for naturalized settings, and its primary goal is to filter out the transient micro-activities captured in individual frames to establish a more meaningful set of stable episodes. For example, an individual might take 5 minutes to walk from her desk to the cafeteria. While this should constitute a single "walk" episode, a micro-scopic frame-level examination of the accelerometer will reveal that the episode constituted of many 'walk frames', plus perhaps a few 'stand' frames (e.g., when the individual slowed down to accommodate other pedestrian traffic) and even a few 'unknown' frames (e.g., when the individual shifted the phone from her trouser to chest pocket). Given the essentially random nature of such individual frames, the segmentation step helps eliminate statistical noise while preserving the key or defining micro-activity associated with that time segment.

The second tier of our hierarchy then involves the transformation of the sequence of "micro-activity segments" into even longer-duration semantic activities (e.g., having lunch at work). As

explained previously, GPS observations are used to distinguish between the individual's semantic locations (i.e., in the office or at home), which in turn helps identifying the range of semantic activities to which the sequence of micro-activity segments can be mapped (e.g., 'cooking' is a valid activity at home, but not in the office). Given the obvious day-to-day fluctuations in the micro-activities associated with a semantic activity, we will utilize a *statistical pattern mining* framework to transform the sequence of micro-activity segments into semantic activities. Note the key difference between the two transformations: the feature-based classification (of sensor frames to individual micro-activities) at the lower layer is memoryless, whereas the pattern-based classification at the upper layer is based on mapping a *temporal sequence* of micro-activities to the most-likely semantic activity. Our central hypothesis thus is that each semantic activity can be associated with a set of *distinct sequences of micro-activity segments.*

Two aspects of this framework are worth highlighting:

(1) Conceptually, the feature-based mapping of individual accelerometer frames to micro-activites may be viewed as a form of scalar quantization, while the mapping of micro-activity sequences to a semantic activity is analogous to a process of vector quantization. We emphasize that our principal innovations lie in the micro-to-semantic activity transformation, and (for the most part) use previously developed approaches for mapping accelerometer feature-vectors to micro-activities. What is perhaps not so obvious is that the higher-layer mapping process differs from classical coding theory in that the mapping does not work on a fixed length of symbols, i.e., the duration of an individual semantic activity is not fixed. Accordingly, our algorithmic approach must be capable of identifying (1) a semantic activity that has statistical fluctuation in duration across different days, and (2) the time instants where there is a change in semantic activities.

(2) The adoption of the two-step hierarchy has both foundational and computational benefits. A reader may indeed ask: why can we not simply look for characteristic temporal patterns of semantic activities over the frame-level accelerometer feature vectors? As explained previously, such an approach would fail to be robust, as microscopic frame-level fluctuations in activities would make it much harder to find clear mappings from low-to-high level activities. Additionally, the approach significantly reduces the computation burden. Imagine semantic activities, lasting roughly about 1 hour, defined over sequences of 30-sec long frames, with each frame associated with a 4-dimensional feature vector. A direct mapping process effectively looks to find characteristic sequences in a ($\frac{60mins}{30sec}$ =) 120 dimensional vector, consisting of 4-dimensional elements. By instead first resolving the low-level frames into, say, 5 minute-long micro-activity segments, we transform the problem into finding characteristic sequences of ($\frac{60mins}{5min}$ =) 12 dimensional vectors, with 1-dimensional elements, as a much easier computational problem.

7.2.3 Challenges and Contributions

There are several challenges in mining spatio-semantic trajectories by using multiple sensors in smartphones. We first describe some challenges from the GPS sensory data in smartphones.

- **GPS Stream discontinuity**: Mobility data (like GPS) has "gaps", as people often enter buildings (office, house) where signals are limited; sampling frequency also plays an important role in how much context can be inferred.

- **Several levels of inferencing**: Most people are usually engaged in multiple and interleaved activities (e.g., browsing the net on the phone while on home, talking while walking etc). Activities can be at different levels: micro-scopic (sit, stand, walk, run, jump), semantic level (e.g., going home, going to office, in a party, lunch at office, lunch with friends on a weekend etc). How can we detect such diverse activities using only the mobile phone as the sensors?

- **Limited Ground truth data**: Supervised learning strategies with training data are often impractical for activity classification and contextual analysis of such spatio-temporal feeds, due to large variation in the patterns. Classification accuracy is primarily guided by how many of the "known" patterns are learnt in the training data.

In addition, a practically useful implementation of our framework must, however, address several challenges that arise from the use of accelerometer-generated data streams under naturalistic settings:

- **Stream Discontinuity & Irrelevance**: We must cope with the fact that the accelerometer data will be both discontinuous (which will not be available when the phone is turned off) and non-representative of the person's true activity state (e.g., when the smartphone is left behind on the office desk during lunch).

- **Noise & Usage Variation in Naturalized Settings:** Accelerometer data is known to be noisy, and can generate different signatures under changing external environments (e.g., an individual walking on a moving train vs. walking on a sidewalk). Moreover, the readings will exhibit distinctive variations depending on the continual change in the placement and orientation of the phone–e.g., an individual can place the phone in different locations (e.g., in their trouser pocket, laptop bag or in their hand).

- **Activity Concurrence:** One of the biggest challenges arises from the fact that humans, in their daily lives, often *multi-task* at both the micro-activity (e.g., walking while talking on the phone) & the semantic-activity (e.g., cooking and watching TV) levels. Most current algorithmic approaches are unable to distinguish between activity patterns that are not temporally disjoint.

- **Personalization**: As users vary in both their micro-activity signatures (e.g., how they walk) and in their semantic activity patterns (e.g., how long they take to cook), the inferencing framework must be personalized to the specific characteristics of each individual.

In this chapter, we will adopt a personalized data mining framework that addresses all of the above challenges, except that of *activity concurrence* – i.e., our inferencing algorithms implicitly assume that an individual will engage in only one semantic activity at any instant. Our key research question is as follows: *Does the accelerometer-based locomotive history of an*

individual contain enough ***predictability*** *(to help establish the typical locomotive profile of a specific activity) and* ***discriminatory power*** *(to help distinguish between multiple alternatives) to allow us to deduce an individual's higher-level lifestyle activities, such as cooking or watching TV? If so, how do we develop a data analytics framework that uncovers such discriminatory locomotion-based features, and what level of accuracy can such an analytics framework hope to achieve?* To address this question with relevant research and technical challenges, this chapter uses real-life observational traces to make the following contributions to smartphone-based activity inferencing:

(1) We present a two-step process of semantic activity inferencing that first uses raw accelerometer streams to derive a sequence of an individual's micro-activities, and then employs statistical feature extraction & mining on the micro-activities to establish the mapping from such micro-activities to the likely semantic activity[2]. This two-step approach helps provide robustness against noise in the underlying sensor observations & accommodates daily behavioral variations in semantic activities.

(2) We explicitly consider the problem of accelerometer-based micro-activity classification in a *naturalistic environment*, where the individual goes about her daily life, and show that a combination of orientation-independent & orientation-sensitive features provides the best classification accuracy (up to 90% in our studies) for such naturalistic settings.

(3) We propose and evaluate two discriminatory feature extraction techniques (utilizing the micro-activity stream) to identify specific semantic activities. The first approach uses features based only on the total *duration* of underlying micro-activities (resulting in a classification accuracy of $\sim 70\%$), while the second approach additionally considers the *order* (or sequence) of these micro-activities (providing an additional $\sim 10\%$ increase in classification accuracy).

7.2.4 Preliminaries and Definitions

Before providing the detailed techniques in the two-tier framework, we first present the necessary mathematical definitions needed to describe the steps in the rest of this chapter.

Definition 7.1 (Accelerometer Data Stream - $\mathcal{A}$)**.** Initially, a sequence of data points recording acceleration along 3 axes, i.e., $\mathcal{A} = \langle A_1, A_2, \ldots, A_n \rangle$, where $A_i = (x_i, y_i, z_i, t_i)$ is a tuple with accelerations (x_i, y_i, z_i) and timestamp t_i.

Definition 7.2 (Micro-Activities- $\mathcal{MA}$)**.** A set of M distinct micro-activities $\mathcal{MA} = \{MA_1, \ldots, MA_M\}$, where each element of $\mathcal{MA}$ corresponds to a pre-defined locomotive state of the individual. For this chapter, we consider a set of 7 ($M = 7$) micro-activities: {'sit', 'sit relax', 'walk', 'slow walk', 'bursty move', 'stand', 'using stairs'}. While most MA elements are self-descriptive, the non-obvious ones are described in Table 7.1.

[2] "Semantic Activity" and "high-level activity" are used synonymously in this chapter

Table 7.1: Descriptions of some non-obvious micro activity labels

Name of MA Label	Exemplary Description of Activity
sitRelax	all types of relaxed sitting (e.g., shaking legs, stretching, ..)
slowWalk	walk at a slow pace, walk inside office rooms
burstyMove	jerky movements (e.g., get up from chair, movements inside kitchen), varies across users

Definition 7.3 (High-Level Activities- $\mathcal{HA}$)**.** A set of H distinct semantic activities $\mathcal{HA} = \{HA_1, \ldots, HA_H\}$, where each element of $\mathcal{HA}$ corresponds to a pre-defined semantic activity of the individual. For this chapter, we consider a semantic hierarchy of High-level activities obtained from the users we survey. Details are in Figure 7.3.

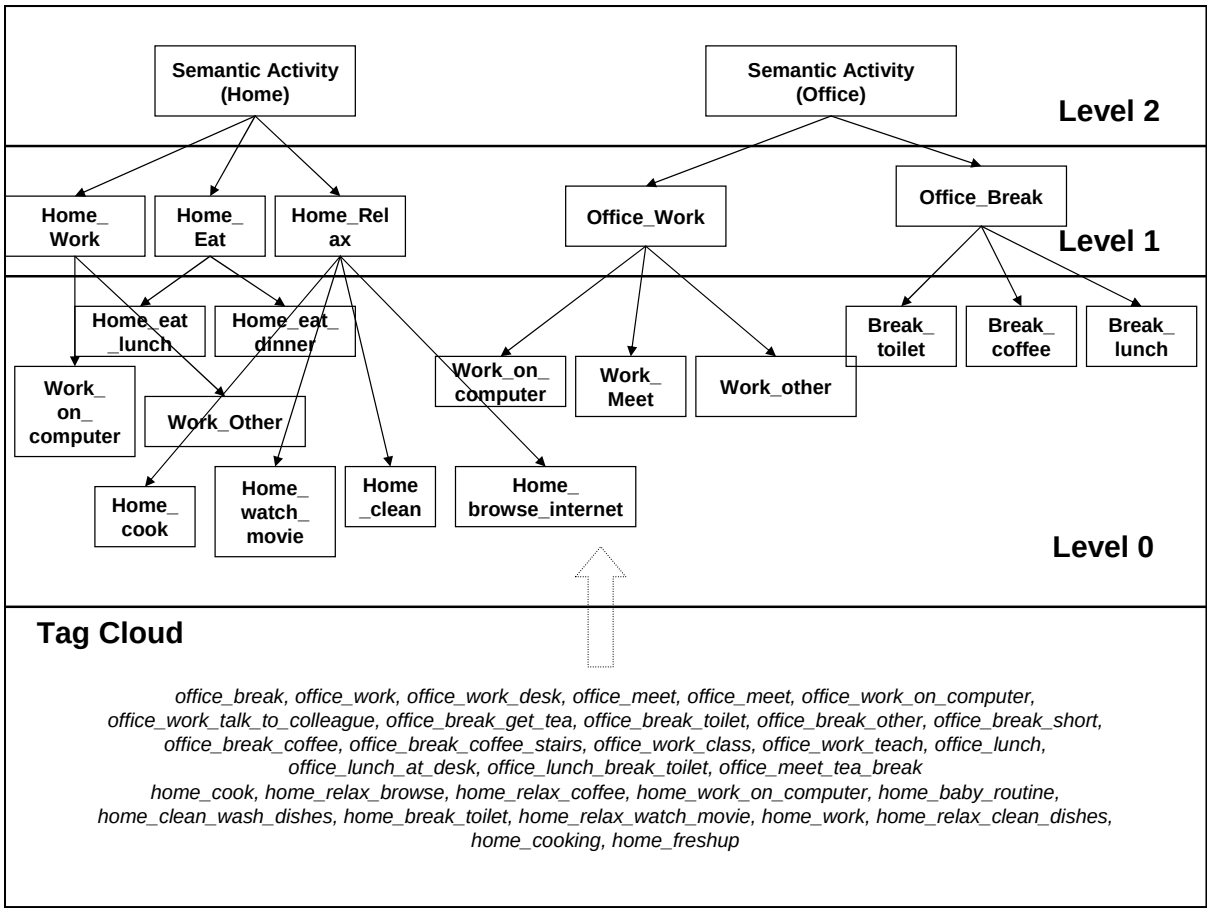


Figure 7.3: User tag cloud and hierarchy of HA activity classes

7.3 Micro-Activity Inference from Accelerometer

Activity detection under completely naturalized settings naturally complicates the stream produced - due to presence of multiple interleaved activities, no restriction of placement, movement etc. Our first goal is to understand how good supervised pattern learning and classification algorithms perform when we look at such streams, considering a limited vocabulary of well-known micro activities that have been tried and tested in semi-naturalized settings. We consider seven micro activities defined in Table 7.1, focusing on people's daily life, especially indoor activities. This is the first part of our two-level activity inferencing approach that requires us to transform the accelerometer data stream into a *MicroActivity Stream* (MS), defined as follows.

Problem 1 (MicroActivity Stream - $\mathcal{MS}$)**.** A transformation of $\mathcal{A}$ such that $\mathcal{MS} = \langle MS_1, MS_2, \ldots, MS_m \rangle$, where MS_i is a tuple (m_i, t_i^b, t_i^e) such that $m_i \in \mathcal{M}$, and t_i^b & t_i^e are timestamps denoting the start and end times of MS_i. Note that we implicitly assume a non-overlapping MicroActivity Stream–i.e., $t_{i+1}^b \geq t_i^e, \forall i$.

7.3.1 Frame based MA Inference

We now detail our approach of using frame-based classification to transform a raw accelerometer data stream $\mathcal{A}$ into a resulting MicroActivity Stream ($\mathcal{MS}$), under naturalistic conditions. We start with an unlabeled accelerometer stream $\mathcal{A}$ as input and first split it into non-overlapping equal-sized *frames* (see Figure 7.4 and Definition 7.4), denoted by $\mathcal{F}$, of duration T_f, such that $\mathcal{F} = \langle F_1, F_2, \ldots F_n \rangle$. We then use an appropriate vector of features (computed over the elements of $\mathcal{A}$ belonging to the i^{th} frame F_i) to classify each F_i, i.e., to associate a value $m_i \in \mathcal{MA}$ with each frame. This classification process results in a MicroActivity Stream such that the i^{th} element of the sequence, MS_i, is defined by the micro-activity value m_i and start & end times, (t_i^b, t_i^e) (with $t_i^e - t_i^b = T_f$), associated with F_i.

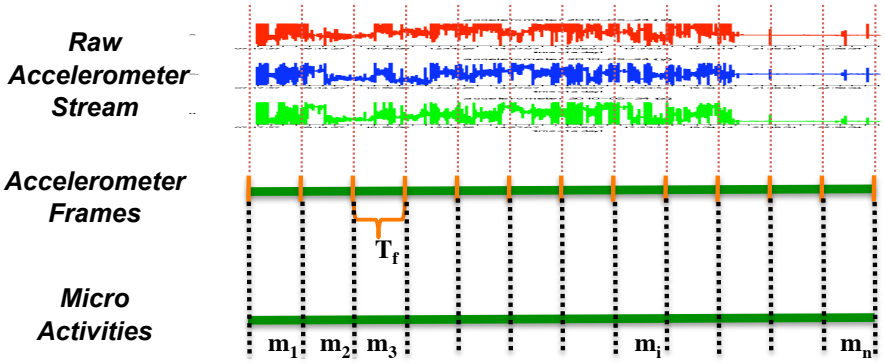


Figure 7.4: Frame-based MA inference

Definition 7.4 (Accelerometer Frame - F_i)**.** A group of continuous accelerometer records, i.e., $F_i = \{a_{i+1}, \ldots, a_{i+k}\}$ in time duration T_f, where $a_j = \langle x, y, z \rangle$ is the three dimensional accelerometer streaming data points.

Feature Calculation – Most prior work in using accelerometer-based *features* for micro-activity classification (e.g., [RDML05, CBC+08]) implicitly assumes that the (phone-embedded)

accelerometer is either placed on a unique on-body position (e.g., in a hip holster) and/or always maintains the same orientation (relative to the ground). Under naturalistic conditions, an individual may, however, not only place the phone in different on-body positions at different times, but its orientation might get modified (due to movements/jerks, varying pocket sizes, usage style). Recent micro-activity recognition literature (e.g., [LYL+10]) studying different types of naturalistic settings stresses the importance of making the right feature choices for such settings. Accordingly, we investigate the important question: *what features, or combination of features, are better suited to activity identification under such naturalistic settings, and what level of accuracy can the resulting classification techniques achieve?*

Broadly speaking, the set of potential features used for accelerometer based micro-activity detection can be grouped into two types:

- **Orientation-Dependent Features:** This set of features is computed independently and distinctly for each of the three axes, x, y, z. For example, one may compute the mean values, represented by $\bar{x_i}, \bar{y_i}, \bar{z_i}$ of the accelerometer samples in frame F_i, independently for each of the 3 accelerometer axes, thereby resulting in 3 independent features. In general, we can expect orientation-dependent features to be useful when the orientation of the phone (relative to the earth's horizontal) remains constant.

- **Orientation-Independent Features:** This set of features is not associated independently with a specific axis of the accelerometer. Instead, it is either an orientation-independent combination of all 3 axes (e.g., the mean or variance of the acceleration magnitude($\sqrt{x^2 + y^2 + z^2}$)) or specifically associated with acceleration values *derived relative to a reference frame, e.g., the ground.* In the latter case, we first need to figure out the phone's current orientation and transform the original x_i, y_i, z_i values to corresponding values along the reference frame (e.g, horizontal or vertical to the ground). In general, orientation-independent features are expected to be more robust to random variation's in a phone's orientation or on-body location, but may offer lesser resolution than their orientation-dependent counterparts.

Most past work in this area has used either *exclusively* orientation dependent or orientation-insensitive features (e.g., [LYL+10]) and looks to distinguish well-separable micro-activities (e.g., sit, walk, jog, cycle). In contrast, we explored various *combinations* of these two feature classes, and evaluated their ability to classify our set of locomotive activities, which have varying degrees of similarity to one other (e.g., 'walk'and 'slowWalk').

In particular, we consider a feature matrix, consisting of both time and frequency domain features from: (1) the 3D axis of the phone (orientation dependent) (refered to as 3D-features), (2) A projection of the readings on the gravity direction and plane perpendicular to gravity, which makes it orientation-independent (refered to as 2D-features). The mean of accelerometer readings along each axis over a long period of time can produce a good estimate of the gravity direction [LYL+10] and we use this to project the signal to this "2D" reference frame. A total of $\sim$ 50 features were used per frame (F_i); Table 7.2 lists the most-important features. For the frequency domain, the features are computed by first transforming the (x_i, y_i, z_i) segment into a

Table 7.2: Key features used in our micro-activity experiments

	name	definition	orientation-independent
Feature Components	calibrated 3 axis data (3D)	(x_i, y_i, z_i)	no
	projected 2D (Vertical) $[\vec{p}]$	$p = \frac{d \cdot v}{v \cdot v} \cdot v$, where $v = \langle \bar{x}, \bar{y}, \bar{z} \rangle$ (the mean of x,y,z) and d $= \langle x - \bar{x}, y - \bar{y}, z - \bar{z} \rangle$	yes
	projected 2D (Horizontal) $[\vec{h}]$	$d - p$	yes
	projected 2D (magnitude) $[mag]$	$\lvert\vec{h}\rvert, \lvert\vec{p}\rvert, corr(\lvert\vec{h}\rvert, \lvert\vec{p}\rvert)$	yes
Time Domain Features	Mean	AVG($\sum x_i$); AVG($\sum y_i$); AVG($\sum z_i$)	no
	Variance	VAR($\sum x_i$); VAR($\sum y_i$); VAR($\sum z_i$)	no
	Mean-Magnitude	AVG($\sqrt{x_i^2 + y_i^2 + z_i^2}$)	yes
	Magnitude-Mean	$\sqrt{\bar{x}^2 + \bar{y}^2 + \bar{z}^2}$)	yes
	Two-Axis Correlation	$corr(xy) = \frac{cov(xy)}{\sigma_x, \sigma_y}$	no
	Signal-Magnitude Area (SMA)	$\frac{1}{n} \sum_{i=1}^{n} (\lvert x_i\rvert + \lvert y_i\rvert + \lvert z_i\rvert)$	yes
Frequency Domain Features	FFT Magnitude	$m_j^{(x)} = \lvert a_j + b_j i\rvert$; similarly, $m_j^{(y)}$, $m_j^{(z)}$	no
	FFT Energy	$\frac{\sum_{j=1}^{N} (m_j^2)}{N}$, for x,y,z respectively	no
	FFT Entropy	$-\frac{\sum_{j=1}^{n} (p * log(p))}{n}$, for x,y,z respectively, where p is normalized histogram count of FFT component magnitudes	no

250-point FFT vector [RDML05]. A state-of-the-art calibration technique tested on the Nokia N95 [YLJ10] was used to calibrate the sensor readings before further processing.

Classification Algorithms – Orthogonal to the choice of the classifying features, we also experimented with a wide variety of state-of-the-art clssifiers, including *decision tree*, *Adaptive Boost* (Adaboost), *SVM*, *bayesian network* and *Naive Bayes*. We should emphasize that our principal goal is not to devise any new feature or classification algorithm, but empirically investigate the performance of such features choices & classification algorithms in naturalized environments. We observed that, while the performance of different classifiers was largely similar, the *LibSVM*[3] classifier provided the best accuracy overall (for both MA and HA classification); accordingly, most of the performance graphs are plotted using the *LibSVM* classifier.

7.3.2 Description of the Empirical User Study

To empirically understand the utility of various features in classifying our $M = 7$ micro-activities under naturalistic settings, we recruited a total of 5 university participants for our experimentation and data collection purposes. Each subject was provided a Nokia N95 phone (4 used it as their primary cellphone for the duration of our study), with embedded Python scripts that sampled the accelerometer and GPS sensors periodically and uploaded the sampled traces back to our centralized data collection server each day. The 3-axis accelerometer values were sampled with a frequency of 30 Hz, while the GPS sensor sampling was guided by an internal state machine [KBD$^+$10], that performs optimized sampling (e.g., sample only when moving and under coverage) to reduce energy overhead.

As requiring each subject to continually record their micro-activity 'ground truth' (e.g., log every change from walk to sit or stand) for extended durations is unacceptably onerous, we performed an in-depth experimental data collection with $User1$; for the other users, we collected locomotion & postural data on only their preferred body positions (where they carried their phone regularly). Each user was asked to perform each of the $M = 7$ MAs consecutively for $\sim 7-10$ minutes each, resulting in an overall study duration of $\sim 50-60$ minutes. In contrast to most constrained-usage studies, $User1$ performed each MA while changing the on-body position of the phone among three pre-defined positions: {*Shirt Pocket, Pant Front Pocket, Pant Back Pocket*}, as well as randomly altering the phone's *orientation* (for each on-body position). Such a change in on-body location allows us to empirically study the accuracy of the feature-based classification approach in naturalized environments, where the exact on-body position (& orientation) of the phone will be unknown and subject to random changes.

We first study the MA classification accuracy achieved by different combinations of the various accelerometer-based features for the representative MA data of $User1$. Results are presented using a *10-fold cross validation approach*, where 90% of the collected data was taken as the *training* set, and the resulting classifier logic was applied to the remaining 10% *test* data; the figures represent the average over 10-such training vs. test partitions of the data.

Figure 7.5 plots the *classification accuracy* achieved on $User1$ for the data collected with *the orientation of the phone* (*within each on-body position*) *fixed*: the plotted data corresponds

[3] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

to a frame duration of $T_f = 5$ secs. Results are plotted (using the *LibSVM* classifier) for two different cases:

(a) when the on-body position of the phone is assumed to be known *a-priori*: in this case, the classification is performed, and the classification accuracy is evaluated, separately for each of the three on-body positions;

(b) when the on-body position of the phone is an unknown–in this case, the classification is performed, and the classification accuracy is evaluated, on the *combined* data of all 3 positions.

The plot shows various combinations of "3D" & "2D" features (see Table 7.2). $3D_{all}$ (or $2D_{all}$) implies the use of all (time & frequency domain) 3D-features (or 2D-features); $2D_{hp}$ refers to features computed on the projected orientation-independent frames – both gravity ($\vec{p}$) and its plane perpendicular ($\vec{h}$) ; $2D_{mag}$ refers to features computed over magnitudes of $\vec{h}$ and $\vec{p}$; additionally, 'correlation', 'PCA' refer to the feature reduction techniques we used, respectively for applying the correlation-based feature selection [SH97] or Principal Component Analysis based feature extraction.

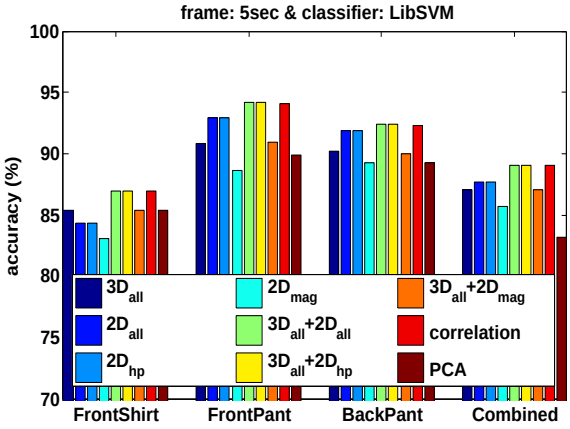


Figure 7.5: MA Classification accuracy for $User1$ with fixed (vertical) phone orientation ($T_f = 5$ secs).

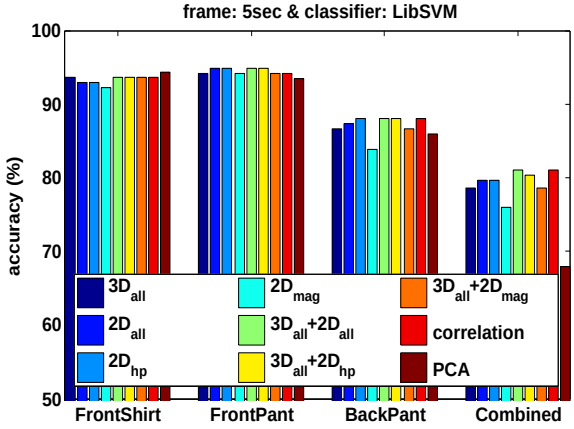


Figure 7.6: MA Classification accuracy for $User1$ with naturalistic (varying) phone orientations ($T_f = 5$ secs)

The figure illustrates three important points:

(a) the MA classification accuracy is higher for certain on-body positions of the smartphone-in particular, the classification accuracy is higher when the phone is placed in the lower segment of the body (an observation previously observed in wearable sensing [BI04, GWT+09]);

(b) the classification accuracy for the "Combined" case (which best corresponds to our target naturalistic environment) case is approx. $85 \sim 90\%$;

(c) most importantly, the highest classification accuracy is achieved when we use *a combination of both orientation-dependent (3D) and orientation-independent (2D) features*

These conclusions hold uniformly across analyses performed with other parameters–e.g., when T_f was varied to be $\{2, 5, 10, 20\}$ secs, or when the other classifiers (such as Bayesian Network, Naive Bayes, Decision Tree - J48, Adaboost) are used. We also observed that the best MA classification accuracy was achieved by T_f choices of $\{5, 10\}$ secs, and thus restrict future experimental results to these two values.

To further understand the salient characteristics of feature-based classification, Figure 7.6 plots the MA classification accuracy for both specific on-body positions & for the "Combined" case, but with *the orientation of the phone varied, even for the same on-body position, during each MA.* Compared to Figure 7.5, it should be clear that the naturalistic phenomenon of changes in orientation results in an observable drop in accuracy, with the classification accuracy for the "Combined" case now reaching only $\sim 75 - 80\%$. The figure also demonstrates that, for naturalistic environments characterized by uncontrolled changes in on-body positions & phone orientation, the use of a combination of orientation-dependent (3D) & orientation-insensitive (2D) features provides higher classification accuracy than the independent use of each feature type. Figure 7.7 plots the "confusion matrix" for the "Combined" case– the results show that the SAMMPLE approach is quite successful in correctly labeling each MicroActivity sequence $\mathcal{MS}$. We believe that our study is the first to establish the need for combining 3D and 2D features for improved MA inferencing accuracy under naturalistic conditions.

$$\mathcal{T}=\begin{pmatrix} & bm & sr & s & sw & r & t & w \\ burstyMove(bm) & 50 & 1 & 0 & 12 & 1 & 0 & 2 \\ sitRelax(sr) & 3 & 47 & 0 & 2 & 0 & 14 & 0 \\ sit(s) & 0 & 2 & 19 & 0 & 0 & 9 & 0 \\ slowWalk(sw) & 6 & 3 & 0 & 54 & 1 & 0 & 0 \\ stairs(r) & 3 & 0 & 0 & 0 & 54 & 0 & 6 \\ stand(t) & 1 & 1 & 0 & 0 & 0 & 64 & 0 \\ walk(w) & 3 & 0 & 0 & 8 & 2 & 0 & 53 \end{pmatrix}$$

Figure 7.7: Confusion matrix for MA classification of $User1$. An entry in the i^{th} row & j^{th} column denotes the number of instances where the classifier labeled the activity as MA_i, while the 'ground truth' was MA_j.

In our SAMMPLE framework, these MA inference models (learnt during the training phase) are applied on each frame of the incoming accelerometer stream $\mathcal{A}$, to give us a MicroActivity Sequence $\mathcal{MS}$, which forms the input to the subsequent HA-classification phase (that we shall discuss next).

7.3.3 Segmentation based MA Inference

In addition to the frame-based micro-activity learning, we also design a segmentation approach for learning accelerometer data stream. The raw accelerometer data stream is collected continuously in naturalized setting. There are no signs when is a new activity start and the previous activity stops; they are all collected together. Therefore, the initial task is to provide a correct data segmentation strategy, which can divide the long sequence into several segments corresponding to separate activity (e.g., 5 minutes dwelling after half hour sitting). However, the frame-based

methods omit such preprocessing step, where the experiment setting has already manually divided the accelerometer data into separate activity frames, and the task is only focusing on the activity recognition on each accelerometer frame [BI04, RDML05, BGC09, KLLK10, KWM10].

We do not segment the raw accelerometer stream directly based on each record a of the raw accelerometer sampling stream $\mathcal{A}$ (see Definition 7.1), but based on each group of accelerometer records in a time window τ, where τ can be values like one second or one minute. There are two advantages of using τ rather than each a directly: (1) *Sensitive to outliers* – The accelerometer data is much sensitive to user monitions and outliers, e.g., changing phone positions, talking with phone when walking; therefore, the neighboring record a can be varying a lot while the user is continuing with the same activity. The group values of a can deduce such sensitivity. (2) *Efficient segmentation* – The raw data stream typically is very huge, as accelerometer data sampling usually has a very high frequency (e.g., 30 records per second in our experiment). With such accelerometer grouping, we can obtain better segmentation performance.

In such segmentation-based MA inference method, we apply the same concept of *Accelerometer Frame* as in the frame-based MA inference (see Definition 7.4). Based on the sequence of accelerometer frames, we first segment the sequence of accelerometer frames F_i into several continuing accelerometer episodes $e^{(A)}$, where the frames in each episode in some sense sharing similar characteristics – corresponding to the same activity, as shown in Figure 7.8. Compared with Figure 7.4, we can apply additional segmentation techniques to identify so call "accelerometer episodes".

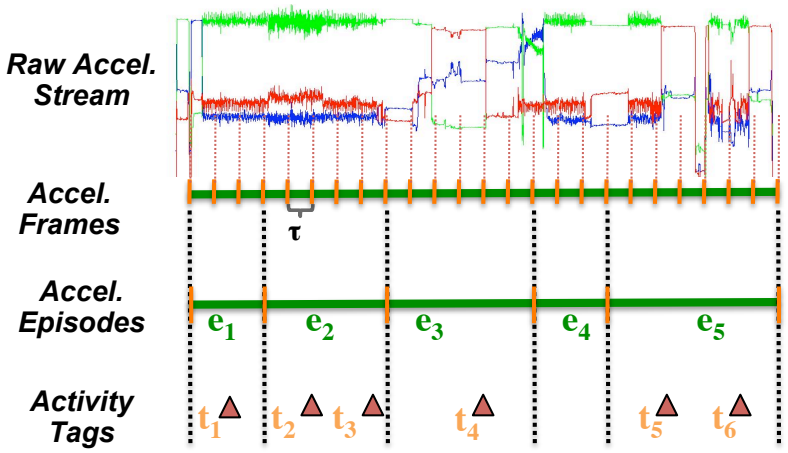


Figure 7.8: Segmentation of accelerometer stream

We extend the SWAB (*Sliding Window And Bottom-up*) online segmentation algorithm [KCHP01, KCHP04], and build a multi-dimensional time series segmentation of accelerometer frames. In our study, we focus on three dimensions and call this extended SWAB method "*surface-regression based segmentation*". The choice of SWAB is because of: (1) online segmentation which can be applied in real-time systems; (2) efficient segmentation which can be easily transplanted in the smartphone platform, as the time complexity is linear and much faster compared with any polynomial or dynamic programming based sophisticated methods.

To validate the segmentation result and tune the segmentation error bound σ in finding the dividing unit, we design the measurement by using real-life user tags. In our Nokia phone

platform, user can provide activity tag in their daily life in real time via the phone. The micro activity tag can be sitting, standing, jogging, walking, climbing etc. As shown in Figure 7.8, there are six tags. Such ground truth tags can validate the segmentation results. We design the measure based on the widely used metrics of *precision* and *recall* to measure the segmentation accuracy, as shown in Formula 7.1. Based on this definition, the optical segmentation is when the final result where each segment has one and only one tag inside, i.e., both precision and recall are 100%.

$$\begin{aligned} precision &= \frac{\#segments\ with\ tag}{\#\mathcal{E}_A} \\ recall &= \frac{\#segments\ with\ tag}{\#total\ tags} \\ f{-}measure &= \frac{2 * precision * recall}{precision + recall} \end{aligned} \tag{7.1}$$

The accelerometer segmentation example in Figure 7.8 has five segmented episodes, where four episodes have user tags inside (e_1, e_2, e_3, e_5) and two episodes have more than one tags. According to Formula 7.1, the precision is $\frac{4}{5} = 80\%$, recall is $\frac{4}{6} = 66.7\%$, and $f{-}measure$ is $\frac{2*4/6*4/5}{4/6+4/5} = 72.7\%$.

In the measurement of Formula 7.1, all the segments are considered equally, similar as all the tags. We can further define the weighted (w) precision and recall (see Formula 7.2), where the length of segment and the tag interval distance are also measured to validate the segmentation result.

$$\begin{aligned} precision^{(w)} &= \frac{\Sigma\{length\ of\ segment\ with\ tag\}}{\Sigma\{length\ of\ e_i\}} \\ recall^{(w)} &= \frac{\Sigma\{tag\ interval\ with\ division\ inside\}}{\Sigma all\ tag\ intervals} \\ f{-}measure^{(w)} &= \frac{2 * precision^{(w)} * recall^{(w)}}{precision^{(w)} + recall^{(w)}} \end{aligned} \tag{7.2}$$

In such case, the weighted measurement of the segmentation accuracy in Figure 7.8 is: $precision^{(w)} = \frac{|e_1|+|e_2|+|e_4|+|e_5|}{\sum_{i=1}^{t}|e_i|}$, and $recall^{(w)} = \frac{|t_1t_2|+|t_3t_4|+|t_4t_5|}{\sum_{i=1}^{5}|t_it_{i+1}|}$, where $|t_it_{i+1}|$ is the distance interval between the two neighboring tags.

After segmentation, the sequence of accelerometer frames are divided into several non-overlapping accelerometer episodes. We need to further calculate the features for episodes, which will be applied in learning micro-activity. The MA classification is the same technique applied in the frame-based MA inferring. The detailed useful features of accelerometer episode $\mathcal{E}_A$ has been defined previously in Table 7.2.

Definition 7.5 (Accelerometer Episodes - $\mathcal{E}_A$)**.** A set of accelerometer episodes, i.e., $\mathcal{E}_A = \{e_1, \ldots, a_m\}$. Episode features are calculated to describe its characteristics $\langle f_1, \cdots, f_n \rangle$, where the features are same as the definitions in Table 7.2.

The procedure of segmentation-based MA inference can be summarized as follows: (1) divide the accelerometer stream into frames, and calculate the basic features of each frame (e.g. three mean values of of the three axes); (2) apply the surface-based segmentation on accelerometer data with basic features, and validate it with real tags to find the best error bound; (3) based on the best error bound, rerun the first two steps and apply them on other accelerometer data, and extract their accelerometer episodes; (4) calculate the features of episodes that are used for later activity learning; and finally (5) apply the classification methods that similar to the frame-based MA inferring.

7.4 High-level Semantic Activity Inference

The second level of our activity inferencing framework approach requires us to transform the *MicroActivity* Stream $\mathcal{MS}$ of an individual into a corresponding High-Level Activity Stream ($\mathcal{HS}$), defined as follows.

Problem 2 (High-Level Activity Stream - $\mathcal{HS}$)**.** A transformation of $\mathcal{MS}$ such that $\mathcal{HS} = \langle HS_1, HS_2, \ldots, \rangle$, where HS_i is a tuple (h_i, t_i^b, t_i^e) such that $h_i \in \mathcal{HA}$ and t_i^b & t_i^e are timestamps denoting the start and end times of HS_i. Note that we implicity assume a non-overlapping High-Level Activity Stream–i.e., $t_{i+1}^b \geq t_i^e, \forall i$.

We now describe our empirical data-driven approach for identifying the discriminatory key features of each *MicroActivity* Sequence $\mathcal{MS}$ associated with a corresponding high-level activity $\mathcal{HA}$, so as to enable accurate classification of unknown elements of $\mathcal{HS}$. We describe and evaluate two broad approaches towards HA classification:

(a) the *Order-Oblivious* (OO) approach simply uses the *total duration* of each distinct MA as elements of the feature vector;

(b) the *Sequence-Aware* (SA) approach uses certain *sub-sequences of MAs* as additional discriminatory features.

To elucidate each approach, we will utilize the illustrative example in Figure 7.9, where we consider a sequence of MAs (e.g., 'walk (w)', 'sit (s)', 'stand (t)') associated with each HA instance. We further assume that, for this example, $T_f = 5$secs. (For our actual empirical study, recall that we have $M = 7$ different MAs, and T_f is varied among $(2, 5, 10, 20)$ seconds).

7.4.1 Description of Empirical User Study

Our HA study involved the same 5 users involved in the limited-duration MA classification study. Each of them was given a Nokia N95 smart phone (running our Python data collection scripts, that collected the 3-axis accelerometer and GPS data streams); 4 subjects used this phone as their primary communication device, while 1 used it solely to support our experimentation needs. Each user was asked to carry the phone with them as they went about performing their regular chores, and also maintained a separate diary of the HAs they performed at their respective office and home locations. Some initial idea was provided on what constituted a *high-level*

HA Label	**MA streams**	**OO Features** [w, s, t]	Temporal-Duration Preserving **(SA-TD)** subsequence (size: 3, θ≥0.6)			**SA-TD** Features [w, s, t, ttw, tww,wwt, tts, tss, sst]	Transition-Preserving **(SA-TP)** subsequence (size: 3, θ≥0.6)			**SA-TP** Features [w, s, t, twt, tst]
			sub_c	cov	supp		sub_c	cov	supp	
HE_1	**[ttttttwwwwt]** T-P seq: [twt]	[4, 0, 7]	[ttt]	1	1/3	[4,0,7,1,1,0,0,0]	**[twt]**	**2**	**2/3**	[4,0,7,1,0]
			[ttw]	**2**	**2/3**					
			[tww]	**3**	**3/3**					
HE_2	**[tww]** T-P seq: [tw]	[2, 0, 1]	[www]	1	1/3	[2,0,1,0,1,0,0,0]				[2,0,1,0,0]
			[wwt]	**2**	**2/3**					
HE_3	**[ttwwtt]** T-P seq: [twt]	[2, 0, 4]	[wtt]	1	1/3	[2,0,4,1,1,0,0,0]				[2,0,4,1,0]
HW_1	**[ttssssst]** T-P seq: [tst]	[0, 5, 3]	**[tts]**	**2**	**2/2**	[0,5,3,0,0,0,1,1,1]	**[tst]** [wts]	**2** 1	**2/2** 1/2	[0,5,3,0,1]
			[tss] [sss]	**2** 1	**2/2** 1/2					
HW_2	**[wwttsst]** T-P seq: [wtst]	[2, 2, 3]	**[sst]** [wwt] [wtt]	**2** 1 1	**2/2** 1/2 1/2	[2,2,3,0,0,1,1,1,1]				[2,2,3,0,1]

Figure 7.9: Example of features explored in our two-tier framework

Table 7.3: Summary of HA data collection campaign for different users

	User 1	User 2	User 3	User 4	User 5
Number of Days	27	17	15	23	19

activity 'HA' (e.g., work, break, lunch, dinner, cook). Although not mandatory, users often provided additional context for each tag (e.g., break_coffee, break_toilet, office_work_at_desk). The lifestyle-generated data was gathered over a span of *6 weeks on working days*, with obvious gaps due to individual variations in lifestyle routines (see Table 7.3). In total, we obtained 101 days of data, with each day containing between $\sim 4-15$ tags. As described in Figure 7.3, we employed a manual process of normalization to hierarchically transform each user tag to 5 high-level semantically-distinct HAs (some tags (e.g., baby_routine) were discarded). We call these HAs as "tier-1" HAs–our principal goal is to perform accurate disambiguation among these HAs by utilizing the discriminatory patterns embedded in the underlying accelerometer data.

The raw accelerometer stream $\mathcal{A}$ generated was first run through the calibration routine (described earlier for MA classification); subsequently, the (start,end) timestamps of each HA instance was used to extract the accelerometer segment for that instance. The MA classification model (learnt individually for each user) was then applied to generate the $\mathcal{MS}$ stream for each HA instance (e.g., Figure 7.9); these streams were then used to learn the HA classification models (using the two approaches described next). The accuracy of the resulting model was then computed using an 8-fold cross validation approach (we used 80% of the HA instances for training and 20% for testing).

7.4.2 The Order-Oblivious (OO) Approach

The order-oblivious approach associates a feature vector (with each instance of a specific HA), defined purely by the total number of frames (or equivalently, duration, as T_f is a constant) of each MA-type in the associated $\mathcal{MS}$. In other words, we have an M dimensional feature, where the i^{th} element of the vector denotes the number of frames classified as MA_i by the MA-classification algorithm. Looking at the example of Figure 7.9, there are 3 instances of the HA "home eat" (HE_1, HE_2 & HE_3), associated with the corresponding feature-vectors $[4,0,7]$, $[2,0,1]$ & $[2,2,2]$, respectively. These feature vectors represent specific points in an M-dimensional feature space; we then use well-known classification algorithms to compute HA classification models from these underlying data.

7.4.3 The Sequence-Aware (SA) Approach

This approach supplements the *OO* approach by extracting & defining additional features that consider the order (or sequence) in which the various MAs occur within a specific HA instance. This approach should provide more discriminatory capability and thus improve the HA classification accuracy, at the expense of higher dimensionality of the feature vector. We consider two pattern mining-based techniques to learn such key discriminatory features from the underlying

traces: **SA-TD**, a *duration-preserving* strategy and **SA-TF**, a *transition-preserving* strategy. To explain them, we first define a few terms.

Let $M_i = [MS_1^i, \ldots MS_l^i]$ be the set of MicroActivity sequences associated with the l different instances of HA_i (e.g., MS_1^1 =[t t t t t t w w w w t] Figure 7.9. Let $\mathcal{S}_j^i$ be the set of all sub-sequences of MS_j^i, i.e., a set of all possible MA-level sequences that occur in MS_j^i. Finally, sub_c be a sub-sequence in M_i, i.e., $sub_c \subseteq \cup_{j=1,\ldots,l}\mathcal{S}_j^i$; intuitively, sub_c is a MicroActivity subsequence that occurs at least once in the combination of all l instances of HA_i.

Definition 7.6 (Cover of sub_c)**.** Denoted as $\mathbf{cov}(sub_c, M_i)$, equals the number of instances in M_i that *contains* at least one instance of sub_c .

Definition 7.7 (Support of sub_c)**.** Denoted as $\mathbf{supp}(sub_c, M_i)$, equals $\frac{cov(sub_c, M_i)}{l}$.

The state-of-the-art has revealed that patterns with low support in individual classes or with very high support globally across classes are typically not useful for classification, as such patterns either occur very infrequently in any class instances or are a common occurrence in multiple classes, respectively [CYHH07]. In our scenario, an individual MA symbol (e.g., 'sit') is likely to be very common in all instances, while long MicroActivity sequences will have low *cover*. As defining new pattern mining algorithms is not the focus of our work, we present two intuitive strategies that aim to augment the *OO*-based feature vector with a few additional features defined by *sequences with high observed support.*

Temporal Duration-preserving strategy (SA-TD) – Given a minimum support threshold Θ_0 and a maximum subsequence size K_{max}, this strategy discovers the set of all sub-sequences (of length $[2, 3, \ldots, K_{max}]$) that $supp(sub_c, M_i) \geq \Theta_0$. For example, the subsequences, of length 3, selected for $Home\_eat$ in Figure 7.9 are [t t w, w w t, t w w]. The SA-TD algorithm finds the union of all such sub-sequences across all the H different HAs; this set of sub-sequences constitute the additional elements of the feature vector. For example, in Figure 7.9), this approach results in the selection of the following 3-element sub-sequences: [t t w, w w t, t w w, t t s, t s s, s s s].

The SA-TD algorithm then appends these features (to the *OO* feature vector) to create a longer feature vector. For classification purposes, the value of a particular element in the sequence-vector, defined for an instance HS_i of the high-level activity stream, is the number (or *frequency*) of occurrences of that element in the corresponding MicroActivity sequence. For example, for the instance HE_1 in Figure 7.9, the SA-TD approach results in a feature vector $[4, 0, 7, 1, 1, 0, 0, 0])$, where the elements [1,1,0,0,0] come from the sub-sequence based features (as the sequences '[ttw]' & '[wwt]' occur once each in HE_1). Note that if sub_c is identified as a feature, it follows that all sub-sequences of sub_c also meet the selection criteria and are considered candidate features. As the resulting feature vector may have fairly high dimensionality, this step of sequence-based feature identification is followed by a step of correlation-based reduction of the feature size (an approach described in [SH97]).

Transition-preserving strategy (SA-TP) – In this approach, the step of feature identification & selection is **preceeded** by a data transformation step, where we consider the MicroActivity sequence for a particular HA instance and only preserve the transitions between different

MAs, by removing (or collapsing) the run-length of consecutive MA symbols occuring in a particular instance of an HA. For example, in Figure 7.9, the sequence associated with HE_1 is transformed to [*twt*]. This transformation has the benefits of both *potentially higher discriminatory power* and *reduction of feature vector dimensionality*. By focusing on the sequence of *transitions* among consecutively occuring MAs, the SA-TP approach is robust to minor variations in the duration of individual MA. Consider for example, an activity defined as a 'smoking break'. It is highly likely that two instances of this HA might have a certain latent sequence (e.g., defined by an order of $walk \rightarrow stand \rightarrow walk$), while differently slightly in the duration of each micro-activity (e.g., [w w w t t w] and [w w t t t w w]). The SA-TP approach will identify both by the underlying common (or characteristic) transition [w t w], whereas SA-TD would treat them as different sequences. Moreover, by creating 1 (instead of 2 separate) sub-sequence from these observations, SA-TD reduces the dimensionality of the resulting feature vector. (For our experimental data, SA-2 results in approx. 2-4 fold reduction in the size of the resulting feature vector, compared to SA-TD).

In both SA-TD and SA-TP, the resulting set of higher dimensional feature vectors (containing both the frequency of individual samples and the frequency of certain MA sequences) are then provided to an appropriate classification algorithm to build a HA classification technique, which is subsequently validated with a set of test instances.

7.4.4 Baseline approach

To compare our 2-tier approach towards semantic activity mining (HA classification), we consider an alternative *baseline algorithm* typically used in traditional activity recognition literature [vKNEK08, GWT+09]. In this traditional scheme, features are extracted directly from the *raw accelerometer data stream*, after the initial step of calibration, associated with each HA instance. Hence, for all training instances of HA_i, we consider time frames (slices of duration T_f) of the sensor data stream, compute the (orientation-dependent and independent) features (just as we did for micro-activity classification) for each frame and build a classifier model after associating the corresponding HA (or semantic) label to each of these frames. Similarly, the test data is also divided into frames of size T_f and classification accuracy is measured using the *Timeslice accuracy* measure [vKNEK08], defined as the percentage of correctly classified frames, or $\frac{\sum_1^N [predicted(i)==real(i)}{N}$, where $predicted(i)$ is the predicted HS_{label} and $real(i)$ is the actual HS_{label} of the timeslice i, N=Total number of time slices (frames).

7.5 Experiment

In this section, we present some results related to accelerometer data segmentation, MA inference, and HA inference.

7.5.1 Accelerometer Segmentation

In the experiment of accelerometer data segmentation, we analyzed the sensitivity of different error bounds σ and various time unit τ for the segmentation accuracy.

From Figure 7.10, we observe (1) in general, the higher error threshold, the better precision; the extreme case is when the error bound is too big and there is only one segment (precision is 100%); (2) the optimal threshold σ in our experiment is 6.3, which can achieve 96% of the f-measure.

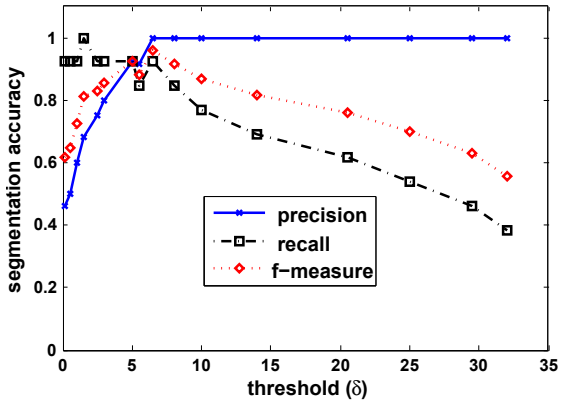


Figure 7.10: Segmentation sensitivity analysis of error bound

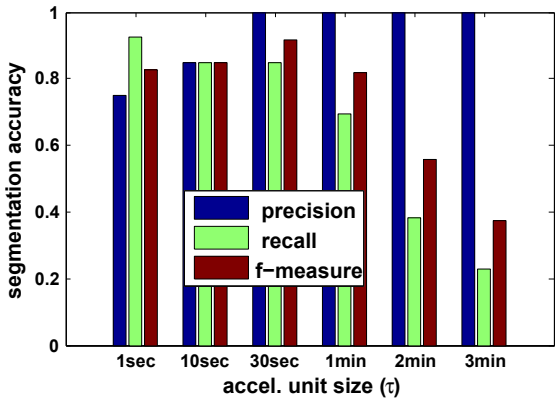


Figure 7.11: Segmentation sensitivity analysis of accelerometer frame size

The result on the sensitivity analysis of different acceleration time unit size is shown in Figure 7.11, which implies the result does not not highly depend on the accelerometer unit size when it is reasonable not too big (see 1 second, 10 second, 30 second in Figure 7.11). However, when it grows too much (closing to the activity duration), then the accuracy decreased significantly (see 3 minutes in Figure 7.11). Therefore, for the later experiment, we can adopt the suitable window size for both low computing cost and high segmentation accuracy.

Figure 7.12 shows the best segmentation result we achieved for one sample accelerometer data stream sequence, where (1) the x,y,z stream is using 30 seconds as the frame size, which is much smoothed compared with 1 second in Figure 7.8; (2) the circle dots are the ground truth user tags; (3) the dark lines are the division places that separate the two consecutive segments. We can observe the division lines are very consistent with the tag dots.

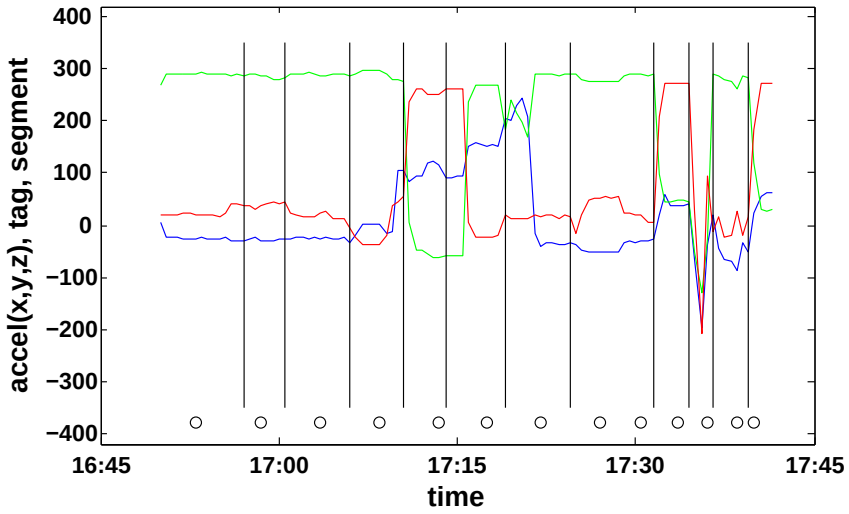


Figure 7.12: Segmentation results (division line) with the ground-truth tags)

7.5.2 Mining Micro-Activity

In Section 7.3.2, we have already addressed some empirical studies on MA classification of one user. The previous Figure 7.5 and Figure 7.6 did present the classification accuracy with the vertical phone orientation and the normal orientation in naturalized settings, as well as using different MA features. In addition, Figure 7.13 shows the sensitivity analysis on MA classification accuracy when using different frame sizes. Based on such results, we can claim the best results are achieved when using frame size as 5 seconds and 10 seconds, in particular for the combined case of phone positions.

Regarding MA classification across different users, Figure 7.14 plots the classification accuracy (various choices of 3D+2D feature vectors) for each of the remaining 4 test subjects, with the phone located in each individual user's most-preferred on-body position. The plot shows that the classification accuracy is uniformly high across all users (dropping to no lower than $\sim 90\%$ in the worst-case) and that the combined set of $3D_{all} + 2D_{all}$ features provides the overall best performance.

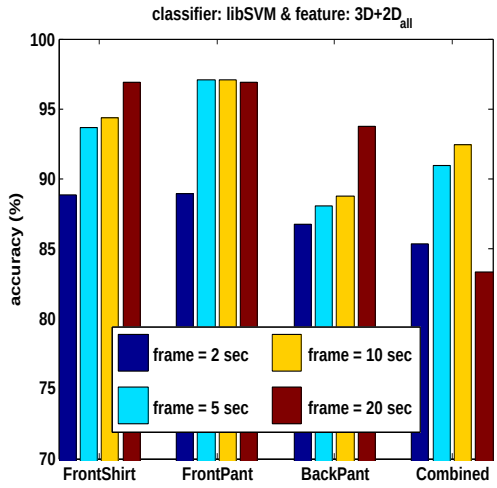


Figure 7.13: MA classification accuracy via different frame sizes

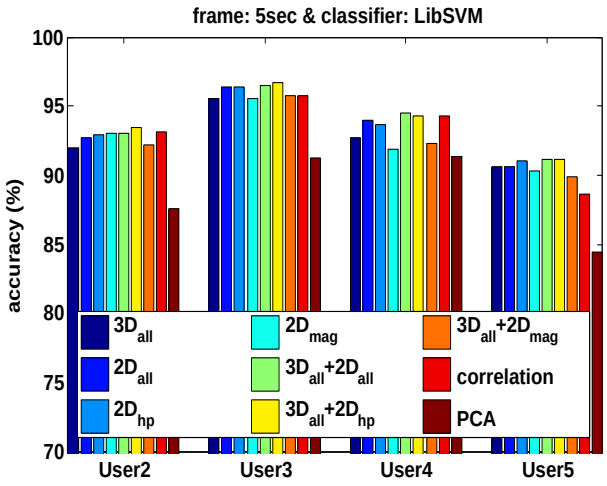


Figure 7.14: MA classification accuracy across all 5 users (T_f = 5secs)

7.5.3 Mining High-level Semantic Activity

Before presenting our two-tier approach's performance on HA inference, we address the performance of using baseline algorithm to infer the HAs. Figure 7.15 provides a comparison of the accuracy of 3 versions of the baseline HA mining algorithm, achieved under 6 candidate classifiers, for a specific user ($User1$). The figure shows that a traditional 'single-tier' approach to HA classification results in only $\sim 50\%$ accuracy values, indicating that the direct use of accelerometer-based features is not powerful enough to provide high-fidelity identification of higher-level semantic activities.

We then study the detailed result for $User1$, who was the most active participant in our data collection. Figure 7.16 presents the overall HA-classification accuracy observed by the 3 feature extraction algorithms (OO, $SA-TD$ and $SA-TF$), as well as a 4^{th} candidate that combines the $SA-TD$ & $SA-TP$-generated features framework, for two different choices

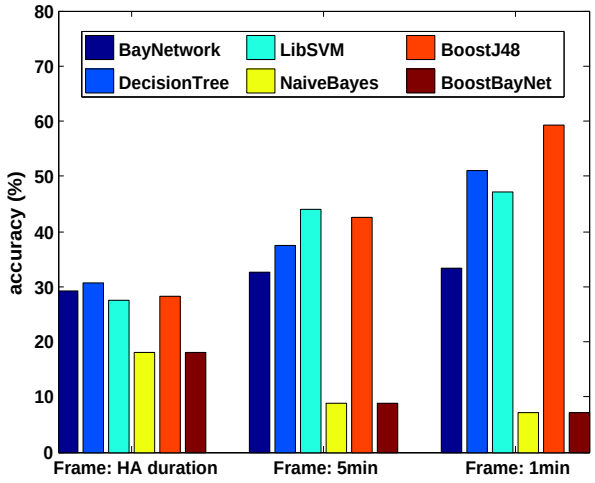


Figure 7.15: Performance of baseline algorithm using accelerometer-based features directly. The "Frame:HA Duration" condition corresponds to the case where the entire length of the HA instance is considered a single frame.

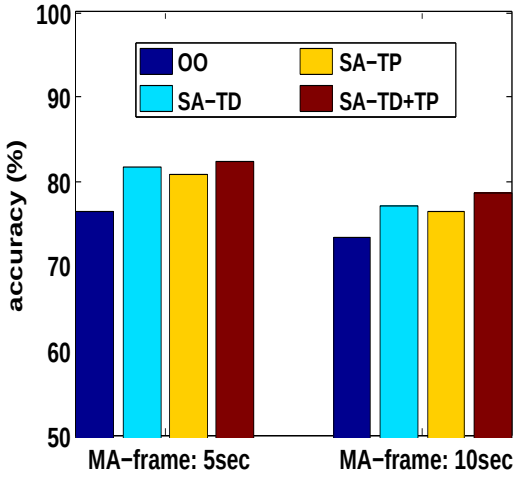


Figure 7.16: HA-Classification accuracy for different feature-extraction approaches in SAMMPLE ($User1$). $K_{max} = 4; \Theta_0 = 0.7$ for the SA-based approaches.

of MA frame-size ($T_f = \{5, 10\}$ sec), with $K_{max} = 4$ & $\Theta_0 = 0.7$. We can see that the OO approach achieves $\sim 70 - 75\%$ accuracy, whereas the addition of sequence-based features increases the accuracy by another $5 - 10\%$ (achieving a best case of $\sim 82\%$). Figure 7.17 also provide additional details about the "confusion matrix", generated by the classification of the 5 tier-1 HAs. Given the many possible sources of error in our data (e.g., user not carrying the smartphone, user forgetting to label certain HAs, lack of time synchronization between user logs of 'ground truth' and time-stamped accelerometer values), these results demonstrate that our SAMMPLE framework provides a powerful technique for correctly classifying a very high fraction of semantic activities.

$$\mathcal{T}= \begin{array}{c} \\ home\_eat(HE) \\ office\_break(OB) \\ office\_work(OW) \\ home\_work(HW) \\ home\_relax(HR) \end{array} \begin{array}{c} \begin{array}{ccccc} HE & OB & OW & HW & HR \end{array} \\ \begin{pmatrix} 10 & 0 & 0 & 4 & 2 \\ 0 & 20 & 6 & 0 & 0 \\ 0 & 1 & 30 & 0 & 0 \\ 0 & 0 & 0 & 24 & 3 \\ 5 & 0 & 0 & 2 & 24 \end{pmatrix} \end{array}$$

Figure 7.17: Confusion matrix for HA classification of $User1$.

In addition, we apply different strategies when inferring the MA list (e.g., $stand \rightarrow sit \rightarrow sitRelax$) for each HA instance, and analyze its influences on the final HA classification accuracy. For example, Figure 7.18 shows five MA classifiers ($maBayNet$, $maJ48$, $maLibSVM$, $maBoostJ48$) we have applied for inferring HA's MA list, as well as the four frame sizes ($2sec$, $4sec$, $10sec$, $20sec$). We can observe the best final HA result is achieved by using frame size as $5sec$ and the MA classifier as $maBayNet$. Similarly, Figure 7.18 shows the effects of the combination of difference choices in choosing MA classifier and HA classifiers.

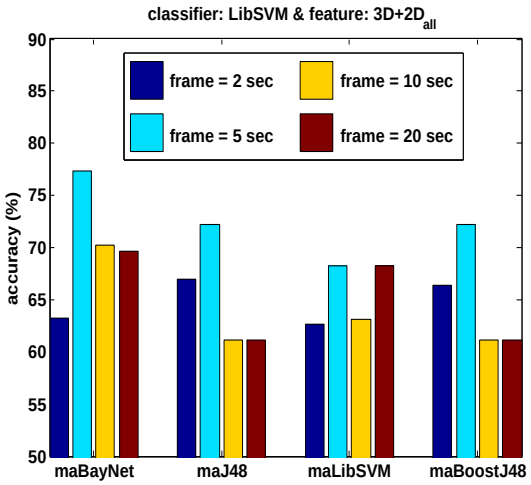


Figure 7.18: HA accuracy via different MA classifiers and MA frames

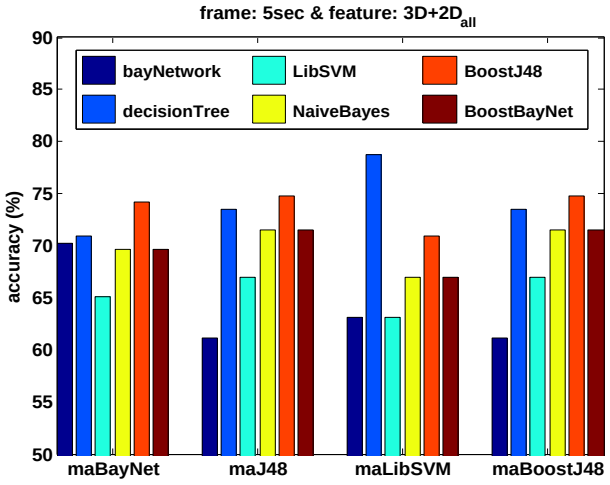


Figure 7.19: HA accuracy via different MA classifiers and HA classifiers

Finally, we study the HA classification across different users. Figure 7.20 reports the HA classification accuracy for each of the 4 other users, for a frame duration $T_f = 5$ secs. The figure demonstrates that the individualized SIMMPLE framework results in $\sim 70 - 80\%$ classification accuracy across users. The lower accuracy observed for $User2$ and $User5$ was due to the paucity of training data (with $User2$ participating only for 3 weeks, the number of HA instances was often less than 10).

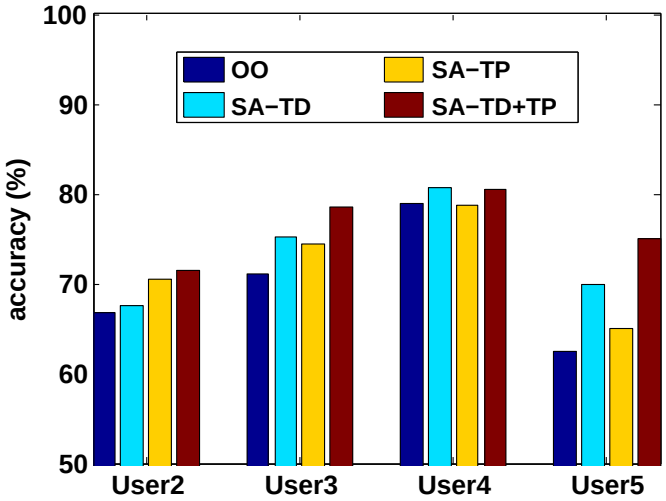


Figure 7.20: HA-classification accuracy for 5 subjects in SAMMPLE ($K_{max} = 4$; $\Theta_0 = 0.7$).

7.6 Summary

This chapter presents the last major contribution of this thesis, i.e., *mining semantic trajectories from multi-sensors* in smartphones, in particular the combination of two sensors, i.e, GPS and accelerometer. In details, we have presented, and evaluated, a two-tier framework for classification and mining of indoor semantic high-level activities (HA) based on smartphone-

generated accelerometer data. Unlike a single-tier approach, where classification features are directly computed from the raw accelerometer stream and associated with each HA instance, our two-tier approach first performs feature-based classification (using both orientation-dependent and orientation-dependent features) of individual frames to derive micro-activities (MAs) and then uses MA-level feature vectors to derive the HA. Experimental studies with human subjects show that our algorithms are able to achieve approximately 90% accuracy in identifying MAs and about 70% accuracy in then classifying the semantic activities (HAs) based purely on "MA duration"-dependent features and about 80% accuracy when the features are augmented to include representative MA *sequences*. Our results provide strong evidence that high-level activities can indeed be deduced from locomotion and posture-based MAs. The initial result of such two-tier learning framework is presented in [YCM+].

In addition to the frame-based MA inference, this chapter also discussed methods of accelerometer data segmentation. It is worth investigating how the two different strategies (*frame-based inference* vs *segmentation-based inference*) can influence the final learning results, both MA accuracy and HA accuracy. More interesting topics for future research directions could be supporting the online MA/HA classification as well as adding extra smartphone sensors, e.g., microphone, WiFi, bluetooth.

Chapter 8

Conclusion and Outlook

People-centric sensing applications can be thought of as having a personal, social, or public focus.

Andrew T. Campbell, 2008

8.1 Conclusion

In the last decades, the large availability of mobility data given by GPS-alike sensing devices has fostered many research interests and challenges for analyzing trajectories. In addition, smartphones with several embedded sensors (e.g., GPS, GSM, accelerometers, gyroscopes, WiFi, bluetooth, etc.) open up a world of new opportunities for sensing almost the whole lifestyle of an individual, including indoor activities (like shopping, eating at a restaurant, cooking at home or watching TV, being in a party) and outdoor activities (e.g., sports like skiing and cycling, driving, being in public transports). The smartphone can further record user's societal interaction with other people. Studies on analyzing smartphone-generated mobile sensing data, typically called "people sensing" [CEL+08, LML+10] and "reality mining" [EP06] in the literature, constitute one of the most exciting interdisciplinary research frontiers, like the computational social sciences [LPA+09].

Existing studies on such mobility data are mainly limited to analyzing the raw trajectories generated by the motion sensors, without enough consideration of associated higher-level semantics. This is the case for many sophisticated techniques for trajectory data ad-hoc storage, indexing, querying and mining that have been proposed in the trajectory database and data mining fields during the last decade. Recently, studies from the semantic web community and the activity recognition domain have started to build some meaningful concepts for representing mobility (e.g., stop and move). However, the gap between low-level sensing data and high-level mobility concept is still unclear. The contribution of this thesis is towards filling this gap. Therefore, the starting point and the first results of this thesis are two rich semantic models for mobility data representation discussed in Chapter 3, i.e., the *trajectory ontological framework* [YMPS08a, YMPS08b] and the *hybrid spatio-semantic trajectory model*

[Yan09, YPSC10]. The ontological framework provides knowledge of conjunctive queries and reasoning support for high-level trajectory concepts; whilst the second model enables progressively representing real-life mobility data at three different levels of data abstraction, i.e., from the low-level *spatio-temporal trajectories*, to the intermediate-level *structured trajectories*, and finally to the high-level *semantic trajectories*. This three-level hybrid model can provide a comprehensive view for understanding mobility data, bridging the gap between the spatio-temporal view and the semantic view.

Based on the hybrid spatio-semantic model, the second main outcome of this thesis is *a computing platform* that can progressively compute trajectories at different levels from the raw data collected by positioning devices. This thesis proposed both offline platform [YPSC10] and online solution [YGK+11] for trajectory computing. The offline computing procedure in Chapter 4 deals with past trajectories and consists of *data preprocessing* (e.g., techniques for data cleaning and compression), *trajectory identification* (i.e., identifying trajectory's start and ending points from the long sequence of tracking data, for generating spatio-temporal trajectories), and *trajectory segmentation* (i.e., the velocity, density, and time series methods for dividing a single spatio-temporal trajectory into a set of non-overlapping episodes, for achieving structured trajectory). Regarding online trajectory computing from streaming mobility data (dealing with ongoing trajectories), this thesis additionally proposed a real-time framework (called "SeTraStream") in Chapter 6, where the main focus is on online trajectory segmentation by using both location features and complementary features.

For achieving semantic trajectories completely from the raw mobility data, additional third party semantic knowledge is required, including both geographic knowledge (e.g., open-source map like Openstreetmap) and application domain knowledge (e.g., database of customers). This thesis defined a heterogeneous annotation framework for semantically enriching various trajectories with heterogeneous third party semantic data sources [YCP+11]. We categorized the third party semantic sources (comprising both geographic and application domain knowledge) in terms of their underlying spatial extents, i.e., region, line and point. Chapter 5 discussed the detailed annotation framework with three dedicated annotation algorithms, i.e., spatial join based annotation with semantic regions, map matching extended annotation with semantic lines, and hidden markov model based annotation with semantic points. Based on the computing platform and the annotation framework, we built a prototype (called "SeMiTri") with a Web interface to query and visualize different levels of trajectories [YSC+10].

As the last outcome, this thesis presented a method for computing semantic trajectories from multiple sensors embedded in smartphones, namely the study of combining GPS and accelerometer [YCM+]. Integrating GPS with accelerometer data can provide further semantics for better understanding mobility data, as it includes both the location (from GPS) and the locomotion (from accelerometer). Chapter 7 presented the two-layer framework to compute semantic indoor activities: firstly the framework identifies micro-activities (MA, e.g., walk, sit, sit-Relax, stand, bursty-Move) from accelerometer data; secondly based on GPS context about home and office, the framework infers high-level activities (HA, like office-work, home-cooking, home-dinner etc.), by discovering the patterns of a group of MAs.

8.2 Open Issues and Future Directions

The novel approaches of computing such "semantic trajectory" for better understanding mobility data in this thesis open many opportunities for future research.

8.2.1 Efficient Real-time & Distributed Trajectory Computing

Real-time system is always an important and challenging topic in computer sciences. In this thesis, we have already addressed relevant online trajectory computing algorithms for streaming movement data. The real-time processing tasks (such as online data cleaning, online compression, online segmentation, online annotation) need to be done in an efficient way. These online algorithms are built for processing data from one single node (e.g., GPS data from one Antenna). However, data in real-time systems is usually coming from different sources in a decentralized way, and cannot be preprocessed and merged together in advance. The real-life systems should be able to perform online trajectory computing in a distributed setting that is often encountered in large-scale application scenarios. Therefore, besides the *intra-computing*, we need to design algorithms for merging or refining results from multiple nodes (e.g., GPS data from multiple Antenna), which will be an *inter-computing* method. How to support trajectory computing in a distributed mode is a challenging problem and can be one of the major research focuses in future. Distributed and cloud computing techniques seem to be promising methodologies towards this direction. In particular, we would study minimizing communication costs, both for trajectory computing and results delivery in the real-time & distributed settings.

8.2.2 Collaborative & Personalized Semantic Trajectory Computing

Recently, there are emerging a lot of exciting geo-social networks, supporting collecting and annotating trajectories, such as Google Latitude, Foursquare, Facebook Place, Gowalla. New studies start to analyze such geo-social data for inferring trajectories and targeting better location & activity predictions as people sensing. It is worth investigating how semantic trajectory computing methods can contribute to the emerging research area of mobile-based people sensing. There would be the following main research problems along this direction:

(1) *Compute semantic trajectories with geo-social network data* – As geo-social networks enable users to conveniently tag (e.g., "twitter") their movement and social activities with user defined tags, we can use these tagging information to annotate and generate semantic trajectories, for extending the semantic annotation algorithms in Chapter 5.

(2) *Personalized semantic trajectory computing* – Semantics is not universal but meaningful in certain context of specific users. For example, the semantics of a stop in a restaurant is totally different between a customer having dinner and a waitress on duty. This thesis discussed common stops and personalized stops in trajectory computing. In future, advanced techniques are required for computing personalized semantic trajectories.

(3) *Collaborative semantic trajectory computing and recommendation* – Following the trend of computing a huge amount of personalized trajectories, we can discover the common behaviors, build individual user profiles, and further infer collaborative semantic trajectories, which will be significantly useful for location predication and recommendation services.

8.2.3 Online Activity Learning from Multiple Sensors

In Chapter 7, this thesis discussed the method of computing semantic trajectories with the combination of GPS and accelerometer. This work would have three major extensions:

(1) *Online HA/MA Classification* – Current MA (*Micro Activity*) and HA (*High-level Activity*) classification is offline, which assumes that there is an *a-priori* demarcation of the (start,end) time of a specific HA instance. In practice, the algorithms will receive a continuous accelerometer stream and thus have to perform an intermediate step of demarcating the boundaries of individual HAs (which are variable with unknown duration), before computing the features to build classifiers. Similar to online GPS-alike trajectory computing, we need to design online algorithms for computing semantic trajectories from accelerometer streaming data, as well as from the combination of GPS and accelerometer data.

(2) *Multi-level HA Inference* – The proposed solution deliberately focused on classifying HAs at level-1 granularity (Figure 7.3). In future, we will study HA tags on more concrete level, i.e., the level-0 tags. Additionally, different users provide different tags (i.e., the personalized semantic trajectory mentioned previously). We need to build techniques (e.g., clustering) to analyze these HA tags, and automatically build the tag hierarchy instead of a manually setting up. In addition to technical innovations of multi-level tags, we must also conduct larger-scale user studies to understand the precise HAs.

(3) *Additional Sensor Streams* – Besides GPS and accelerometer, we will explore extra sensors like WiFi, sound, bluetooth, gyroscopes, for exploring unsupervised or semi-supervised learning and classification techniques. A multi-sensory model supports more efficient computing as sensors are complementary to each other, e.g., accelerometer contributes more to indoor activities, GPS help inferring outdoor movement, and microphone samples can better distinguish if a person is working at his desk or participating in a meeting. In turn, the learning can rely less on ground-truth data, significantly reducing user tagging workload. For multi-sensor data, we will have to both derive the sensor-specific features that provide accurate classification under naturalistic conditions and adjust the classification logic to deal with potentially conflicting learning results from different sensor streams.

8.2.4 Smartphone-based Trajectory & LBS Applications

The goal of semantic trajectory approaches is to help building more useful trajectory or LBS (Location-based Services) applications, in particular on the smartphone platform. For real-life mobile applications, two important issues cannot be ignored:

(1) *Energy-efficient computing.* Considering the power limitation of smartphones, we will analyze the energy sensitivity issues regarding different kinds of activities, and provide more dynamic sampling frequency and computing strategies to save the smartphone energy.

(2) *Privacy and security aspect.* With more mobile application deployed and more data collected, the problem of privacy and security is bound to increase. This thesis did not cover the issues of privacy-preserving semantic trajectory computing, but this definitely will be an important research topic in future.

Bibliography

[ABK+07] Luis Ottavio Alvares, Vania Bogorny, Bart Kuijpers, Jose Macedo, Bart Moelans, and Alejandro Vaisman. A Model for Enriching Trajectories with Semantic Geographical Information. In *GIS '07: Proceedings of the 15th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems*, page 22, 2007. 3, 7, 29, 76, 101, 110

[ABKS99] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. OPTICS: Ordering Points To Identify the Clustering Structure. *SIGMOD '99: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 49–60, 1999. 25

[AF00] Alessandro Artale and Enrico Franconi. A Survey of Temporal Extensions of Description Logics. *Annals of Mathematics and Artificial Intelligence*, 30(1-4):171–210, 2000. 28

[AGLW07] Mattias Andersson, Joachim Gudmundsson, Patrick Laube, and Thomas Wolle. Reporting Leadership Patterns Among Trajectories. In *SAC '07: Proceedings of the 2007 ACM Symposium on Applied Computing*, pages 3–7, 2007. 26

[AR99] Tamas Abraham and John F Roddick. Survey of Spatio-Temporal Databases. *Geoinformatica*, 3(1):61–99, 1999. 11

[AVH+06] Aris Anagnostopoulos, Michail Vlachos, Marios Hadjieleftheriou, Eamonn Keogh, and Philip S Yu. Global distance-based segmentation of trajectories. In *KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 34–43, 2006. 23

[BDKS10] Maike Buchin, Anne Driemel, Marc Van Kreveld, and Vera Sacristan. An Algorithmic Framework for Segmenting Trajectories based on Spatio-Temporal Criteria. In *GIS '10: Proceedings of the 18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 202–211, 2010. 24, 87

[BDS09] Thomas Bittner, Maureen Donnelly, and Barry Smith. A spatio-temporal ontology for geographic information integration. *International Journal of Geographical Information Science*, 23(6):765–798, 2009. 28

[BFS10] Athanasios Bamis, Jia Fang, and Andreas Savvides. A Method for Discovering Components of Human Rituals from Streams of Sensor Data. In *CIKM '10: Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, 2010. 30

[BGC09] Tomas Brezmes, Juan-Luis Gorricho, and Josep Cotrina. Activity Recognition from Accelerometer Data on a Mobile Phone. In *IWANN '09 Proceedings of the 10th International Work-Conference on Artificial Neural Networks*, pages 796–799, 2009. 144, 157

[BGHW08] Marc Benkert, Joachim Gudmundsson, Florian Hübner, and Thomas Wolle. Reporting flock patterns. *Computational Geometry*, 41(3):111–125, 2008. 26

[BI04] Ling Bao and Stephen Intille. Activity Recognition from User-Annotated Acceleration Data. In *Pervasive '04: Proceedings of the 2nd International Conference on Pervasive Computing*, pages 1–17, 2004. 31, 155, 157

[BJKS02] Rimantas Benetis, Christian S Jensen, Gytis Karciauskas, and Simonas Saltenis. Nearest Neighbor and Reverse Nearest Neighbor Queries for Moving Objects. In *IDEAS '02: Proceedings of the 2002 International Symposium on Database Engineering and Applications*, pages 44–53, 2002. 2

[BJKS06] Rimantas Benetis, S Jensen, Gytis Karciauskas, and Simonas Saltenis. Nearest and Reverse Nearest Neighbor Queries for Moving Objects. *The VLDB Journal*, 15(3):229–249, 2006. 2, 18

[BJR94] George Box, Gwilym M Jenkins, and Gregory C Reinsel. *Time Series Analysis: Forecasting and Control*. Prentice-Hall, 1994. 82

[BK96] D Bernstein and A Kornhauser. *An Introduction to Map Matching for Personal Navigation Assistants*. Number August. Princeton University, 1996. 20, 21

[BK07] Derya Birant and Alp Kut. ST-DBSCAN: An Algorithm for Clustering Spatial-Temporal Data. *Data and Knowledge Engineering*, 60(1):208–221, 2007. 25

[BKS93] Thomas Brinkhoff, Hans-Peter Kriegel, and Bernhard Seeger. Efficient Processing of Spatial Joins using R-Trees. In *SIGMOD '93: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 237–246, 1993. 102

[BKS07] Faisal I Bashir, Ashfaq A Khokhar, and Dan Schonfeld. Object Trajectory-Based Activity Classification and Recognition Using Hidden Markov Models. *IEEE Transactions on Image Processing*, 16(7):1912–1919, 2007. 26

[BKSS90] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. *SIGMOD Record*, 19(2):322–331, 1990. 15, 103

[BLPW08] Bhuvan Bamba, Ling Liu, Péter Pesti, and Ting Wang. Supporting Anonymous Location Queries in Mobile Environments With Privacygrid. In *WWW '08: Proceedings of the17th International Conference on World Wide Web*, pages 237–246, 2008. 19

[BPSW05] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. On Map-Matching Vehicle Tracking Data. In *VLDB '05: Proceedings of the 31st International Conference on Very Large Data Bases*, pages 853–864, 2005. 20, 21, 63, 66

[Bri03] Thomas Brinkhoff. Generating Traffic Data. *IEEE Data Engineering Bulletin*, 26(2):19–25, 2003. 63

[CBC+08] Tanzeem Choudhury, Gaetano Borriello, Sunny Consolvo, Dirk Hähnel, Beverly L Harrison, Bruce Hemingway, Jeffrey Hightower, Predrag V Klasnja, Karl Koscher, Anthony LaMarca, James A Landay, Louis LeGrand, Jonathan Lester, Ali Rahimi, Adam D Rea, and Danny Wyatt. The Mobile Sensing Platform: An Embedded Activity Recognition System. *IEEE Pervasive Computing*, 7(2):32–41, 2008. 31, 32, 151

[CCJ10] Xin Cao, Gao Cong, and Christian S Jensen. Mining Significant Semantic Locations From GPS Trajectories. In *VLDB '10: Proceedings of the VLDB Endowment*, pages 1009–1020, 2010. 30

[CD10] Driss Choujaa and Naranker Dulay. Predicting Human Behaviour from Selected Mobile Phone Data Points. In *UbiComp '10: Proceedings of the 12th international conference on Ubiquitous computing*, pages 105–108, 2010. 31

[CEL+08] Andrew Campbell, Shane Eisenman, Nicholas Lane, Emiliano Miluzzo, Ronald Peterson, Hong Lu, Xiao Zheng, Mirco Musolesi, Krist Fodor, and Gahng-Seop Ahn. The Rise of People-Centric Sensing. *IEEE Internet Computing*, 12:12–21, 2008. 31, 144, 169

[CJL08] Su Chen, Christian S Jensen, and Dan Lin. A Benchmark for Evaluating Moving Object Indexes. In *VLDB '08: Proceedings of the VLDB Endowment*, volume 1(2), pages 1574–1585, 2008. 2, 16

[CMC06] Huiping Cao, Nikos Mamoulis, and David W Cheung. Discovery of Collocation Episodes in Spatiotemporal Data. In *ICDM '06: Proceedings of the 2006 IEEE International Conference on Data Mining*, pages 823–827, 2006. 134

[CMC07] Huiping Cao, Nikos Mamoulis, and David W Cheung. Discovery of Periodic Patterns in Spatiotemporal Sequences. *IEEE Transactions on Knowledge and Data Engineering*, 19(4):453–467, 2007. 25

[CWT06] Hu Cao, Ouri Wolfson, and Goce Trajcevski. Spatio-Temporal Data Reduction With Deterministic Error Bounds. *The VLDB Journal*, 15(3):211–228, 2006. 22

[CYHH07] Hong Cheng, Xifeng Yan, Jiawei Han, and Chih-Wei Hsu. Discriminative Frequent Pattern Analysis for Effective Classification. In *ICDE '07: Proceedings of the 2007 IEEE 23th International Conference on Data Engineering*, pages 716–725, 2007. 162

[DAG05a] De, Victor Teixeira Almeida, and Ralf Hartmut Güting. Indexing the Trajectories of Moving Objects in Networks. *GeoInformatica*, 9(1):33–60, 2005. 15

[dAG05b] Victor Teixeira de Almeida and Ralf Hartmut Güting. Supporting Uncertainty in Moving Objects in Network Databases. In *GIS '05: Proceedings of the 13th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 31–40, 2005. 14

[dAGB06] Victor Teixeira de Almeida, Ralf Hartmut Güting, and Thomas Behr. Querying Moving Objects in SECONDO. In *MDM '06: Proceedings of the 7th International Conference on Mobile Data Management*, page 47, 2006. 2

[DBG07] Christian Düntgen, Thomas Behr, and Ralf Hartmut Güting. BerlinMOD: A Benchmark for Moving Object Databases. *The VLDB Journal*, 18(4):1335–1368, 2007. 13, 16

[DKV+09] A Deligiannakis, Y Kotidis, V Vassalos, V Stoumpos, and A Delis. Another Outlier Bites the Dust: Computing Meaningful Aggregates in Sensor Networks. In *ICDE '09: Proceedings of the 25th International Conference on Data Engineering*, pages 988–999, 2009. 133, 135, 136

[DP73] David Douglas and Thomas Peucker. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature. *The Canadian Cartographer*, 10(2):112–122, 1973. 2

[DP06] Evangelos Dellis and Georgios Paliouras. Management of Large Spatial Ontology Bases. In *ODBIS '06: Proceedings of the Ontologies-Based Databases and Information Systems*, pages 102–118, 2006. 27

[DWL08] Somayeh Dodge, Robert Weibel, and Anna-Katharina Lautenschütz. Towards a taxonomy of movement patterns. *Information Visualization*, 7(3):240–252, 2008. 26

[EGSV99] Martin Erwig, Ralf Hartmut Güting, Markus Schneider, and Michalis Vazirgiannis. Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases. *Geoinformatica*, 3(3):269–296, 1999. 40

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD '96: Proceedings of the 2nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996. 25, 80

[EP06] Nathan Eagle and Alex Pentland. Reality Mining: Sensing Complex Social Systems. *Personal and Ubiquitous Computing*, 10(4):255–268, 2006. 31, 169

[FGPT05] Elias Frentzos, Kostas Gratsias, Nikos Pelekis, and Yannis Theodoridis. Nearest Neighbor Search on Moving Object Trajectories. In *SSTD '05: Proceedings of the 9th International Symposium Advances in Spatial and Temporal Databases*, pages 328–345, 2005. 18

[For73] George David Forney. The Viterbi Algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973. 114

[FPSSU96] Usama M Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996. 139

[Fre08] Elias Frentzos. *Trajectory Data Management in Moving Object Databases*. PhD thesis, University of Piraeus, 2008. 13, 14, 15, 23

[FT07] Elias Frentzos and Yannis Theodoridis. On the Effect of Trajectory Compression in Spatiotemporal Querying. In *ADBIS '07: Proceedings of the 11th East European Conference on Advances in Databases and Information Systems*, pages 217–233, 2007. 2, 23

[GÖ5] Ralf Hartmut Güting. SECONDO: A Database System for Moving Objects. *GeoInformatica*, 9(1):33–60, 2005. 2, 13

[GÖ7] Ralf Hartmut Güting. How to Build Your Own Moving Objects Database System. In *MDM '07: Proceedings of the 2007 International Conference on Mobile Data Management*, pages 1–2, 2007. 13

[GBE+00] Ralf Hartmut Güting, Michael H Böhlen, Martin Erwig, Christian S Jensen, Nikos A Lorentzos, Markus Schneider, and Michalis Vazirgiannis. A Foundation for Representing and Querying Moving Objects. *ACM Transactions on Database Systems*, 25(1):1–42, 2000. 2, 12, 13, 14, 40

[GdAA+05] Ralf Hartmut Güting, Victor Teixeira de Almeida, Dirk Ansorge, Thomas Behr, Zhiming Ding, Thomas Höse, Frank Hoffmann, Markus Spiekermann, and Ulrich Telle. SECONDO: An Extensible DBMS Platform for Research Prototyping and Teaching. In *ICDE '05: Proceedings of the 21th International Conference on Data Engineering*, pages 1115–1116, 2005. xv, 12, 13

[GdAD06] Ralf Hartmut Güting, Victor Teixeira de Almeida, and Zhiming Ding. Modeling and Querying Moving Objects in Networks. *The VLDB Journal*, 15(2):165–190, 2006. 14, 20, 59, 62

[GDV08] Aris Gkoulalas-Divanis and Vassilios S. Verykios. A privacy-aware trajectory tracking query engine. *SIGKDD Explorations*, 10(1):40–49, 2008. 14

[GHB08] Marta C Gonzalez, A Hidalgo, and Albert-Laszlo Barabasi. Understanding Individual Human Mobility Patterns. *Nature*, 2008. 31

[GHP+03] Chris Giannella, Jiawei Han, Jian Pei, Xifeng Yan, and Philip S Yu. Mining Frequent Patterns in Data Streams at Multiple Time Granularities. In *Next Generation Data Mining*, pages 191–212. AAAI/MIT Press, 2003. 128, 129, 137

[GKD+10] Nikos Giatrakos, Yannis Kotidis, Antonios Deligiannakis, Vasilis Vassalos, and Yannis Theodoridis. Taco: Tunable Approximate Computation of Outliers in Wireless Sensor Networks. *SIGMOD '10: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, June 2010. 133, 135, 136

[GKV08] Leticia I Gómez, Bart Kuijpers, and Alejandro A Vaisman. Aggregation Languages for Moving Object and Places of Interest. *SAC '08: Proceedings of the 2008 ACM Symposium on Applied Computing*, pages 857–862, 2008. 29, 53

[GLC+07] Yun-Jun Gao, Chun Li, Gen-Cai Chen, Ling Chen, Xian-Ta Jiang, and Chun Chen. Efficient K-Nearest-Neighbor Search Algorithms for Historical Moving Object Trajectories. *Journal of Computer Science and Technology*, 22(2):232–244, 2007. 2, 18

[GLW08] Joachim Gudmundsson, Patrick Laube, and Thomas Wolle. Movement Patterns in Spatio-temporal Data. *Encyclopedia of GIS*, pages 726–732, 2008. 26

[GNPP07] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. Trajectory Pattern Mining. In *KDD '07: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 330–339, 2007. 24, 25, 144

[GP08] Fosca Giannotti and Dino Pedreschi. *Mobility, Data Mining and Privacy, Geographic Knowledge Discovery.* Springer-Verlag, 2008. 12, 24

[GP09] Gyözö Gidófalvi and Torben Bach Pedersen. Mining Long, Sharable Patterns in Trajectories of Moving Objects. *GeoInformatica*, 13(1):27–55, 2009. 25

[Gre02] Joshua S Greenfeld. Matching GPS Observations to Locations on a Digital Map. In *81th Annual Meeting of the Transportation Research Board*, number 3, pages 164–173, 2002. 21

[Gru95] Thomas R Gruber. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal of Human Computer Studies*, 43(5-6):907–928, 1995. 26

[GS95] Ralf Hartmut Güting and Markus Schneider. Realm-Based Spatial Data Types: The ROSE Algebra. *VLDB Journal*, 4:243–286, 1995. 98

[GS04] Pierre Grenon and Barry Smith. SNAP and SPAN: Towards Dynamic Spatial Ontology. *Spatial Cognition & Computation: An Interdisciplinary Journal*, 4(1):69–104, 2004. 27

[GS05] Ralf Hartmut Güting and Markus Schneider. *Moving Objects Databases*. Morgan Kaufmann, 2005. 2, 11, 12, 13, 50

[Gut84] Antonin Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. *SIGMOD '84: Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, pages 47–57, 1984. 15

[GWT+09] Tao Gu, Zhanqing Wu, XianPing Tao, Hung Keng Pung, and Jian Lu. epSICAR: An Emerging Patterns based Approach to Sequential, Interleaved and Concurrent Activity Recognition. In *PerCom '09: Proceedings of the IEEE Pervasive Computing and Communication Conference*, pages 1–9, 2009. 31, 155, 163

[HA06] Milan Horemuž and Johan Vium Andersson. Polynomial Interpolation of GPS Satellite Coordinates. *GPS Solutions*, 10(1):67–72, 2006. 20

[HBK+07] Jo A Helmuth, Christoph J Burckhardt, Petros Koumoutsakos, Urs F Greber, and Ivo F Sbalzarini. A Novel Supervised Trajectory Segmentation Algorithm Identifies Distinct Types of Human Adenovirus Motion in Host Cells. *Journal of Structural Biology*, 159(3):347–358, 2007. 24

[HFS08] Tâm Huynh, Mario Fritz, and Bernt Schiele. Discovery of Activity Patterns Using Topic Models. In *UbiComp '08: Proceedings of the 10th international conference on Ubiquitous computing*, pages 10–19, 2008. 31

[HGmRM10] Nicola Hönle, Matthias Groß mann, Steffen Reimann, and Bernhard Mitschang. Usability Analysis of Compression Algorithms for Position Data Streams. In *GIS '10: Proceedings of the 18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 240–249, 2010. 23

[HLGL08] Jiawei Han, Jae-Gil Lee, Hector Gonzalez, and Xiaolei Li. Mining Massive RFID, Trajectory, and Traffic Data Sets (Tutorial). In *KDD '08: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008. 24, 26, 144

[HMA+04] Moustafa A. Hammad, Mohamed F. Mokbel, Mohamed H. Ali, Walid G. Aref, Ann Christine Catlin, Ahmed K. Elmagarmid, Mohamed Y. Eltabakh, Mohamed G. Elfeky, Thanaa M. Ghanem, Robert Gwadera, Ihab F. Ilyas, Mirette S. Marzouk, and Xiaopeng Xiong. Nile: A query processing engine for data streams. In *ICDE '04: Proceedings of the 2004 IEEE 20th International Conference on Data Engineering*, page 851, 2004. 14

[HMTT08] Marios Hadjieleftheriou, Yannis Manolopoulos, Yannis Theodoridis, and Vassilis J Tsotras. R-Trees - A Dynamic Index Structure for Spatial Searching. *Encyclopedia of GIS*, pages 993–1002, 2008. 15

[HP04] Jerry R Hobbs and Feng Pan. An Ontology of Time for the Semantic Web. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(1):66–85, 2004. 27

[HYD99] Jiawei Han, Yiwen Yin, and Guozhu Dong. Efficient Mining of Partial Periodic Patterns in Time Series Database. In *ICDE '99: Proceedings of the 15th International Conference on Data Engineering*, page 106, 1999. 25

[JGO06] Jungwook Jun, Randall Guensler, and Jennifer Ogle. Smoothing Methods to Minimize Impact of Global Positioning System Random Error on Travel Distance, Speed, and Acceleration Profile Estimates. *Transportation Research Record: Journal of the Transportation Research Board*, 1972(1):141–150, January 2006. 19, 59, 61, 132, 133

[JLO04] Christian S Jensen, Dan Lin, and Beng Chin Ooi. Query and Update Efficient B^+-Tree Based Indexing of Moving Objects. In *VLDB '04: Proceedings of the 30th International Conference on Very Large Data Bases*, pages 768–779, 2004. 16

[JLO07] Christian S Jensen, Dan Lin, and Beng Chin Ooi. Continuous Clustering of Moving Objects. *IEEE Transactions on Knowledge and Data Engineering*, 19(9):1161–1174, 2007. 25

[JSZ07] Hoyoung Jeung, Heng Tao Shen, and Xiaofang Zhou. Mining Trajectory Patterns Using Hidden Markov Models. In *DaWaK '07: Proceedings of the 9th International Conference on Data Warehousing and Knowledge Discovery*, pages 470–480, 2007. 2, 26

[JTT06a] Christian S Jensen, Dalia Tiesyte, and Nerius Tradisauskas. Robust B+-Tree-Based Indexing of Moving Objects. In *MDM '06: Proceedings of the 7th International Conference on Mobile Data Management*, page 12, 2006. 16

[JTT06b] Christian S Jensen, Dalia Tiesyte, and Nerius Tradisauskas. The COST Benchmark-Comparison and Evaluation of Spatio-temporal Indexes. In *DASFAA '06: Proceedings of the 2006 Database systems for advanced applications*, pages 125–140, 2006. 16

[JYZ+08] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S Jensen, and Heng Tao Shen. Discovery of Convoys in Trajectory Databases. In *VLDB '08: Proceedings of the 34th International Conference on Very Large Data Bases*, pages 1068–1080, 2008. 2, 26

[KBD+10] Niko Kiukkonen, Jan Blom, Olivier Dousse, Daniel Gatica-Perez, and Juha Laurila. Towards rich mobile phone datasets: Lausanne data collection campaign. In *ICPS '10: the ACM International Conference on Pervasive Services*, 2010. 32, 34, 118, 140, 154

[KCHP01] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. An Online Algorithm for Segmenting Time Series. In *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 289–296, 2001. 157

[KCHP04] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. Segmenting Time Series: A Survey and Novel Approach. In *Data mining in Time Series Databases*, pages 1–22. World Scientific, 2004. 72, 157

[KD08] Bhargav Kanagal and Amol Deshpande. Online Filtering, Smoothing and Probabilistic Modeling of Streaming data. *ICDE '08: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 1160–1169, 2008. 133

[KLLK10] A M Khan, Y.-K. Lee, S Y Lee, and T.-S. Kim. Human Activity Recognition via an Accelerometer-Enabled-Smartphone Using Kernel Discriminant Analysis. In *FutureTech '10: Proceedings of the 5th International Conference on Future Information Technology*, pages 1–6, 2010. 157

[KMB05] Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. On Discovering Moving Clusters in Spatio-temporal Data. In *SSTD '05: Proceedings of the 9th International Symposium Advances in Spatial and Temporal Databases*, pages 364–381, 2005. 25

[KMOV09] Bart Kuijpers, Bart Moelans, Walied Othman, and Alejandro A Vaisman. Analyzing Trajectories Using Uncertainty and Background Information. In *SSTD '09: Proceedings of the 11th International Symposium Advances in Spatial and Temporal Databases*, pages 135–152, 2009. 29

[KO07] Bart Kuijpers and Walied Othman. Trajectory Databases: Data Models, Uncertainty and Complete Query Languages. In *ICDT '07: Proceedigs of the 2007 International Conference on Database Theory*, pages 224–238, 2007. 12, 14, 40

[KPT09] Georgios Kellaris, Nikos Pelekis, and Yannis Theodoridis. Trajectory Compression under Network Constraints. In *SSTD '09: Proceedings of the 11th International Symposium Advances in Spatial and Temporal Databases*, pages 392–398, 2009. 2, 134

[KSF+03] Manolis Koubarakis, Timos K Sellis, Andrew U Frank, Stéphane Grumbach, Ralf Hartmut Güting, Christian S Jensen, Nikos A Lorentzos, Yannis Manolopoulos, Enrico Nardelli, Barbara Pernici, Hans-Jörg Schek, Michel Scholl, Babis Theodoulidis, and Nectaria Tryfona, editors. *Spatio-Temporal Databases: The CHOROCHRONOS Approach*, volume 2520 of *Lecture Notes in Computer Science*. Springer, 2003. 2, 11, 12

[KVD+07] Y Kotidis, V Vassalos, A Deligiannakis, V Stoumpos, and A Delis. Robust Management of Outliers in Sensor Network Aggregate Queries. In *MobiDE '07: Proceedings of the 6th ACM International Workshop on Data Engineering for Wireless and Mobile Access*, 2007. 133, 135, 136

[KWM10] J R Kwapisz, G M Weiss, and S A Moore. Activity Recognition using Cell Phone Accelerometers. In *SensorKDD '10: Proceedings of the 4th International Workshop on Knowledge Discovery from Sensor Data*, pages 10–18, 2010. 157

[Las06] Matthew Lashley. Kalman Filter Based Tracking Algorithms For Software GPS Receivers. Master's thesis, Auburn University, the Netherlands, 2006. 19

[LDH+10] Zhenhui Li, Bolin Ding, Jiawei Han, Roland Kays, and Peter Nye. Mining Periodic Behaviors for Moving Objects. In *KDD '10: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1099–1108, 2010. 25, 30

[LFK07] L Liao, D Fox, and H Kautz. Extracting Places and Activities from GPS Traces Using Hierarchical Conditional Random Fields. *The International Journal of Robotics Research*, 26(1):119–134, 2007. 29

[LHL08] Jae-Gil Lee, Jiawei Han, and Xiaolei Li. Trajectory Outlier Detection: A Partition-and-Detect Framework. In *ICDE '08: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 140–149. Ieee, 2008. 26

[LHLG08] Jae-Gil Lee, Jiawei Han, Xiaolei Li, and Hector Gonzalez. TraClass: Trajectory Classification Using Hierarchical Region-Based and Trajectory-Based Clustering. In *VLDB '08: Proceedings of the VLDB Endowment*, pages 1081–1094, 2008. 2, 23, 26

[LHW07] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory Clustering: a Partition-and-Group Framework. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 593–604, 2007. 2, 23, 26, 135

[LJL+10] Zhenhui Li, Ming Ji, Jae-Gil Lee, Lu-An Tang, Yintao Yu, Jiawei Han, and Roland Kays. MoveMine: Mining Moving Object Databases. In *SIGMOD '10: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, pages 1203–1206, 2010. xv, 26, 27

[LLHL09] Xiaolei Li, Zhenhui Li, Jiawei Han, and Jae-Gil Lee. Temporal Outlier Detection in Vehicle Traffic Data. In *ICDE '09: Proceedings of the 25th International Conference on Data Engineering*, pages 1319–1322, 2009. 26

[LLLH10] Zhenhui Li, Jae-Gil Lee, Xiaolei Li, and Jiawei Han. Incremental Clustering for Trajectories. *DASFAA '10: Proceedings of the 2010 Database systems for advanced applications*, pages 32–46, 2010. 22

[Llo82] Stuart P Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–136, 1982. 25

[LML+10] Nicholas D Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T Campbell. A survey of mobile phone sensing. *Communications Magazine*, 48(9):140–150, September 2010. 31, 169

[LOY05] Bernhard Lorenz, Hans Jürgen Ohlbach, and Laibing Yang. Ontology of Transportation Networks, 2005. 42

[LPA+09] David Lazer, Alex Pentland, Lada A Adamic, Sinan Aral, Albert-Laszlo Barabasi, Devon Brewer, Nicholas Christakis, Noshir Contractor, James Fowler, Myron Gutmann, Tony Jebara, Gary King, Michael Macy, Deb Roy, and Marshall Van Alstyne. Computational Social Science. *Science*, 323:721–723, 2009. 31, 169

[LPFK06] Lin Liao, Donald J Patterson, Dieter Fox, and Henry Kautz. Building Personal Maps from GPS Data. *Annals Of The New York Academy Of Sciences*, 1093(1):249–265, 2006. 29

[Lut02] Carsten Lutz. *The Complexity of Description Logics with Concrete Domains*. PhD thesis, RWTH Aachen, 2002. 28

[LWC+07] Caifeng Lai, Ling Wang, Jidong Chen, Xiaofeng Meng, and Karine Zeitouni. Effective Density Queries for Moving Objects in Road Networks. In *APWeb/WAIM '07: Proceedings of the Joint 9th Asia-Pacific Web Conference and the 8th International Conference on Web-Age Information Management*, pages 200–211, 2007. 25

[LYL+10] Hong Lu, Jun Yang, Zhigang Liu, Nicholas D Lane, Tanzeem Choudhury, and Andrew T Campbell. The Jigsaw Continuous Sensing Engine for Mobile Phone Applications. In *SenSys '10: Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 71–84, 2010. 31, 32, 152

[LZZ+09] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. Map-Matching for Low-Sampling-Rate GPS Trajectories. In *GIS '09: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 352–361. ACM, 2009. 21

[MD03] Xiaofeng Meng and Zhiming Ding. DSTTMOD: A Future Trajectory Based Moving Objects Database. In *DEXA '03: Proceedings of the 14th International Conference on Database and Expert Systems Applications*, pages 444–453, 2003. 13

[MdB04] Nirvana Meratnia and Rolf A de By. Spatiotemporal Compression Techniques for Moving Point Objects. In *EDBT '04: Proceedings of the 9th International Conference on Extending Database Technology*, pages 765–782, 2004. 2, 22, 23, 67, 134

[MEB+99] D Mark, M Egenhofer, L Bian, K Hornsby, P Rogerson, and J Vena. Spatiotemporal GIS Analysis for Environmental Health Using Geospatial Lifelines. In *GEOMED '99: Proceedings of 2nd International Workshop on Geography and Medicine*, Paris, France, 1999. 42

[Men06] Yu Meng. *Improved Positioning of Land Vehicle in ITS Using Digital Map and Other Accessory Information*. PhD thesis, Hong Kong Polytechnic University, 2006. 21

[MFN+08] Gerasimos Marketos, Elias Frentzos, Irene Ntoutsi, Nikos Pelekis, Alessandra Raffaetà, and Yannis Theodoridis. Building Real-World Trajectory Warehouses. In *MobiDE '08: Proceedings of the 7th ACM International Workshop on Data Engineering for Wireless and Mobile Access*, 2008. 19

[MGA03] Mohamed F Mokbel, Thanaa M Ghanem, and Walid G Aref. Spatio-Temporal Access Methods. *IEEE Data Engineering Bulletin*, 26(2):40–49, 2003. 16

[MHLR10] Jonathan Muckell, Jeong-Hyon Hwang, Catherine T Lawson, and S S Ravi. Algorithms for Compressing GP Trajectory Data: an Empirical Evaluation. In *GIS '10: Proceedings of the 18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 402–405, 2010. 23

[MJEM02] Richard Mann, Allan D Jepson, and Thomas F El-Maraghi. Trajectory Segmentation Using Dynamic Programming. In *ICPR '02: Proceedings of the 16th International Conference on Pattern Recognition*, pages 331–334, 2002. 23

[MK03] Jussi Myllymaki and James H Kaufman. DynaMark: A Benchmark for Dynamic Spatial Indexing. In *MDM '03: Proceedings of the 4th International Conference on Mobile Data Management*, pages 92–105, 2003. 16

[MLF+08] Emiliano Miluzzo, Nicholas D Lane, Kristof Fodor, Ronald A Peterson, Hong Lu, Mirco Musolesi, Shane B Eisenman, Xiao Zheng, and Andrew T Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *SenSys '08: Proceedings of the 6th ACM Conference on Embedded Networked Sensor Systems*, pages 337–350, 2008. 31, 32

[MNPT05] Yannis Manolopoulos, Alexandros Nanopoulos, Apostolos N Papadopoulos, and Y Theodoridis. *R-Trees: Theory and Applications*. Springer, 1st edition, 2005. 15

[MPTG09] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. WhereNext: A Location Predictor on Trajectory Pattern Mining. In *KDD '09: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 637–646, 2009. 25

[MWH98] Spyros G Makridakis, Steven C Wheelwright, and Rob J Hyndman. *Forecasting: Methods and Applications*. WILEY, 1998. 82

[MXA+04] Mohamed F Mokbel, Xiaopeng Xiong, Walid G Aref, Susanne E Hambrusch, Sunil Prabhakar, and Moustafa A Hammad. PLACE: A Query Processor for Handling Real-time Spatio-temporal Data Streams. In *VLDB '04: Proceedings of the 30th International Conference on Very Large Data Bases*, pages 1377–1380, 2004. 13

[NDAM10] Long-Van Nguyen-Dinh, Walid G Aref, and Mohamed F Mokbel. Spatio-Temporal Access Methods: Part 2 (2003 - 2010). *IEEE Data Engineering Bulletin*, 33(2):46–55, 2010. xv, 16, 17

[NFM10] Jacinto C Nascimento, Mário A T Figueiredo, and Jorge S Marques. Trajectory Classification Using Switched Dynamical Hidden Markov Models. *IEEE Transactions on Image Processing*, 19(5):1338–1348, 2010. 26

[NK09] Paul Newson and John Krumm. Hidden Markov Map Matching Through Noise and Sparseness. In *GIS '09: Proceedings of the 17th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems*, pages 336–343, 2009. 21, 92

[NP06] Mirco Nanni and Dino Pedreschi. Time-focused clustering of trajectories of moving objects. *Journal of Intelligent Information Systems*, 27(3):267–289, 2006. 25

[NRRT04] Mirco Nanni, Alessandra Raffaetà, Chiara Renso, and Franco Turini. Deductive and Inductive Reasoning on Spatio-Temporal Data. *Applications of Declarative Programming and Knowledge Management*, pages 98–115, 2004. 28

[OGC06] Open Geospatial Consortium Inc., OpenGIS Implementation Specification for Geographic Information - Simple Feature Access, Part 2: SQL option, 2006. 40, 42

[ora07] Oracle Database, Semantic Technologies Developer's Guide, 11g Release 1, 2007. 45

[OS95] Gultekin Ozsoyoglu and Richard Thomas Snodgrass. Temporal and Real-Time Databases: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):513–532, 1995. 40

[PBKA08] Andrey Tietbohl Palma, Vania Bogorny, Bart Kuijpers, and Luis Otávio Alvares. A Clustering-based Approach for Discovering Interesting Places in Trajectories. In *SAC '08: Proceedings of the 2008 ACM Symposium on Applied Computing*, pages 863–868, 2008. 25, 29, 53, 76

[PFGT08] Nikos Pelekis, Elias Frentzos, Nikos Giatrakos, and Yannis Theodoridis. HERMES: Aggregative LBS via a Trajectory DB Engine. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1255–1258, 2008. 13

[Pfo02] Dieter Pfoser. Indexing the Trajectories of Moving Objects. *IEEE Data Engineering Bulletin*, 25(2):3–9, 2002. 16

[PHMA+01] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth. *ICDE '01: Proceedings of the 17th International Conference on Data Engineering*, pages 215–224, 2001. 25

[PHMA+04] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1424–1440, 2004. 25

[PJ99] Dieter Pfoser and Christian S Jensen. Capturing the Uncertainty of Moving-Object Representations. In *SSD '99: Proceedings of the 6th International Symposium on Advances in Spatial Databases*, pages 111–132, 1999. 14

[PJ05] Dieter Pfoser and Christian S Jensen. Trajectory Indexing Using Movement Constraints. *GeoInformatica*, 9(2):93–115, 2005. 15

[PJT00] Dieter Pfoser, Christian S Jensen, and Yannis Theodoridis. Novel Approaches in Query Processing for Moving Object Trajectories. In *VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases*, pages 395–406, 2000. 2, 15, 18

[PKLL02] Pranav Patel, Eamonn J Keogh, Jessica Lin, and Stefano Lonardi. Mining Motifs in Massive Time Series Databases. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining*, pages 370–377, 2002. 26

[PPS06] Michalis Potamias, Kostas Patroumpas, and Timos Sellis. Sampling Trajectory Streams with Spatiotemporal Criteria. In *SSDBM '06: Proceedings of the 18th International Conference on Scientific and Statistical Database Management*, pages 275–284. Ieee, 2006. 23, 67, 134

[PSS08] Christine Parent, Stefano Spaccapietra, and Heiner Stuckenschmidt. *Ontology Modularization*. Springer Verlag, 2008. 38, 46

[PSZ06] Christine Parent, Stefano Spaccapietra, and Esteban Zimanyi. *Conceptual Modeling for Traditional And Spatio-Temporal Applications*. Springer Verlag, 2006. 3, 28, 40

[PTVP06] Nikos Pelekis, Yannis Theodoridis, Spyros Vosinakis, and Themis Panayiotopoulos. HERMES - A Framework for Location-Based Data Management. In *EDBT '06: Proceedings of the 10th International Conference on Extending Database Technology*, pages 1130–1134, 2006. xv, 2, 13, 14

[PvJ06] Mindaugas Pelanis, Simonas Šaltenis, and Christian S Jensen. Indexing the past, present, and anticipated future positions of moving objects. *ACM Transactions on Database Systems*, 31(1):255–298, March 2006. 16

[QON07] Mohammed A Quddus, Washington Y Ochieng, and Robert B Noland. Current Map-Matching Algorithms for Transport Applications: State-Of-The Art and Future Research Directions. *Transportation Research Part C: Emerging Technologies*, 15(5):312–328, 2007. 20, 63, 66

[QOZN03] MohammedA. Quddus, WashingtonYotto Ochieng, Lin Zhao, and RobertB. Noland. A General Map Matching Algorithm for Transport Telematics Applications. *GPS Solutions*, 7(3):157–167, 2003. 21

[Rab90] Lawrence R Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Readings in speech recognition*, pages 267–296, 1990. 110

[RCC92] David A Randell, Zhan Cui, and Anthony G Cohn. A Spatial Logic based on Regions and Connection. In *KR '92: Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, pages 165–176, 1992. 28

[RDML05] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L Littman. Activity Recognition from Accelerometer Data. In *AAAI '05: Proceedings of the 20th National Conference on Artificial Intelligence*, pages 1541–1546, 2005. 31, 144, 151, 154, 157

[Rei05] Femke Reitsma. Modeling with the Semantic Web in the Geosciences. *IEEE Intelligent Systems*, 20(2):86–88, 2005. 42

[RMB+10] Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin, Mark H Hansen, and Mani B Srivastava. Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks*, 6(2), 2010. 30

[RSEN05] Slobodan Rasetic, Jörg Sander, James Elding, and Mario A Nascimento. A Trajectory Splitting Model for Efficient Spatio-Temporal Indexing. In *Proceedings of the 31st international conference on Very large data bases*, pages 934–945, 2005. 23

[RTO+10] Jose Antonio M R Rocha, Valéria Cesário Times, Gabriel Oliveira, Luis Otávio Alvares, and Vania Bogorny. DB-SMoT: A direction-based spatio-temporal clustering method. In *IS '10: Proceedings of the 5th IEEE International Conference on Intelligent Systems*, pages 114–119, 2010. 86

[RTR02] Alessandra Raffaetà, Franco Turini, and Chiara Renso. Enhancing GISs for Spatio-Temporal Reasoning. In *GIS '02: Proceedings of the 10th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems*, pages 42–48, 2002. 28

[SA96] Ramakrishnan Srikant and Rakesh Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. In *EDBT '96: Proceedings of the 5th International Conference on Extending Database Technology*, pages 3–17, 1996. 25

[SA09a] Nadine Schüssler and Kay W. Axhausen. Map-Matching of GPS Traces on High-Resolution Navigation Networks Using the Multiple Hypothesis Technique (MHT). Technical report, ETH Zurich, 2009. 21

[SA09b] Nadine Schüssler and Kay W. Axhausen. Processing GPS Raw Data Without Additional Information. *Transportation Research Record: Journal of the Transportation Research Board*, 8:28–36, 2009. 19, 30, 61, 133

[SCPV04] Stefano Spaccapietra, Nadine Cullot, Christine Parent, and Christelle Vangenot. On Spatial Ontologies. In *GEOINFO '04: Proceedings of the VI Brazilian Symposium on GeoInformatics*, 2004. 27

[SH97] Lloyd Smith and Mark Hall. Feature Subset Selection: a Correlation Based Filter Approach. In *ICONIP '97: Proceedings of the International Conference on Neural Information Processing*, pages 855–858, 1997. 155, 162

[SJK03] Laurynas Speičvcys, Christian S Jensen, and Augustas Kligys. Computational Data Modeling for Network-Constrained Moving Objects. In *GIS '03: Proceedings of the 11th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems*, pages 118–125, 2003. 14

[SJLL00] Simonas Saltenis, Christian S Jensen, Scott T Leutenegger, and Mario A Lopez. Indexing the Positions of Continuously Moving Objects. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 331–342, 2000. 2, 15

[Sot06] Anastasiya Sotnykova. *Semantic Validation in Spatio-Temporal Schema Integration*. PhD thesis, Swiss Federal Institute of Technology - Lausanne (EPFL), 2006. 28

[SPD+08] Stefano Spaccapietra, Christine Parent, Maria Luisa Damiani, Jose Antonio de Macedo, Fabio Porto, and Christelle Vangenot. A Conceptual View on Trajectories. *Data and Knowledge Engineering*, 65:126–146, 2008. 3, 6, 24, 28, 29, 36, 40, 50, 53, 73, 91, 144

[SRL09] Falko Schmid, Kai-Florian Richter, and Patrick Laube. Semantic Trajectory Compression. In *SSTD '09: Proceedings of the 11th International Symposium Advances in Spatial and Temporal Databases*, pages 411–416, 2009. 2, 108

[STK02] Ivo F Sbalzarinii, Julie Theriot, and Petros Koumoutsakos. Machine Learning for Biological Trajectory Classification Applications. In *Proceedings of the Summer Program 2002, Center for Turbulence Research*, pages 305–316, 2002. 26

[SYRS05] Roger D Santer, Yoshifumi Yamawaki, F Claire Rind, and Peter J Simmons. Motor Activity and Trajectory Control During Escape Jumping in the Locust Locusta Migratoria. *Journal of Comparative Physiology A*, 191(10):965–975, 2005. 53

[TG01] Ilias Tsoukatos and Dimitrios Gunopulos. Efficient Mining of Spatiotemporal Patterns. In *SSTD '01: Proceedings of 7th International Symposium on Spatial and Temporal Databases*, pages 425–442, 2001. 25

[The03] Yannis Theodoridis. Ten Benchmark Database Queries for Location-based Services. *Computer Journal*, 46(6):713–725, 2003. 18

[TOY80] S Tsuji, M Osada, and M Yachida. Tracking and Segmentation of Moving Objects in Dynamic Line Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:516–522, 1980. 23

[TPS03] Yufei Tao, Dimitris Papadias, and Jimeng Sun. The TPR*-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries. In *VLDB '03: Proceedings of the 29th International Conference on Very Large Data Bases*, pages 790–801, 2003. 16

[TVS96] Yannis Theodoridis, Michalis Vazirgiannis, and Timos K Sellis. Spatio-Temporal Indexing for Large Multimedia Applications. In *ICMCS '96: Proceedings of the 1996 International Conference on Multimedia Computing and Systems*, page 441, 1996. 2, 15

[TWHC04] Goce Trajcevski, Ouri Wolfson, Klaus Hinrichs, and Sam Chamberlain. Managing Uncertainty in Moving Objects Databases. *ACM Transactions on Database Systems*, 29(3):463–507, 2004. 14

[vKNEK08] Tim van Kasteren, Athanasios K Noulas, Gwenn Englebienne, and Ben J A Kröse. Accurate Activity Recognition in a Home Setting. In *UbiComp '08 Proceedings of the 10th international conference on Ubiquitous computing*, pages 1–9, 2008. 163

[VLL+10] La Vinh, Sungyoung Lee, Hung Le, Hung Ngo, Hyoung Kim, Manhyung Han, and Young-Koo Lee. Semi-Markov conditional random fields for accelerometer-based activity recognition. *Applied Intelligence*, pages 1–16, 2010. 31

[VW01] Michalis Vazirgiannis and Ouri Wolfson. A Spatiotemporal Model and Language for Moving Objects on Road Networks. In *SSTD '01: Proceedings of 7th International Symposium on Spatial and Temporal Databases*, pages 20–35, 2001. 14

[Wol02] Ouri Wolfson. Moving Objects Information Management: The Database Challenge. In *NGITS '02: Proceedings of the 5th International Workshop on Next Generation Information Technologies and Systems*, pages 75–89, 2002. 11

[WSX+99] Ouri Wolfson, Prasad Sistla, Bo Xu, Jutai Zhou, and Sam Chamberlain. DOMINO: Databases fOr MovINg Objects tracking. In *SIGMOD '99: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 547–549, New York, NY, USA, 1999. ACM. xv, 2, 12

[WXC00] Ouri Wolfson, Bo Xu, and Sam Chamberlain. Location Prediction and Queries for Tracking Moving Objects. In *ICDE '00: Proceedings of the 16th International Conference on Data Engineering*, page 687, 2000. 2

[WXCJ98] Ouri Wolfson, Bo Xu, Sam Chamberlain, and Liqin Jiang. Moving Objects Databases: Issues and Solutions. In *SSDBM '98: Proceedings of the 10th International Conference on Scientific and Statistical Database Management*, pages 111–122, 1998. 12

[WYM97] Wei Wang, Jiong Yang, and Richard R Muntz. STING: A Statistical Information Grid Approach to Spatial Data Mining. *VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 186–195, 1997. 25

[XDZ09] Kexin Xie, Ke Deng, and Xiaofang Zhou. From Trajectories to Activities: a Spatio-Temporal Join Approach. In *LBSN '09: Proceedings of the SIGSPATIAL ACM GIS 2009 International Workshop on Location Based Social Networks*, pages 25–32, 2009. 7, 29, 101, 110, 120

[XMA+04] Xiaopeng Xiong, Mohamed F Mokbel, Walid G Aref, Susanne E Hambrusch, and Sunil Prabhakar. Scalable Spatio-temporal Continuous Query Processing for Location-aware Services. In *SSDBM '04: Proceedings of the 16th International Conference on Scientific and Statistical Database Management*, page 317, 2004. 13

[Yan09] Zhixian Yan. Towards Semantic Trajectory Data Analysis: A Conceptual and Computational Approach. In *VLDB PhD Workshop*, 2009. 170

[Yan10] Zhixian Yan. Traj-ARIMA: A Spatial-time Series Model for Network-constrained Trajectory. In *IWCTS '10: Proceedings of the 3rd International ACM Workshop on Computational Transportation Science*, pages 11–16, 2010. 82, 93

[YCM+] Zhixian Yan, Dipanjan Chakraborty, Archan Misra, Hoyoung Jeung, and Karl Aberer. Semantic Activity Extraction Using Locomotive Signatures from Mobile Phones. In *EPFL Technical Report. In Submission.* 9, 168, 170

[YCP+11] Zhixian Yan, Dipanjan Chakraborty, Christine Parent, Stefano Spaccapietra, and Aberer Karl. SeMiTri: A Framework for Semantic Annotation of Heterogeneous Trajectories. In *EDBT '11: Proceedings of the 14th International Conference on Extending Database Technology*, pages 259–270, 2011. 7, 116, 124, 140, 144, 170

[YGK+11] Zhixian Yan, Nikos Giatrakos, Vangelis Katsikaros, Nikos Pelekis, and Yannis Theodoridis. Semantic-Aware Trajectory Computation over Streaming Movement Data. In *SSTD '11: Proceedings of the 12th International Symposium Advances in Spatial and Temporal Databases*, 2011. 8, 142, 144, 170

[YLJ10] Bin Yang, Hua Lu, and Christian S Jensen. Probabilistic Threshold k Nearest Neighbor Queries over Moving Objects in Symbolic Indoor Space. In *EDBT '10: Proceedings of the 13th International Conference on Extending Database Technology*, pages 335–346, 2010. 154

[YMPS08a] Zhixian Yan, Jose Macedo, Christine Parent, and Stefano Spaccapietra. Trajectory Ontologies. In *International Semantic Web Conference (Terra Cognita Workshop)*, 2008. 5, 54, 169

[YMPS08b] Zhixian Yan, Jose Macedo, Christine Parent, and Stefano Spaccapietra. Trajectory Ontologies and Queries. *Transactions in GIS*, 12(s1):75–91, 2008. 3, 5, 54, 169

[YPSC10] Zhixian Yan, Christine Parent, Stefano Spaccapietra, and Dipanjan Chakraborty. A Hybrid Model and Computing Platform for Spatio-Semantic Trajectories. In *ESWC '10: Proceedings of the 7th Extended Semantic Web Conference*, pages 60–75, 2010. 6, 19, 24, 54, 93, 133, 144, 170

[YSC+10] Zhixian Yan, Lazar Spremic, Dipanjan Chakraborty, Christine Parent, Stefano Spaccapietra, and Aberer Karl. Automatic Construction and Multi-level Visualization of Semantic Trajectories. In *GIS '10: Proceedings of the 18th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 524–525, 2010. 101, 102, 103, 116, 124, 170

[YTM08] Man Lung Yiu, Yufei Tao, and Nikos Mamoulis. The Bdual-Tree: Indexing Moving Objects by Space Filling Curves in the Dual Space. *VLDB Journal*, 17(3):379–400, 2008. 16

[Zak01] Mohammed Javeed Zaki. SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning*, 42(1/2):31–60, 2001. 25

[ZCL+10] Yu Zheng, Yukun Chen, Quannan Li, Xing Xie, and Wei-Ying Ma. Understanding Transportation Modes based on GPS Data for Web Applications. *ACM Transactions on the Web (TWEB)*, 4(1):1–36, January 2010. 30

[ZFL+07] Changqing Zhou, Dan Frankowski, Pamela J Ludford, Shashi Shekhar, and Loren G Terveen. Discovering Personally Meaningful Places: an Interactive Clustering Approach. *ACM Transactions on Information Systems*, 25(3):12, 2007. 29

[ZG02] Jingxiong Zhang and Michael F Goodchild. *Uncertainty in Geographical Information.* CRC, 1 edition, 2002. 14, 19, 61

[ZKS09] Max Zimmermann, Thomas Kirste, and Myra Spiliopoulou. Finding Stops in Error-Prone Trajectories of Moving Objects with Time-Based Clustering. In *Intelligent Interactive Assistance and Mobile Multimedia Computing*, pages 275–286. 2009. 76

[ZLWX08] Yu Zheng, Like Liu, Longhao Wang, and Xing Xie. Learning Transportation Mode from Raw GPS Data for Geographic Applications on the Web. In *WWW '08: Proceedings of the 17th International Conference on World Wide Web*, number 49, page 247. ACM Press, 2008. 24

[ZRL96] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *SIGMOD '96: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, 1996. 25

[ZSI02] Hongjun Zhu, Jianwen Su, and Oscar H Ibarra. Trajectory Queries and Octagons in Moving Object Databases. In *CIKM '02: Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management*, pages 413–421, 2002. 2, 15, 18

[ZZXM09a] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. Mining Correlation Between Locations Using Human Location History. In *GIS '09: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 472—-475, November 2009. 53

[ZZXM09b] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. Mining Interesting Locations and Travel Sequences from GPS Trajectories. In *WWW '09: Proceedings of the 18th International Conference on World Wide Web*, pages 791–800, 2009. 24

[ZZXY10] Vincent W Zheng, Yu Zheng, Xing Xie, and Qiang Yang. Collaborative location and activity recommendations with GPS history data. *WWW '10: Proceedings of the 19th International Conference on World Wide Web*, page 1029, 2010. 30

Curriculum Vitae

Zhixian YAN
Chemin de Veilloud 10,
Ecublens
CH-1024
Switzerland

URL: `http://people.epfl.ch/zhixian.yan`
eMail: zhixian.yan@epfl.ch
zhixian.yan@gmail.com
Date of Birth: October 7, 1982
Nationality: Chinese

EDUCATIONAL BACKGROUND

- *Ph.D. in Computer Science* (2007.10–2011.08)
  Swiss Federal Institute of Technology - Lausanne (EPFL),
  Lausanne, Switzerland

- *M.Sc. in Computer Science* (2003.09–2006.07)
  Institute of Computing Technology (ICT)
  Chinese Academy of Sciences (CAS), Beijing, China

- *B.Sc. in Information System* (1999.09–2003.07)
  Chongqing University of Posts and Telecommunications (CQUPT),
  Chongqing, China (in University's Top-student Class)

PROFESSIONAL EXPERIENCES

- *Junior Researcher at EPFL, Switzerland* (2007.10–2011.08)
  Database Laboratory (LBD) & Distributed Information Systems Laboratory (LSIR)
  Topics: "Mobility data analysis: data modeling, computing and mining", "Reality mining, smartphone based people centric sensing"

- *Research Visitor at University of Piraeus, Greece* (2010.09–2010.10)
  Information Management Lab (InfoLab)
  Topics: "Online and distributed trajectory data computing"

- *Junior Researcher at Innsbruck University, Austria* (2006.08–2007.10)
  Semantic Execution Environment (SEE), Digital Enterprise Research Institute (DERI)
  Topics: "Business processes management towards semantic web service, Semantic web, Social web"

- *Research Intern at IBM China Lab, China* (2006.04–2006.07)
  Information and Knowledge Group, IBM China Research Laboratory
  Topics: "Ontology Management System and Benchmark Platform"

BIBLIOGRAPHY

- *Research Assistant at ICT-CAS, China* (2004.07–2006.05)
  Key Lab of Intelligent Information Processing,
  Institute of Computing Technology, Chinese Academy of Sciences
  Topics: "Knowledge and intelligent systems, Semantic web"

- *Engineer Intern at Sohu, China* (2005.06–2005.11)
  Wireless and SMS (Short Message Service) Group,
  Sohu.com Inc. (China's premier online brand)
  Topics: "J2EE implementation on social networks & Internet forum"

SELECTED PUBLICATIONS

(1) Published Papers Related to This Thesis

[*SSTD'11*] Zhixian Yan, Nikos Giatrakos, Vangelis Katsikaros, Nikos Pelekis, Yannis Theodoridis. *SeTraStream: Semantic-Aware Trajectory Computation over Streaming Movement Data.* the 12^{th} Intl. Symposium on Spatial and Temporal Databases (SSTD2011), Minneapolis, MN, USA, August 2011

[*EDBT'11*] Zhixian Yan, Dipanjan Chakraborty, Christine Parent, Stefano Spaccapietra, Karl Aberer. *SeMiTri: A Framework for Semantic Annotation of Heterogeneous Trajectories*, the 14^{th} Intl. Conf. on Extending Database Technology (EDBT2011), Uppsala, Sweden, March 2011

[*GIS'10*] Zhixian Yan, Lazar Spremic, Dipanjan Chakraborty, Christine Parent, Stefano Spaccapietra, Karl Aberer. *Automatic Construction and Multi-level Visualization of Semantic Trajectories*, the 18^{th} ACM SIGSPATIAL Intl. Conf. on Advances in Geographic Information Systems (GIS2010), San Jose, CA, USA, November 2010

[*CTW'10*] Zhixian Yan. *Traj-ARIMA: A Spatial-Time Series Model for Network-Constrained Trajectories*, the 3^{rd} ACM SIGSPATIAL Intl. Workshop on Computational Transportation Science (IWCTS2010), San Jose, CA, USA, November 2010

[*ESWC'10*] Zhixian Yan, Christine Parent, Stefano Spaccapietra, Dipanjan Chakraborty. *A Hybrid Model and Computing Platform for Spatio-Semantic Trajectories.* the 7^{th} Extended Semantic Web Conf. (ESWC2010), Heraklion, Greece, May 2010

[*VLDB'09*] Zhixian Yan, Stefano Spaccapietra. *Towards Semantic Trajectory Data Analysis: A Conceptual and Computational Approach*, 35^{th} Intl. Conf. on Very Large Data Bases (VLDB2009), PhD Workshop, Lyon, France, August 2009

[*TransGIS'08*] Zhixian Yan, Jose Macedo, Christine Parent, Stefano Spaccapietra. *Trajectory Ontologies and Queries*, Volume 12 Issue s1, Pages 75-91, Transactions in GIS, Wiley Blackwell, December 2008

[*ISWC'08*] Zhixian Yan, Jose Macedo, Christine Parent, Stefano Spaccapietra. *Trajectory Ontologies*, Terra Cognita Workshop, the 7^{th} Intl. Semantic Web Conf. (ISWC2008), Karlsruhe, Germany, October 2008

(2) Papers in Submission Related to This Thesis

Zhixian Yan, Dipanjan Chakraborty, Archan Misra, Hoyoung Jeung, Karl Aberer. *Semantic Activity Extraction Using Locomotive Signatures from Mobile Phones.*

Zhixian Yan, Dipanjan Chakraborty, Christine Parent, Stefano Spaccapietra, Karl Aberer. *Semantic Trajectories: Mobility Data Computation and Annotation.*

Christine Parent, Stefano Spaccapietra, Chiara Renso, Gennady Andrienko, Natalya Andrienko, Vania Bogorny, Maria Luisa Damiani, Aris Gkoulalas-Divanis, Jose Macedo, Nikos Pelekis, Yannis Theodoridis, Zhixian Yan.
Survey on Semantic Trajectories Modeling, Computing, and Analysis.

(3) Publications not Directly Related to This Thesis

[*SMC'08*] *Integrating Social Tagging Data*
Ying Ding, Sin-Jae Kang, Ioan Toma, Michael Fried, Zhixian Yan
IEEE International Conference on Systems, Man and Cybernetics (SMC), Singapore, October 2008

[*SESA'08*] *Semantically Enabled Service Oriented Architecture (SESA) Middleware*
Matthew Moran, Tomas Vitvar, Zhixian Yan, Michal Zaremba, in book "Implementing Semantic Web Services - The SESA Framework", 2008, Springer

[*ISWC'07*] Zhixian Yan, Emilia Cimpian, Michal Zaremba, Ying Ding
SemBiz BPMO: Business Process Modeling Ontology
6^{th} International Semantic Web Conference (ISWC), Busan, Korea, November 2007

[*ICWS'07*] Zhixian Yan, Emilia Cimpian, Michal Zaremba, Manuel Mazzara,
BPMO: Semantic Business Process Modeling and WSMO Extension
IEEE International Conference on Web Services (ICWS), Salt Lake City, Utah, USA, July, 2007

[*BPSC'07*] Zhixian Yan, Emilia Cimpian, Michal Zaremba, Manuel Mazzara
Business Process Modeling: Classification and Perspective
International Conference on Business Process & Services Computing (BPSC), Leipzig, Germany, September 2007

[*WWW/Internet'07*] Kathrin Prantner, Ying Ding, Michael Luger, Zhixian Yan
Tourism Ontology & Semantic Management System: State-of-the-Arts Analysis
IADIS International Conference on WWW/Internet 2007, Vila Real, Portugal, October 2007

[*KES'07*] Zhixian Yan, Emilia Cimpian, Michal Zaremba, Manuel Mazzara
Towards a Domain Oriented and Independent Semantic Search Model
11^{th} Intl. Conf. on Knowledge-Based and Intelligent Information & Engineering Systems (KES), Vietri sul Mare, Italy, September 2007

[*AWIC'07*] Zhixian Yan, Franǫis Scharffe, Ying Ding
Semantic Search on Cross-Media Cultural Archives
5^{th} Atlantic Web Intelligence Conference (AWIC), Fontainebleau, France, June 2007