



HAL
open science

A Framework for Automatic Annotation of Semantic Trajectories

Tales Paiva Nogueira

► **To cite this version:**

Tales Paiva Nogueira. A Framework for Automatic Annotation of Semantic Trajectories. Web. Université Grenoble Alpes, 2017. English. NNT : 2017GREAM004 . tel-01490179v2

HAL Id: tel-01490179

<https://hal.science/tel-01490179v2>

Submitted on 11 Jan 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : **Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Tales PAIVA NOGUEIRA

Thèse dirigée par **Mr Hervé Martin**

préparée au sein du **Laboratoire d'Informatique de Grenoble (LIG)**
et de l'**École Doctorale Mathématiques, Sciences et Technologies de**
l'**Information, Informatique (EDMSTII)**

A Framework for Automatic Annotation of Semantic Trajectories

Thèse soutenue publiquement le **30 janvier 2017**,
devant le jury composé de :

M Thomas Devogele

Professeur, Université de Tours, Rapporteur

M Alain Bouju

Maître de Conférences HDR, Université de La Rochelle, Rapporteur

M Hervé Martin

Professeur, Université Grenoble Alpes, Directeur de thèse

M Ahmed Lbath

Professeur, Université Grenoble Alpes, Président

Mme Rossana Maria de Castro Andrade

Professeur, Federal University of Ceará, Brésil, Examineur



A Framework for Automatic Annotation of Semantic Trajectories

Abstract: Location data is ubiquitous in many aspects of our lives. We are witnessing an increasing usage of this kind of data by a variety of applications. As a consequence, information systems are required to deal with large datasets containing raw data in order to build high level abstractions. Semantic Web technologies offers powerful representation tools for pervasive applications. The convergence of location-based services and Semantic Web standards allows an easier interlinking and annotation of trajectories.

In this thesis, we focus on modeling mobile object trajectories in the context of the Semantic Web. First, we propose an ontology that allows the representation of generic episodes. Our model also handles contextual elements that may be related to trajectories. Second, we propose a framework containing three algorithms for automatic annotation of trajectories. The first one detects moves, stops, and noisy data; the second one is able to compress generic time series and create episodes that resumes the evolution of trajectory characteristics; the third one exploits the linked data cloud to annotate trajectories with geographic elements from OpenStreetMap.

As results of this thesis, we have a new ontology that can represent spatiotemporal phenomena at different levels of granularity. Moreover, our framework offers three novel algorithms for trajectory annotation. The move-stop-noise detection method is able to deal with irregularly sampled traces and do not depend on external data of the underlying geography; our time series compression method is able to find values that summarize a series at the same time that too small segments are avoided; and our spatial annotation algorithm explores linked data and the relationships among concepts to find relevant types of spatial features to describe the environment where the trajectory took place.

Keywords: semantic web, trajectory analysis, ontologies, trajectory segmentation

Un Framework pour l'Annotation Automatique des Trajectoires Sémantiques

Résumé: Les données de localisation sont présentes dans plusieurs aspects de notre vie. Nous assistons à une utilisation croissante de ce type de données par une grande variété d'applications. En conséquence, les systèmes d'information doivent construire des abstractions de haut niveau pour traiter de grands volumes de données brutes. La convergence des services de localisation et des standards de la Web sémantique peut faciliter les tâches d'interconnexion et d'annotation des trajectoires.

Dans cette thèse, nous nous concentrons sur la modélisation de trajectoires dans le contexte du Web sémantique. Nous proposons une ontologie pour représenter des épisodes génériques. Notre modèle couvre aussi des éléments contextuels qui peuvent être liés à des trajectoires. Nous proposons un framework basé sur trois algorithmes d'annotation des trajectoires. Le premier détecte les mouvements, les arrêts et les données manquantes; le deuxième est utilisé pour compresser des séries temporelles et créer des épisodes caractéristiques de l'évolution de la trajectoire; le troisième exploite les données liées pour annoter les trajectoires en s'appuyant notamment sur OpenStreetMap.

Une de nos contributions est une nouvelle ontologie qui peut représenter des phénomènes spatiotemporels avec différents niveaux de granularité. Notre méthode de détection de mouvement-arrêt-bruit est capable de traiter des traces échantillonnées irrégulièrement et ne dépend pas des données externes; notre méthode de compression des séries temporelles est capable de construire un résumé en évitant les segments trop courts; notre algorithme d'annotation spatiale exploite des données liées et des relations entre concepts pour extraire des types pertinents d'entités spatiales afin de décrire l'environnement où la trajectoire a eu lieu.

Mots-clés: web sémantique, analyse des trajectoires, ontologies, segmentation des trajectoires

Acknowledgments

I'd like to thank the jury that participated in my thesis defense. To the reporters Alain Bouju and Thomas Devogele, whose reports have added a great value to the final version of the manuscript. To Ahmed Lbath, that accepted to be an examiner and president of the jury. To Rossana Andrade, who I specially thank for participating in the jury and also because most of my professional career was developed in the research group directed by her in Brazil, a place that also gave me the opportunity to make this thesis.

I'd like to give special thanks to Hervé Martin, my supervisor, for his patience even in the most difficult moments, research guidance, and help with administrative formalities. Thank you very much for pushing me beyond what I thought it was my limits.

Thanks to the French Ministry of Higher Education and Research (Ministère de l'Enseignement Supérieur et de la Recherche de la France – MESR) and the Brazilian National Council for Scientific and Technological Development (Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq) for financing this research.

A special mention goes to Reinaldo and Carina. I will be eternally thankful for you two for welcoming us in Grenoble and helping with the most challenging part of moving to a new place, which is even more difficult when we don't speak the local language very well.

For all staff of the Grenoble-Alpes University and Grenoble Informatics Laboratory, thank you to provide the best possible environment for the development of this work, specially for the STEamer team represented by Paule-Annick, Marlène, Danielle, Jérôme, Philippe, Sylvain and the fellow colleagues Rafaella, Mouna, Betül, Anthony, Cecile, David, André, Thiago, Camille, Jacques, Aline et Clément.

I would also like to thank all the awesome people that I knew during my stay in Grenoble that is not directly related to this thesis, but that surely helped by sharing good moments.

I have no words to express how grateful I am to my wife, Josy, who accepted to embark in this journey with me and have taught me how to be a better person everyday. To my family in Brazil, my mother Joice, my late father Hélio, and my sisters Aline and Danielly, thanks for supporting me. I wouldn't get here if it wasn't for you.

Contents

1	Introduction	1
1.1	Core Issues	4
1.2	Contributions	7
1.3	Thesis outline	9
2	State-of-the-Art	11
2.1	Semantic Web and Trajectory Models	12
2.1.1	Semantic Web	12
2.1.1.1	Geospatial Semantic Web	19
2.1.1.2	Temporal Modeling	22
2.1.2	Trajectory Models	24
2.1.2.1	Semantic Trajectory Ontologies	25
2.1.2.2	Other trajectory models	32
2.2	Trajectory Annotation	35
2.2.1	Stops identification	39
2.2.2	Time Series Segmentation	41
2.2.3	Spatial Annotation	45
3	Contributions	49
3.1	STEP Ontology	50
3.1.1	Requirements	50
3.1.2	The QualiTraj ontology	54
3.1.3	STEP: Semantic Trajectory Episodes	56
3.1.4	Updates on the STEP ontology	66
3.1.5	Structuring Trajectories with STEP	68
3.1.6	Querying Episodes	70
3.1.7	Evaluation	75
3.2	STEP Framework	81
3.3	Automatic Annotation Algorithms	85
3.3.1	MSN: Move-Stop-Noise Classification	85
3.3.1.1	Exploratory Data Analysis	86
3.3.1.2	Outlier Labeling Rule	91
3.3.1.3	The MSN algorithm	93
3.3.1.4	Evaluation	99

3.3.2 ANBA: A daptive N atural B reaks A pproximation	102
3.3.2.1 Fisher-Jenks Natural Breaks	103
3.3.2.2 The ANBA algorithm	104
3.3.2.3 Evaluation	110
3.3.3 ISA: I ntersection-based S patial A notation	117
4 Conclusions and Perspectives	131
4.1 Conclusions	131
4.2 Future Work	134
Publications	137
Bibliography	139
Appendix	163
Appendix A: MSN	163
Appendix B: ANBA	177
Appendix C: ISA	184

List of Figures

1.1	Examples of applications that deal with sportive trajectories	2
1.2	Overview of the data flow from raw trajectory data to representations of movement in higher abstractions levels.	4
2.1	The Semantic Web stack (adapted from [Nowack 2009])	15
2.2	Linked Open Data cloud as of August 2014 Schmachtenberg et al. [2014]	20
2.3	Excerpt of the GeoSPARQL vocabulary. The red dotted line represents a <code>hasGeometry</code> relationship, all other links represent subclassing relationships.	21
2.4	Allen’s Interval Algebra.	23
2.5	Conceptual view of trajectories [Spaccapietra et al. 2008]	26
2.6	Example of trajectory ontologies for an application in the traffic domain [Yan et al. 2008]	27
2.7	The Simple Event Model ontology [van Hage et al. 2009b]	29
2.8	The trajectory design pattern proposed by Hu et al. [2013]	30
2.9	The Baquara ² ontology overview [Fileto et al. 2015b]	32
2.10	The CONceptual model of Semantic TrAJecTories (CONSTAnT) conceptual model for semantic trajectories [Bogorny et al. 2014]	34
2.11	Overview of a trajectory enriching process [Marketos et al. 2013]	37
2.12	Trajectory computing platform [Yan et al. 2012]	38
2.13	Example of Piecewise Aggregated Approximation of a time series	43
2.14	Different approaches of Piecewise Linear Segmentation	44
3.1	Trajectory of a runner. The person started the workout in a park, run next to a river, and then returned to the park.	51
3.2	An example of multi-layered representation of trajectory and context based on episodes. Features of Interest are represented in the y axis and time is in x axis.	53
3.3	The QualiTraj ontology	55
3.4	Example of abstract speed evolution of a trajectory.	56
3.5	First version of the Semantic Trajectory Episodes (STEP) ontology [Nogueira and Martin 2015]	58

3.6	Part of the Simple Features ontology that defines Geometry subclasses.	61
3.7	Part of the OWL:Time ontology that defines TemporalEntity subclasses.	62
3.8	Overview of the STEP ontology. Concepts from other ontologies are indicated by their namespaces and dotted rectangles, diamond-shaped nodes are datatypes, notes represent class restrictions.	65
3.9	An excerpt example of STEP ontology instances. The Feature of Interest in this case is the topological relations of the trajectory and nearby geographic features.	69
3.10	Overview of the STEP framework. The colored blocks were implemented during this thesis.	82
3.11	For each three points of a trajectory, we calculate the angle formed by them to get the turning angle.	83
3.12	An example trajectory.	87
3.13	Speed, distance, and duration between points of the example trajectory.	88
3.14	Two cases demonstrating the better performance of Spearman correlation over Pearson correlation in (a) non-linearity of relationship and (b) presence of outliers.	89
3.15	The density plot of distances in (a) and the three trajectory points with long distances in (b)	94
3.16	Time series of turning angles. There are no group of consecutive angles with less than 45 degrees, therefore no direction outliers are identified.	96
3.17	Distribution plot of slightly “jittered” durations where two outliers are identified.	97
3.18	Difference of speed data before and after natural logarithmic transformation. Also, the outlier threshold is shown in (b)	98
3.19	Modified z-scores of speed and duration with their thresholds.	99
3.20	Final classification of the MSN algorithm.	100
3.21	Stages of the Adaptive Natural Breaks Approximation (ANBA) process from the raw data to the approximation in an example speed time series extracted from a trajectory.	109

3.22	Differences in level of detail with two and three approximate values	111
3.23	Methods considered in the evaluation. paa : Piecewise Aggregate Approximation, tdi : Top-Down with Interpolation, tdr : Top-Down with Regression, bui : Bottom-Up with Interpolation, bur : Bottom-Up with Regression, swi : Sliding Window with Interpolation, swr : Sliding Window with Regression, anba-n , anba-gvf : Adaptive Natural Breaks Approximation with number of classes and Goodness of Variance Fit (GVF), respectively. Root Mean Squared Error (RMSE) is in meters per seconds and running time is in seconds.	115
3.24	Data flow of the intersection-based spatial annotation algorithm with OSM features.	122
3.25	Example of polygons retrieved by queries on Overpass API	127
3.26	Some example trajectories along with the features selected by the Intersection-based Spatial Annotation (ISA) algorithm and the corresponding LinkedGeoData concepts.	129

List of Tables

3.1 Comparative table of trajectory models	76
3.2 Median of Spearman correlations among movement attributes of 2226 trajectories	90
3.3 Mean of Spearman correlations among movement attributes of 2226 trajectories	90
3.4 Standard deviation of Spearman correlations among movement attributes of 2226 trajectories	90
3.5 Comparison of stop detection algorithms	101
3.6 Dataset summary	112
3.7 Mean value of metrics for the testing dataset.	116
3.8 Feature tags, duration ratio, and distance ratios regarding the example trajectory.	127

List of Listings

3.1 SPARQL query to detect a speed pattern	71
3.2 Query for retrieving the points of the trajectory during which a park was nearby.	72
3.3 Equivalent queries in Baquara ² (left) and STEP (right)	74
3.4 Equivalent queries in CONSTAnT (top) and STEP (bottom)	75
3.5 Java code for getting position updates only if the moving object has moved for at least 10 meters.	86
3.6 SPARQL query that finds all OSN concepts that are related to the <i>leisure</i> key. Additionally, only related terms that have an exact match in LGD ontology are returned.	123
3.7 Example of Overpass API query that retrieve all ways and relations intersecting the geometry specified in the <i>poly</i> field. The tags are selected by a regular expression.	124
3.8 SPARQL query to retrieve all subjects, types, geometries and labels of entities that intersect the area around 1 meter of a given line (the trajectory) and have one of the classes inside the “IN” filter.	125
3.9 Except of RDF/XML file automatically generated by the STEP framework.	128

List of Acronyms

ADCM	Absolute Distance from the Class Median
ANBA	Adaptive Natural Breaks Approximation
APCA	Adaptive Piecewise Constant Approximation
AV	Approximate Values
CAV	Candidate Approximate Value
CB-SMoT	Clustering-Based Stops and Moves of Trajectories
CONSTAnT	CONceptual model of Semantic TrAJecTories
CRS	Coordinate Reference System
CSS	Cascading Style Sheets
DE-9IM	Dimensionally Extended nine-Intersection Model
DOM	Document Object Model
FOAF	Friend of a Friend
GPS	Global Positioning System
GPX	GPS Exchange Format
GUC	Generic Use Cases
GVF	Goodness of Variance Fit
HMM	Hidden Markov Model
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
ISA	Intersection-based Spatial Annotation
LGD	LinkedGeoData

LoI Line of Interest

LOSM Linked Open Street Map

MAD Median Absolute Deviation

MADS Modeling of Application Data with Spatio-temporal features

MSN Stop, Move, and Noise

OGC Open Geospatial Consortium

OSM OpenStreetMap

OSN Open Semantic Network

OWL Web Ontology Language

PAA Piecewise Aggregated Approximation

PIE Point-of-Interest Extraction

PLS Piecewise Linear Segmentation

PoI Point of Interest

RCC Region Connection Calculus

RDF Resource Description Framework

RDFS RDF Schema

RIF Rule Interchange Format

RMSE Root Mean Squared Error

RoI Region of Interest

SDAM Squared Deviations from the Array Mean

SDCM Squared Deviations from the Class Mean

SEM Simple Event Model

SKOS Simple Knowledge Organization System

SMoT Stops and Moves of Trajectories

SPARQL SPARQL Protocol and RDF Query Language

SPIN SPARQL Inferencing Notation

SQL Structured Query Language

SRTM Shuttle Radar Topography Mission

STEP Semantic Trajectory Episodes

SWRL Semantic Web Rule Language

URI Universal Resource Identifier

URL Uniform Resource Locator

W3C World Wide Web Consortium

WGS84 World Geodetic System 1984

Introduction

Contents

1.1 Core Issues	4
1.2 Contributions	7
1.3 Thesis outline	9

Location data is ubiquitous in many aspects of our digital lives. We are witnessing the production of increasing volumes of this kind of data every day by a variety of applications and devices usually equipped with Global Positioning System (GPS) sensors. Additionally, new devices are being developed with the ability of capturing our traces. The interest in movement analysis crosses many application fields such as behavioral ecology, mobility of people using various means of transportation, sports analysis, surveillance, among others. Figure 1.1 shows some examples of such applications. As a consequence of this increasing flow of data, information systems are frequently required to deal with large datasets that in some cases contains raw and noisy data in order to build high level abstractions to be later analyzed and organized for delivering useful functionalities to end users.

When location data (generally latitude and longitude in a given Coordinate Reference System (CRS)) is constantly collected and associated with the instant in time when it was sampled, we have a trajectory. From photos to sports activities, the presence of this kind of data at different levels of granularity is constantly growing. Therefore, there is a need for novel data structures,

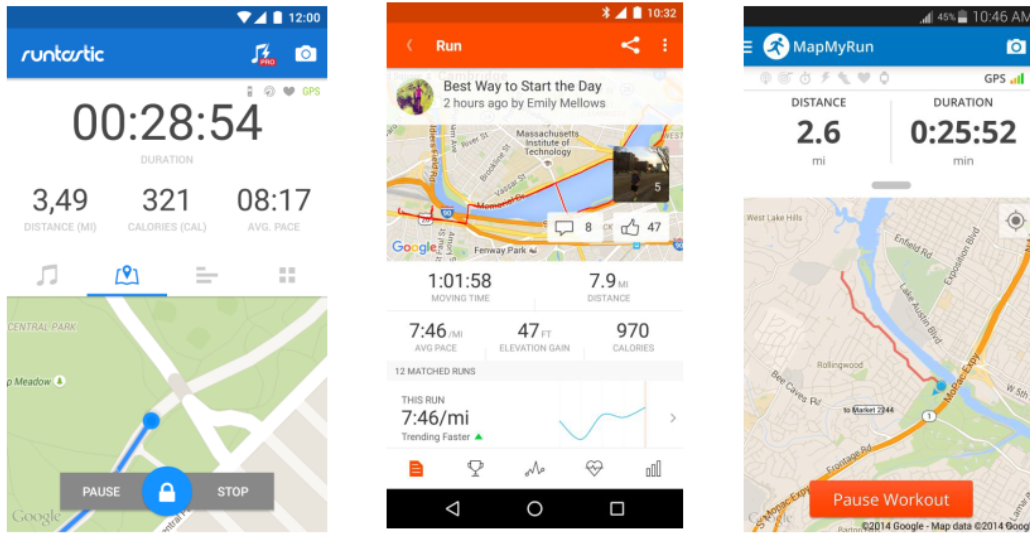


Figure 1.1: Examples of applications that deal with sportive trajectories

algorithms and techniques to deal with the spatiotemporal aspect of this information.

The variety and complexity of spatiotemporal data have led to the establishment of an interdisciplinary field called Computational Movement Analysis devoted to studying the development and application of computational techniques for capturing, processing, managing, structuring, and ultimately analyzing data describing movement phenomena, both in geographic and abstract spaces, aiming for a better understanding of the processes governing that movement [Laube 2014].

Trajectory data mining shows itself as an important area of research that is usually subdivided into more specific procedures that are required to transform raw location data into useful knowledge about trajectories. Many authors have provided definitions of what can constitute a trajectory data mining framework. Laube [2014] delimits movement mining into four groups: segmentation and filtering, similarity and clustering, movement patterns, exploratory analysis and visualization. Zheng [2015] describes a more detailed framework containing elements such as preprocessing, indexing and retrieval, uncertainty,

pattern mining, outlier/anomaly detection, classification, as well as ways of modeling trajectories.

Another field that has experience growth in the last few years is the Semantic Web. The increasing adoption of semantic web technologies has changed the way information is made available on the web. As we can see, the web of documents is going toward the web of data, where machines are able to understand and reason about the connections among the entities of a dataset and also their interactions with entities from different datasets, thus enabling the development of smarter applications.

Semantic Web technologies offers powerful representation facilities and reasoning techniques for pervasive applications and are rapidly gaining attention towards facing a range of issues such as data and knowledge modeling, querying, reasoning, service discovery, privacy and provenance [Ye et al. 2015]. The growth of geolocation capabilities embedded in smart phones, smart watches, tablets, and even glasses has facilitated the development of intelligent mobile systems for the acquisition of data. The convergence of these technologies with Semantic Web standards allows the easy acquisition, interlinking, and annotation of information about trajectories. The management, modeling, and analysis of such data provide many challenges related to its integration with existing systems. Therefore, the multidimensional and multifaceted aspects of geolocated data should be taken into account due to their rich semantics.

In the context of trajectory data mining, this thesis is positioned in the earlier stages of the process. More specifically, we deal with issues in the modeling, preprocessing and annotation stages that are vital for further development and smart applications by exploring statistic properties of trajectories and using the Semantic Web infrastructure and standards. Figure 1.2 shows the general process explored in this thesis.

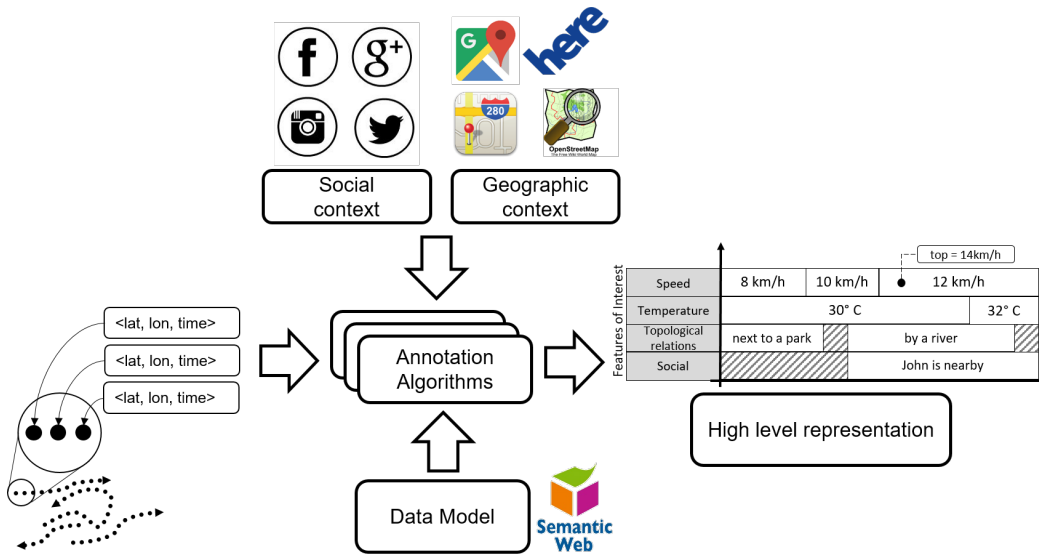


Figure 1.2: Overview of the data flow from raw trajectory data to representations of movement in higher abstractions levels.

1.1 Core Issues

A vast amount of work can be found in the literature proposing new approaches to bridge the gap between trajectory representation and Semantic Web technologies, mainly regarding the representation with ontologies [Baglioni et al. 2008, Camossi et al. 2013, Hu et al. 2013, Parent et al. 2013, van Hage et al. 2009a, Wannous et al. 2013, Yan et al. 2008; 2011]. However, as noticed by Fileto et al. [2015a], the development and the evaluation of efficient and effective methods for enriching movement data with ontologies and knowledge bases such as Linked Open Data collections is still an open research challenge.

Structuring trajectories into periods of stops and moves has been proved to be a fundamental task [Spaccapietra et al. 2008]. This is even more noticeable as research in trajectory analysis has been mostly focused on the annotation of stops rather than on the parts of trajectories where movement actually happens. However, different criteria can be used to segment trajectories [Buchin et al. 2011, Alewijnse et al. 2014], expanding the possibilities of structuring moving object traces beyond the stop-move model. In fact, viewing a trajec-

tory as a sequence of moves and stops can be the first step towards a more complex model of trajectories that some applications require.

Regarding the stop-move segmentation problem, we observe that the major methods for identifying these events rely on the assumption that trajectories are sampled at regular intervals of time. This assumption allows the application of algorithms based on clustering methods in order to identify points that have been recorded at near locations, and therefore may characterize a stop. However, this assumption may not be true in many applications due to a variety of reasons, such as periods of **GPS** failure, noisy measurements, the sampling strategy of the device or application or even pre-processing steps where the original trajectory is modified to contain less points in order to have a smoother trace and save storage.

This concern is supported if we take the work of [Andrienko et al. \[2008\]](#), for instance, where the authors tries to define what is a trajectory and how it can be observed. Diverse ways of movement recording are listed, namely *time-based*, when positions are recorded at regular intervals of time, which is the assumption considered by the majority of methods; *change-based*, when positions are recorded only when the object moves; *location-based*, when the location is collected only if the object approaches a specific location, e.g. near a sensor; *event-based*, when the moving object performs a specific action, e.g. making a call; and various combinations of these methods. While most state-of-the-art methods of stop detection deal mainly with the *time-based* recording strategy, we observe that some applications may prefer to store trajectories following the *change-based* approach. Another possibility is that these applications may make modifications to the captured data to eliminate near points. In this scenario, algorithms that rely on clustering near points for stay-point detection would fail.

Once periods of movement, stop and noise are identified, a usual step in a

data mining workflow consists in aggregating data about some characteristics of a trajectory. This process is crucial for reducing the amount of data to analyze and being able to build representations in higher abstraction levels. Due to its temporal essence, many trajectory attributes can be represented as time series, which a vast body of research has been established long before the trajectory analysis field. Time series compression techniques can help in building smaller representations of some trajectories characteristics. More specifically, this technique can be helpful to find homogeneous periods along trajectories where a single value can summarize a group of measurements without decreasing too much the correlation between the original and compressed versions of the data.

The issue that we have identified in these algorithms is that many of them does not yield constant approximations, instead they return a set of lines that best fit parts of the original time series according to a maximum error parameter. Additionally, considering a directed graph model as it is the case with Semantic Web ontology instances, it is possible to benefit of storage savings if a single tuple can be reused several times, i.e. if a tuple is linked to more than one other tuple. Choosing a set of values that approximates as many points as possible for the entire time series instead of just segments of it is then a desirable characteristic of a compression algorithm that we explore in this thesis.

Another point that needs further attention is about linking trajectories to the underlying geography where it happens. This is important to enable a better explanation about the behavior of a moving object. As noticed by [Laube \[2015\]](#), “one challenge for understanding movement in its context lies in the difficulty of accessing the comprehensive data required for such studies”. For this problem, Semantic Web technologies may help as they make easier data integration not only by shared vocabularies, but also through federated que-

ries that are capable of retrieving results from different database endpoints. We explore the interlinking of vocabularies and datasets in a scenario of trajectory enrichment with geographic features around it. For this we have used ontologies that have been built around the OpenStreetMap project for aiding the annotation process.

Considering the problems of structuring trajectory data (segmentation problem) and linking trajectories to context information, we identify that there is still a gap in research for representing structural properties of trajectories and contextual data. On the other hand, structuring spatiotemporal data in the form of episodes has been proven to be a natural approach [Mountain and Raper 2001, Parent et al. 2013, Pelekis and Theodoridis 2014]. For this reason, we conceived a model that is flexible enough to represent different aspects of traces and also enables the linkage of trajectories and the underlying geography. The ability of representing spatiotemporal data in different granularity levels and with varying temporal and spatial extents was a core issue that has been addressed in this thesis.

1.2 Contributions

In this thesis, we describe a series of contributions for the automated annotation of trajectories built around an extensible framework and data model for trajectory management in the context of the Semantic Web. The following points summarize our contributions:

- We present our model by means of an ontology called Semantic Trajectory Episodes (STEP). Ontologies provide an expressive formalism for representing data and, additionally, we see an increasing availability of datasets containing location information [Patrourmpas et al. 2014] for a variety of resources that can potentially be used to enrich trajectories.

Among the advantages of choosing a Semantic Web approach with ontologies for modeling and implementing our solution, we can highlight the possibility of integrating different data sources through federated queries, a transparent data model, support to machine reasoning by inference and the integration of data and metadata. We have focused on proposing a high-level data model for representing trajectory episodes and contextual elements with multiple levels of granularity and different options of representation of spatial and temporal extents as well as the ability of expressing quantitative and qualitative semantic descriptions.

- A framework based on the proposed ontology that serves as an interface between applications and our data model. The framework has three main algorithms for automatic annotation of trajectories:
 - An algorithm to detect stops, moves and noisy points in trajectories with sampling strategy based on the changes of position of the moving object. Our algorithm, called Stop, Move, and Noise (*MSN*) classification uses robust statistic measurements to identify outliers of distance, turning angle, time interval, and speed between points and determine in which of the three states each trajectory point belongs;
 - A method for approximating a time series based on the variability of its values. We propose a new time series compression algorithm called Adaptive Natural Breaks Approximation (*ANBA*) that finds a set of classes that provides a constant approximation of a series in a way that the intraclass variation of values is minimized and the interclass variation is maximized;
 - An algorithm for enriching trajectories with OpenStreetMap data complements the automatic annotation capabilities of the *STEP*

framework. We present the Intersection-based Spatial Annotation (ISA) algorithm, a method that explore the relationships of crowdsourced tags of OpenStreetMap features for selecting what types of geographic contextual element might be relevant for trajectory annotation.

1.3 Thesis outline

The remainder of the thesis is organized as follows:

In [chapter 2](#), the relevant general information about the thesis subject and related technologies is explored. We start by giving an overview of the Semantic Web and trajectory models in [section 2.1](#), we focus on the technologies of the Semantic Web stack in [subsection 2.1.1](#) and then discuss some of the most important works on trajectory models in [subsection 2.1.2](#). Works that lie in the intersection between Semantic Web and trajectories, specifically the ones that propose ontologies, are also discussed.

Still in [chapter 2](#), [section 2.2](#) is devoted to the presentation of trajectory annotation algorithms. More specifically, we discuss stop detection methods in [subsection 2.2.1](#), time series compression in [subsection 2.2.2](#), and trajectory annotation with intersecting spatial geographic features in [subsection 2.2.3](#).

The contributions of this thesis are concentrated in [chapter 3](#). First, we present the [STEP](#) ontology in [section 3.1](#) and discuss its evolution and current state. We also make a comparison with other data models in terms of their capabilities of representing episodes, contextual elements, granularity, and flexible episode extents.

The [section 3.2](#) presents the architecture of the [STEP](#) framework and its preprocessing methods. The three main algorithms present in the framework are presented in [subsection 3.3.1](#) ([MSN](#)), [subsection 3.3.2](#) ([ANBA](#)), and [subsection 3.3.3](#) ([ISA](#)).

We finish this work in [chapter 4](#) with a discussion of the achievements presented in it and the possibilities opened for future works.

State-of-the-Art

Contents

2.1	Semantic Web and Trajectory Models	12
2.1.1	Semantic Web	12
2.1.1.1	Geospatial Semantic Web	19
2.1.1.2	Temporal Modeling	22
2.1.2	Trajectory Models	24
2.1.2.1	Semantic Trajectory Ontologies	25
2.1.2.2	Other trajectory models	32
2.2	Trajectory Annotation	35
2.2.1	Stops identification	39
2.2.2	Time Series Segmentation	41
2.2.3	Spatial Annotation	45

In this chapter, we review fundamental concepts that form the groundwork of this thesis. The technologies that form the Semantic Web stack are briefly explained, followed by trajectory data models that have relevant impact on moving objects research. We give special attention to models that are tailored for Semantic Web applications, i.e. ontologies. In the second part, we look back to trajectory annotation methods, specifically for the ones related to stop identification, time series approximation and spatial context annotation.

2.1 Semantic Web and Trajectory Models

2.1.1 Semantic Web

The World Wide Web has experienced many changes in all of its aspects. The widespread of Internet connection to many parts of the globe that were hard to reach not long time ago associated with a decrease in manufacturing costs of connected devices has changed the way we use this global network. These changes reach a level where the Web can be considered a vital tool for the daily activities of some people. This increasing usage of tools and services based on the Web is assisted in great part by technological advances in its infrastructure and the development of standards.

The first idea of the web network has been conceived by [Berners-Lee \[1989\]](#) and since its embryonic stage it is based on nodes and links. Nodes can represent, for instance, documents, people, software modules, projects, or any other concept, while links can also have different meanings, e.g. dependency, composition, participation, usage, reference and so on. The idea of interlinking that made possible to navigate in a non linear way among resources in the Web was called hypertext, a term coined by [Nelson \[1965\]](#).

The concrete implementation of hyperlinks have been introduced by the HyperText Markup Language ([HTML](#)) and constitute the main format that composed the first document-oriented resources in the Web. However, the first web pages consisted basically of static content without much interaction between the client-side (mostly user's browsers) and the server-side (the content providers). The only small level of interaction of web pages were restricted interactions that had only effects in the currently displayed resource through the combination of Javascript, Cascading Style Sheets ([CSS](#)), and Document Object Model ([DOM](#)) manipulations.

Further development of server-side technologies have enabled the usage of messages to the server which call applications capable of building [HTML](#)

documents dynamically, creating an output more tailored in response to a specific request rather than retrieved from a previously stored file. This change in the way users consume web resources has allowed the raise of the Social Web, or Web 2.0.

An orientation towards metadata has always been part of the Web [Guns 2013]. This is evident with the usage of the `meta` tag in `HTML`, which provides a way of identifying the author of a document as well as its keywords, description, among other attributes. Rudimentary metadata support for links is also possible through the `rel` attribute of hyperlinks. However, these feature are rather weak in terms of semantic description to be useful for smart applications as they accept generic, arbitrary values that are not standardized.

Still in the realm of web documents, other initiatives addressed the inclusion of metadata to pages. Examples of these projects are the microformat¹ and the microdata² standards. The common objective of these specifications have always been to include more semantic information to documents and the entities referenced in them.

In an effort to create resources on the Web that are not only human-readable but also machine-readable, the Semantic Web approach (also referred as the Web of Data or Linked Data) has emerged as an alternative organization of resources on the Web. Bizer et al. [2009] summarizes the Linked Data as “using the Web to create typed links between data from different sources. These may be as diverse as databases maintained by two organisations in different geographical locations, or simply heterogeneous systems within one organisation that, historically, have not easily interoperated at the data level”.

Berners-Lee [2006] have introduced a set of best practices for publishing and interlinking structured data on the Web, that later became the Linked Data principles, as follows:

¹www.microformats.org

²<https://html.spec.whatwg.org/multipage/microdata.html>

1. Use Universal Resource Identifier ([URI](#)) as names for things.
2. Use Hypertext Transfer Protocol ([HTTP](#)) [URIs](#), so that people can look up those names.
3. When someone looks up a [URI](#), provide useful information, using the standards.
4. Include links to other [URIs](#), so that they can discover more things.

The first Linked Data principle advocates using [URI](#) references to identify, not just Web documents and digital content, but also real world objects and abstract concepts [[Heath and Bizer 2011](#)], and the fact of using [HTTP URIs](#) (second principle) comes from the fact that the current infrastructure of the Web should support the access to the entities as their description. The third principle advises that, once a resource is found by its [URI](#), it should be represented by using an agreed standard. The last principle concerns the essence of linked data, which is the easy discovery of new information by following links on the data set that is being first explored. This also encourages the utilization and spreading of popular vocabularies for defining the semantics of these links.

A visualization of the Semantic Web stack is shown in [Figure 2.1](#) where we can observe the concepts and abstractions of the stack along with some examples of specifications and solutions that implement the corresponding concept. As we have mentioned, the web platform is the base of the technologies that build the rest of the Semantic Web stack. In this basic building block, [URIs](#) are used to identify an abstract or physical resource. When a [URI](#) is linked to a retrievable resource, it can also be called as Uniform Resource Locator ([URL](#)).

The Resource Description Framework ([RDF](#))³ is the recommended stan-

³<https://www.w3.org/2001/sw/wiki/RDF>

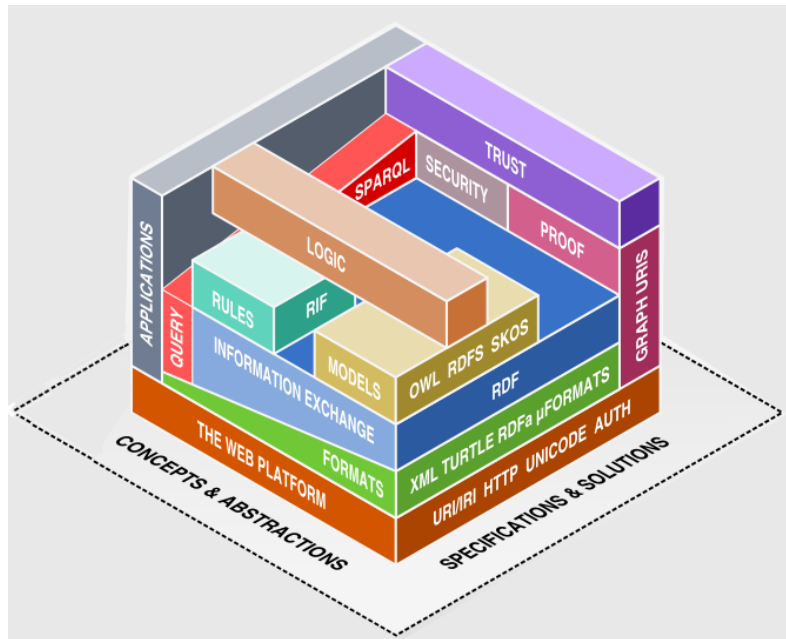


Figure 2.1: The Semantic Web stack (adapted from [Nowack 2009])

dard for information exchange. It consists in a graph-based data model represented as triples in the form *subject–predicate–object*. Each triple expresses a relation (represented by the *predicate* resource) between the *subject* and *object* resources. URIs can represent either of the three components of a tuple. However, an exception is made to *subject* and *object* resources, which can be represented as blank nodes, i.e. nodes that do not have a specified URI. Furthermore, *object* resources can also represent literals like strings, numbers or boolean values.

RDF Schema (RDFS) is a semantic extension of RDF that provides mechanisms for describing groups of related resources and the relationships between these resources [Brickley and Guha 2014]. It makes possible to specify vocabularies by using constructs related to classes and properties. RDFS specifies that a generic Resource, which can be subclassed by Container, Class or Literal, can be a member of other Resources. The Classes, which can be subclassed by other classes or datatypes, can also have the roles range and

domain. In practice, these roles define the origins and targets of Property entities.

Another important standard at the models layer of the Semantic Web stack is the Web Ontology Language (OWL)⁴, which uses RDFS to provide an implementation of a description logic capable of expressing more complex general statements about individuals, classes and properties [Simplerl et al. 2013]. OWL have introduced a new way of forming classes from other classes, properties and instances; the classification of properties into *object properties* and *data properties* that denote resources and literals as values, respectively; a richer set of statements linked to properties that allows expressing, for instance, equivalence or transitivity; new properties about instances to represent their relationships among them, for instance, whether two resources represent the same individual.

One of the main benefits of the Linked Data paradigm is the standardized data representation and access, which makes integrating data from different sources easier compared to integration from proprietary Web 2.0 APIs [Heath and Bizer 2011]. In order to access the data sets available in the Semantic Web, the standard query language SPARQL Protocol and RDF Query Language (SPARQL) has been established. At the time of this writing, SPARQL is composed by two language specifications, one for querying⁵ and other for updating⁶ the contents of a data set.

The basic SPARQL query (SELECT) is similar to its counterpart in Structured Query Language (SQL). It works by matching triple patterns, i.e. subject–predicate–object triads where one or more of the components are missing, in a graph by following some criteria specified in a WHERE clause. More sophisticated queries can be built with other operators such as OPTIONAL for including

⁴<http://www.w3.org/2001/sw/wiki/OWL>

⁵<https://www.w3.org/TR/rdf-sparql-query/>

⁶<https://www.w3.org/TR/sparql11-update/>

triples that matches only part of the criteria or filters that retrieve, for instance, triples containing a component that matches a regular expression.

While the `SELECT` clause is the most familiar and used one as it returns a table in a similar way of `SQL`, there is the `CONSTRUCT` query that acts in the exact same way, but returns a `RDF` graph instead of a table. Other useful types of `SPARQL` queries are `ASK`, which returns a boolean indicating whether a query pattern exist or not in the data set, and `DESCRIBE`, which retrieves a `RDF` graph containing all available information about one or more resources.

Easily accessing different data sets is one of the appealing features of Linked Data. `SPARQL` enables the merging of data distributed across the Web through the `SERVICE` keyword. This feature allows consulting `SPARQL` endpoints and combine results with local resources if needed. These types of queries are called Federated Queries⁷.

Concerning the Rules layer, we observe a rather fragmented environment in terms of implementation. Each triple store is likely to contain a different level of rules support. For instance, SPARQL Inferencing Notation (`SPIN`)⁸, a World Wide Web Consortium (`W3C`) member submission that expresses rules in `SPARQL`, is supported by AllegroGraph⁹ and RDF4J¹⁰. On the other hand, the Stardog triple store¹¹ supports a completely different set consisting in an implementation of the Semantic Web Rule Language (`SWRL`)¹² and a proprietary rules syntax. The idealized stack of Figure 2.1 exemplifies as technology of this layer the Rule Interchange Format (`RIF`)¹³ standard, which contains a series of *dialects* representing rule languages with various features and was conceived to facilitate ruleset integration and synthesis.

⁷<https://www.w3.org/TR/sparql11-federated-query/>

⁸<http://spinrdf.org>

⁹<http://franz.com/agraph/allegrograph>

¹⁰<http://rdf4j.org>

¹¹<http://stardog.com>

¹²<https://www.w3.org/Submission/SWRL>

¹³<https://www.w3.org/2005/rules/wiki/Primer>

Other yet to be standardized layers compose the rest of the Semantic Web stack. The Proof layer executes the rules and the Trust layer provides an assurance of confidence whether to trust the given proof or not.

Ontologies define the concepts and relationships used to describe an area of interest. This term is also interchangeably used with the term “vocabularies” as there is not clear division between them [W3C 2015]. They are used to classify the terms that can be used in a particular application, characterize possible relationships, and define possible constraints on using those terms.

There are now tens of billions of facts spanning hundreds of linked datasets on the Web covering a wide range of topics [Vandenbussche et al. 2016]. Many of these dataset share common vocabularies that are most popular than other in the Semantic Web. As examples of widely used ontologies, we can cite Dublin Core¹⁴, a set of vocabulary terms used in order to describe the resources, such as videos, images, web pages, etc.; Friend of a Friend (FOAF)¹⁵, a vocabulary to describe the relations among Agents in a social network; Simple Knowledge Organization System (SKOS)¹⁶, an OWL ontology used to represent controlled vocabularies, taxonomies and thesauri; geo¹⁷, an RDF vocabulary for representing latitude, longitude and altitude information in the World Geodetic System 1984 (WGS84); Geonames¹⁸, an ontology describing more than 10 million geographic features all over the world with information such as names of places in various languages, elevation, population, postal codes, etc.; OWL:Time¹⁹, an ontology of temporal concepts such as instants and intervals, together with information about durations, and descriptions of date and time; and DBpedia²⁰, a mapping of the main contents of Wikipedia

¹⁴<http://purl.org/dc/elements/1.1/>

¹⁵<http://xmlns.com/foaf/0.1/>

¹⁶<http://www.w3.org/2004/02/skos/core#>

¹⁷http://www.w3.org/2003/01/geo/wgs84_pos#

¹⁸<http://www.geonames.org/ontology#>

¹⁹<http://www.w3.org/2006/time#>

²⁰<http://dbpedia.org/resource/>

articles to an ontology.

2.1.1.1 Geospatial Semantic Web

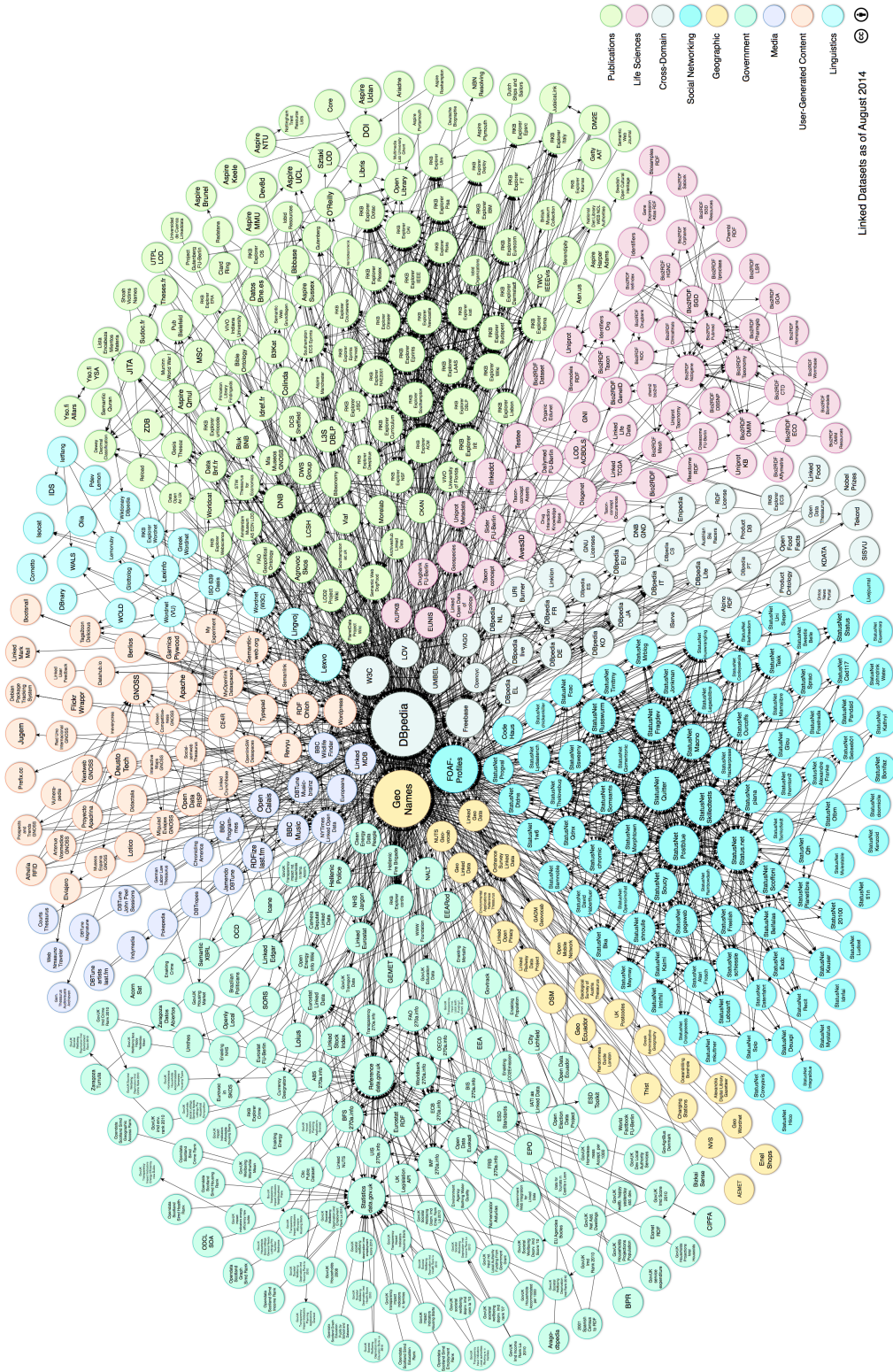
The Semantic Web technologies have provided new ways of structuring, reusing, and accessing data in general. For geographic data, it made some problems approachable and make new challenges emerge [Kuhn et al. 2014], making a real paradigm shift in this area. Janowicz et al. [2012] identifies two major problems in geospatial semantics: how geographic information should be modeled in an information ontology, and how geographic entities can be semantically linked to other kinds of information with related meaning.

In Figure 2.2, we can see the state of the Linked Open Data cloud and the interlinks among its datasets. It is possible to notice that a considerable part of this network is composed by databases containing information related to geographic entities. In 2011, at least 31 open data sets were available in the web containing more than six billion triples²¹ without taking into consideration the links from other non-geographic related data sets.

One of the first approaches to model geospatial data in the Semantic Web was the Basic Geo (WGS84 lat/long) Vocabulary [W3C 2003]. This initiative provided a minimalistic RDF vocabulary for describing points with latitude, longitude, and altitude properties from the WGS84 reference datum specification.

Geo OWL, inspired by GeoRSS [OGC 2006], have been proposed as a more comprehensive geospatial ontology [Lieberman et al. 2007]. This work has focused mainly in the definition of properties like `geo:featurename`, `geo:featuretype`, `geo:relationship`, `geo:floor`, and `geo:radius`. Subclasses for point, line string, polygon and envelope have also been addressed in this extension.

²¹<http://lod-cloud.net/state>



Linked Datasets as of August 2014

Figure 2.2: Linked Open Data cloud as of August 2014 [Schmachtenberg et al. \[2014\]](#)

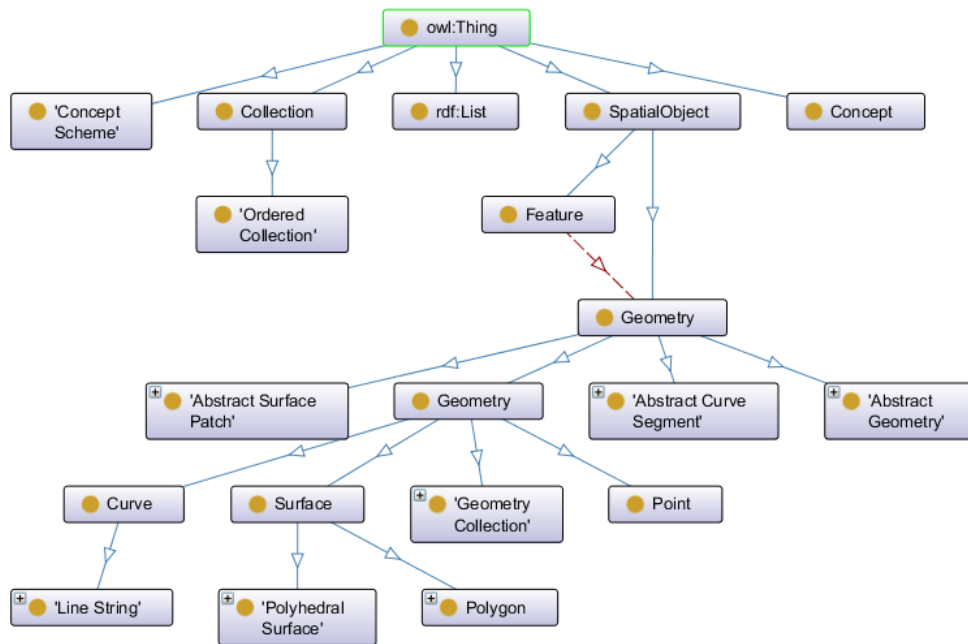


Figure 2.3: Excerpt of the GeoSPARQL vocabulary. The red dotted line represents a `hasGeometry` relationship, all other links represent subclassing relationships.

Open Geospatial Consortium (OGC) and W3C are prominent organizations that have been pushing the development of geospatial standards in the last few years as illustrated by their collaboration in the Spatial Data on the Web Working Group²². OGC have proposed GeoSPARQL, a standard for representing geospatial data in RDF and an extension to SPARQL for querying geospatial data.

Figure 2.3 depicts part of the GeoSPARQL vocabulary, where the main classes representing geometries are shown. The `SpatialObject` class, that represents everything that can have a spatial representation, also have object properties that express topological relations among spatial objects such as `contains`, `crosses`, `inside`, and other relations based on the Dimensionally Extended nine-Intersection Model (DE-9IM) pattern [Clementini et al. 1993;

²²<https://www.w3.org/2015/spatial>

1994], which is also compatible with Region Connection Calculus (RCC) [Randell et al. 1992] and point-set topological spatial relations [Egenhofer and Franzosa 1991]. At the time of this writing, the GeoSPARQL standard is composed by two namespaces: `geo`²³ and `geof`²⁴.

2.1.1.2 Temporal Modeling

The spatial aspect is one of the main elements when creating semantic web applications. However, other facet is equally important for the development of spatiotemporal applications. In the following, we discuss the relevance of some works that help modeling temporal characteristics.

The general standard for data and time is the ISO 8601 [ISO 1988], which specifies a format for representing dates, times, durations, time intervals, and time zones. However, the standard does not assign any specific meaning to elements of the date/time to be represented, letting the meaning dependent of the context of its use.

Allen [1983] have defined a set of relationships that can take place between two temporal intervals that became known as the Allen's interval algebra. Considering the inverses of these relations, there are a total of thirteen ways in which an ordered pair of intervals can be related. These relationships, illustrated in Figure 2.4, are the following:

- X before Y: interval X takes place before interval Y with some time between them;
- X meets Y: interval X happens exactly before interval Y, they meet in time because there is no gap between them;
- X overlaps Y: interval X starts before Y and they have some time in common, then X ends and Y continues;

²³<http://www.opengis.net/geosparql#>

²⁴<http://www.opengis.net/def/function/geosparql>


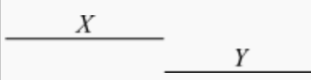
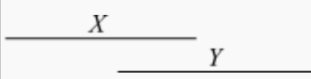
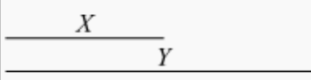
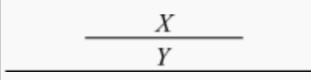
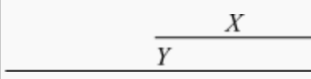
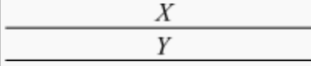
Relation	Illustration	Interpretation
$X < Y$ $Y > X$		X takes place before Y
$X m Y$ $Y mi X$		X meets Y (<i>i</i> stands for <i>inverse</i>)
$X o Y$ $Y oi X$		X overlaps with Y
$X s Y$ $Y si X$		X starts Y
$X d Y$ $Y di X$		X during Y
$X f Y$ $Y fi X$		X finishes Y
$X = Y$		X is equal to Y

Figure 2.4: Allen's Interval Algebra.

- X starts Y: interval X shares the same beginning as Y, but ends before Y ends;
- X during Y: interval X is fully contained within Y;
- X finishes Y: interval X shares the same end as Y, but begins after Y begins;
- X equal Y: X and Y share both starting and finishing times.

Hobbs and Pan [2004] have developed OWL:Time, an ontology of temporal concepts for describing the temporal content of Web pages and the temporal properties of Web services. The ontology defines the entities `Interval` and `Instant` that represent things with extent and point-like things, i.e. with no interior points, respectively. These entities allow the definition of all Allen's interval algebra through entities like `intervalEquals`, `intervalBefore`, `intervalOverlaps`, etc.

OWL:Time also covers other aspects of time representation. The `DurationDescription` class and its subclasses allows an interval to have multiple duration descriptions (e.g., 2 days, 48 hours), but still have only one duration. Similarly, `DateTimeDescription` have the properties `unitType`, `year`, `month`, `week`, `day`, `dayOfWeek`, `dayOfYear` `hour`, `minute`, and `second` to describe intervals or instants. A time zone ontology is also described as an appendix of the OWL:Time standard. More recently, there were efforts in updating the Time ontology to extent it to cases which the temporal reference system is not fixed to the Gregorian Calendar in advance, to enable position or duration to be described using a number or nominal value, and add support to different temporal reference systems²⁵.

2.1.2 Trajectory Models

In this section, we present some important works regarding trajectory modeling. The term Semantic Trajectory has been used for referring to models that can be used to enrich trajectories beyond latitude, longitude and timestamp information. Semantic Trajectories do not necessarily refer to the Semantic Web, however, there is an intersection in some cases where Semantic Web tools are used to model trajectories.

One of the most influential works in trajectory modeling is the one by [Spaccapietra et al. \[2008\]](#) where the authors elicit some requirements for trajectory modeling. First, three kinds of trajectories are identified: i) metaphorical trajectory, consisting in changes of a time varying attribute of an entity, e.g. the career trajectory of a person or the price for a stock; ii) naïve trajectory, consisting in geographic trajectories that are not defined in terms of spatial coordinates, and iii) spatiotemporal trajectory, consisting in changes of position in a 2D or 3D space of an object. Then, two facets of trajectories are

²⁵<https://www.w3.org/TR/2016/WD-owl-time-20160712/>

defined: the *geometric facet* – the spatiotemporal recording of the position of the traveling point – and the *semantic facet* – the information that conveys the application-oriented meaning of the trajectory and its related characteristics.

Spaccapietra et al. [2008] also gives a generic characterization of semantics of trajectories. The main conceptualization of their work consist in defining trajectories as a sequence of stops and moves as can be seen in Figure 2.5. Besides other requirements, the authors summarize what a conceptual trajectory model should cover as the following:

In summary, a conceptual model for trajectories must support the characterization of trajectories and their components with attributes, semantic constraints, topological constraints, and links to application objects. All trajectory components are spatio-temporal data, with begin, end and stops having a point geometry, while moves have a time-varying point geometry. From the temporal perspective, begin and end are of type *Instant*, while stops and moves (as trajectories) are of type *TimeInterval* and can thus be annotated with non-varying and varying properties [Spaccapietra et al. 2008].

2.1.2.1 Semantic Trajectory Ontologies

The Stop/Move model have then inspired works that translated similar ideas into ontologies. Baglioni et al. [2008], for instance, have proposed an ontological approach for semantic modeling and reasoning on trajectories where the authors highlighted the need of transforming raw traces into high level representations, i.e. semantic trajectories. The authors have formulated OWL DL axioms for identifying malicious behavior in a recreational game activity based on the duration and location of stops.

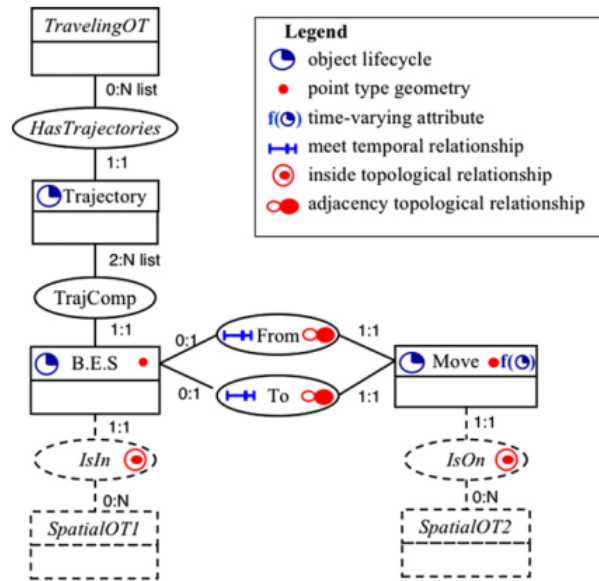


Figure 2.5: Conceptual view of trajectories [Spaccapietra et al. 2008]

The same strategy was followed in Baglioni et al. [2009] for the specification of touristic activities, however, the aim was the detection of frequent patterns. Similarly, Renso et al. [2013] proposes the slightly modified version of this ontology as a core model containing the entities *SyntacticTrajectory*, *Stop*, *Move*, *Time*, *Place*, and *Pattern* to represent human behavior. The behavior in these cases are predefined sequences of stops manually annotated and encoded as ontological axioms, e.g. Home to Work behavior.

The work of Yan et al. [2008] describes a semantics-oriented framework for structuring, modeling and querying trajectory data motivated by the fact that most models did not take into consideration non-geometric information that could help explain movement. In their work, trajectories have a geometric component following the Stop/Move model, a geographic component, which represents the semantics of a trajectory, and an application domain component that links the trajectories to data that is specific to the application where it is applied. These three components are exemplified in Figure 2.6.

Yan et al. [2008] present a framework for a semantics-oriented structu-

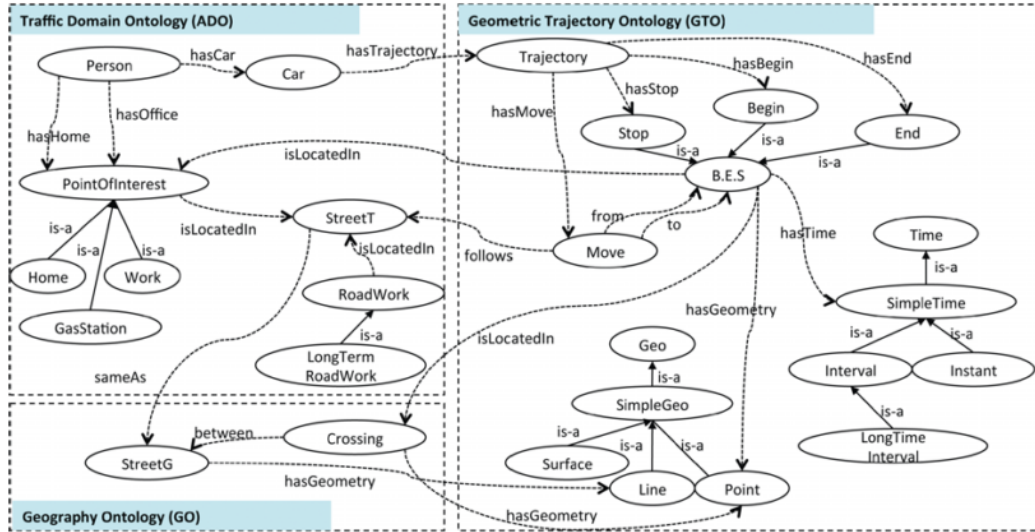


Figure 2.6: Example of trajectory ontologies for an application in the traffic domain [Yan et al. 2008]

ring, modeling and querying of trajectory data. They define that an ontological infrastructure should have three modules: geometric, geographic, and application domain ontologies. The Geometric Trajectory Ontology is the only one that is proposed as application-independent and it encloses four sub-ontologies, namely *Spatial*, *Temporal*, *Spatiotemporal*, and *Trajectory* ontologies. Spatial and temporal concepts are reused from the Modeling of Application Data with Spatio-temporal features (MADS) [Parent et al. 2006] specifications, which define well known simple concepts of these two domains (*Point*, *Line*, *Interval*, *Instant*) and are equivalent to other spatial and temporal ontologies that we used in our work. The *Spatiotemporal* ontology defines pairwise combinations of the spatial and temporal concepts (for instance, a *TimePoint* has a geometry of type *Point* and is linked to an *Instant*, similar structures are defined for *TimeLine* and *TimeSurface*) as well as time-varying classes that consists of lists of the mentioned classes. From the *Spatiotemporal* ontology, only the concept of *TimePoint* is used in the *Trajectory* ontology. The main difference compared to our model is that stops and moves

are classes, whereas our model do not enforce even the existence of instances to represent stops and moves, therefore our model is more flexible by keeping the possibility of having episodes to represent these concepts.

Orellana and Renso [2010] modeled pedestrian movement in a national park as a system based on interactions between movement patterns and context. For this, the authors have created an *Interactions ontology and taxonomy* as well as encoded a *Movement patterns taxonomy*. These interactions are the basis for identification of behavior when associated with further contextual information (in this case, information about the availability of attractions, location of information points, among other park facilities and their status). Examples of interactions are: *encounter*, *approaching*, *guiding-following*, *visiting*, *route choosing*, *attraction*, *flocking*, *aggregation*, and *trail formation*. Behaviors could be, for instance, *exploring* (when the visitor has a route choosing interaction and stops at information points), *socializing* (when the visitor presents an encounter interaction located at some crossroad or path), and *disturbing* (when a visitor is located at a park feature with forbidden access). The available dataset came from three different information sources. A questionnaire that recorded visitor characteristics; a set of point coordinates captured by GPS receivers given to the visitors; and a dataset containing the path network and the locations of access points. These patterns are identified thanks to subclass reasoning through predefined axioms such as the following:

```
Disturbing ≡ visitor_has_interaction some (Visiting and
(is_located_at some Park_Feature and (has_Accessibility_level
has Forbidden))).
```

Predefined description logic axioms have also been used by Camossi et al. [2013] to identify anomalous behavior of ship trajectories. A generic ontology called Moving Object Ontology is presented and specialized to the maritime surveillance domain. The authors show a pattern discovery workflow based

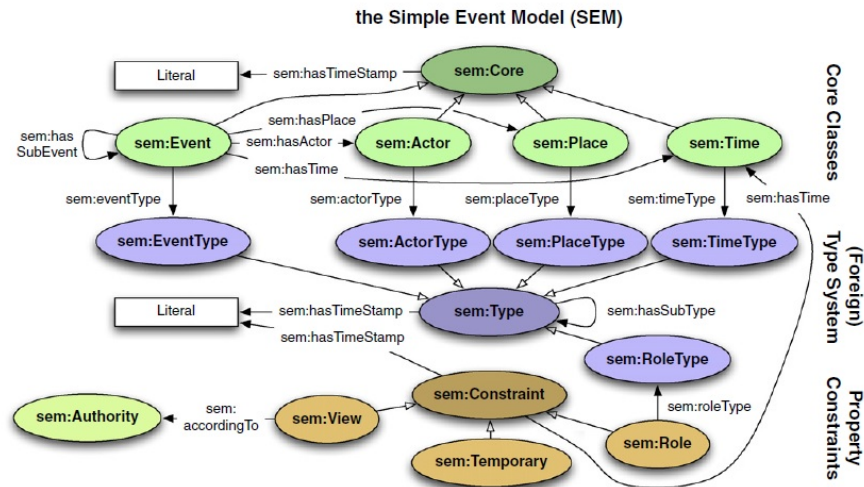


Figure 2.7: The Simple Event Model ontology [van Hage et al. 2009b]

on the combination of ontology concepts with logical operators, defining implicitly the class of objects describing the container itineraries following the corresponding suspicious pattern. The patterns are related to the points where ships stop during their trajectories.

Trajectories can also be modeled as a sequence of events. van Hage et al. [2012] propose such approach to represent ship trajectories using the generic Simple Event Model (SEM) [van Hage et al. 2011] shown in Figure 2.7. The authors argue that events are central elements in the representation of data from a variety of domains such as history, cultural heritage, geography and multimedia. In the case of trajectories, stops and moves are considered as events that can be further detailed with sub-events.

Ontology design patterns have emerged to help to cope with problems faced when reusing third-party ontologies. According to Gangemi and Presutti [2009], some issues that may arise in these cases is related to “limited assistance in using unfriendly logical structures, some large, hardly comprehensible ontologies, and a bunch of good practices that must be discovered from the literature”. Gangemi [2005] coined the term ontology design pattern, which

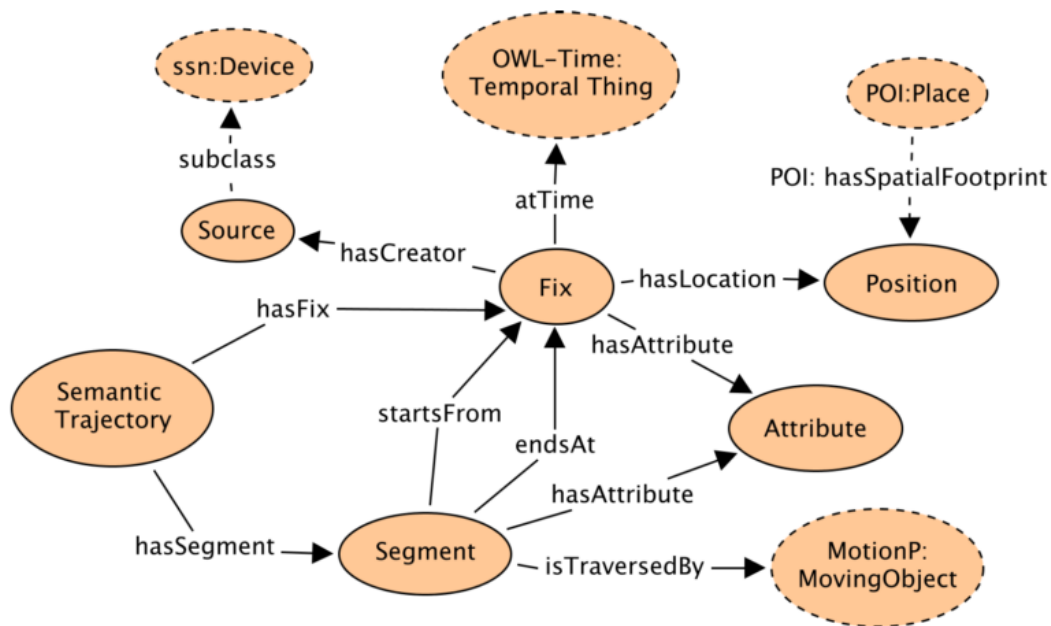


Figure 2.8: The trajectory design pattern proposed by [Hu et al. \[2013\]](#)

consists in small or modularized ontologies with explicit documentation of design rationales and best reengineering practices.

This approach was taken by [Hu et al. \[2013\]](#), who proposes a geo-ontology design pattern addressing the design of classes and properties found commonly in semantic trajectories across application domains, shown in Figure 2.8. In order to create the design pattern, the authors have created a generic use case with some potential queries that could be answered once data is modeled following their pattern. The resulting model allows composing semantic trajectories with fixes and segments. Segments are entities that start and end at fixes and may have attributes, just like the fixes. Each fix may have a location and a time besides a source to represent the person or device that originated it. Although not showed in Figure 2.8, there is a sequential relationship between the fixes linked to segments though `startFrom` and `endsAt` properties. This relationship is enforced by logic axioms encoded in the ontology.

A framework supported by ontologies is presented by [Vandecasteele et al.](#)

[2014]. By reusing the ideas of Yan et al. [2008] (the separation of geometric, geographic, and domain ontologies as shown in Figure 2.6), the authors represent trajectories with GeoRSS Simple ontology²⁶ and OWL:Time instances, besides using Geonames and the Geopolitical ontology²⁷. An ontology was created containing the terms related to the maritime domain. Human knowledge encoded in the form of semantic rules written in SWRL automatically create semantic events that are represented using the Simple Event Model ontology.

The authors have evaluated their framework by annotating vessel trajectories with “speeding down” and “speeding up” events based on the speed of each trajectory point. In another example, they have created rules that create a new semantic event each time the vessel changes its heading by a predefined threshold generating thus a “suspicious heading change” event.

Fileto et al. [2015b] presented the ontology Baquara², an evolution of the Baquara ontology created by Fileto et al. [2013]. The proposed framework consists in a domain independent ontology that provides a conceptual model to enrich data and support knowledge-based queries for movement analysis. In Baquara², a moving object’s position sequence is composed by movement segments. These segments can represent trails (time-ordered sequences of posts in social media), trajectories, or episodes (e.g. stops and moves). The movement segments of Baquara² are tuples containing type, geometry, initial and final positions, time span, possible references to hierarchical relationship with other segments (father, previous, next, level), order, and annotations.

Authors also present a general architecture for semantic enrichment and analysis of movement data. They propose an algorithm that takes preprocessed segments containing a user stops and posts in social media and matches them to geographic features based on the distance and textual similarity. An

²⁶http://www.georss.org/rdf_rss1.html

²⁷<http://aims.fao.org/geopolitical.owl>

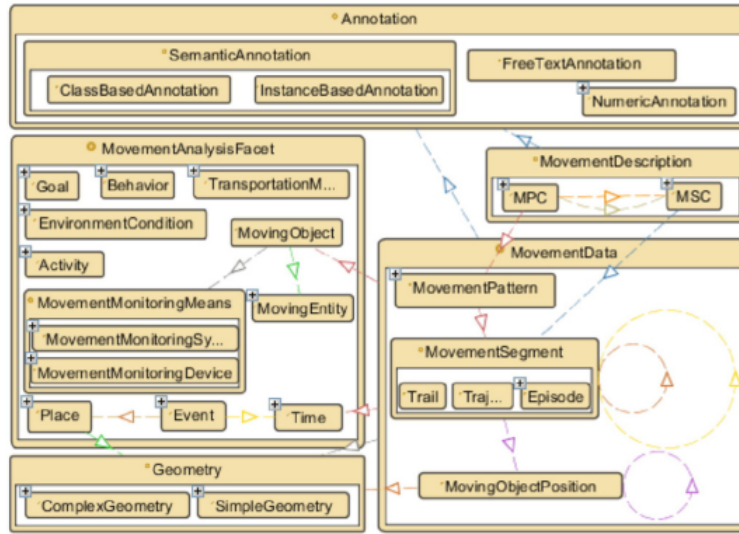


Figure 2.9: The Baquara² ontology overview [Fileto et al. 2015b]

innovative aspect of Baquara² is the introduction of explicit constructs to express hierarchical relations among segments. This allows the representation of trajectories at different levels of details and the analysis of traces at varying granularities.

2.1.2.2 Other trajectory models

Besides ontology-based models, we highlight in this section some important modeling solutions for trajectories as some models can be easily transformed into ontologies. We also explore conceptualizations that allow the representation of segments of trajectories into episodes.

One of the first definitions of episodes in the area of trajectory analysis is presented in the work of Mountain and Raper [2001], where “an episode represents a discrete time period for which the user’s spatiotemporal behavior was relatively homogeneous”. Other authors have used similar concepts such as Segment of Interest (SoI) [Braga 2012], Segment [Hu et al. 2013, Nogueira et al. 2014], or SemanticSubtrajectory [Bogorny et al. 2014] to represent parts of trajectories that share common characteristics.

In the work of [Guc et al. \[2008\]](#), trajectories are semantically enriched by two types of annotations: episodes and trips where the former describes homogeneous sections of the trajectory and the latter serves to group a sequence of episodes on a higher semantic level. However, they do not present a concrete conceptual data model. There is a strong assumption that episodes partition leaves no gaps in the annotation, implying that there are annotations for the entire extent of the trajectory, which is frequently not the case in many real world applications.

A conceptual model has been proposed by [Andrienko et al. \[2011\]](#) and also extends the work of [Spaccapietra et al. \[2008\]](#) by focusing on the concepts of events and spatiotemporal context. In this model, stops and moves are particular types of spatial events. The building blocks of this model are the following entities: **Spatial Object** are things having one or more positions in space, **Event** have a position in time, **Spatial Event** are things that have positions in space and time, **Static Spatial Object** have only one position in space, **Mover** is something which position changes over time, and **Moving Event** is the action performed by the mover.

[Andrienko et al. \[2011\]](#) argue that movement behaviors can only be understood by considering the relations that happen between moving objects and the environment in which they move. They define what elements constitute the spatiotemporal context as complex and heterogeneous physical space, in which characteristics vary from place to place and change over time; complex and heterogeneous physical time, in which day differs from night, summer from winter and so on; static and dynamic objects existing in space; events occurring over time.

The CONceptual model of Semantic TrAJecTories (**CONSTAnT**) model [[Bogorny et al. 2014](#)] is another conceptual model for semantic trajectories that has been recently presented. The model focus on describing the mo-

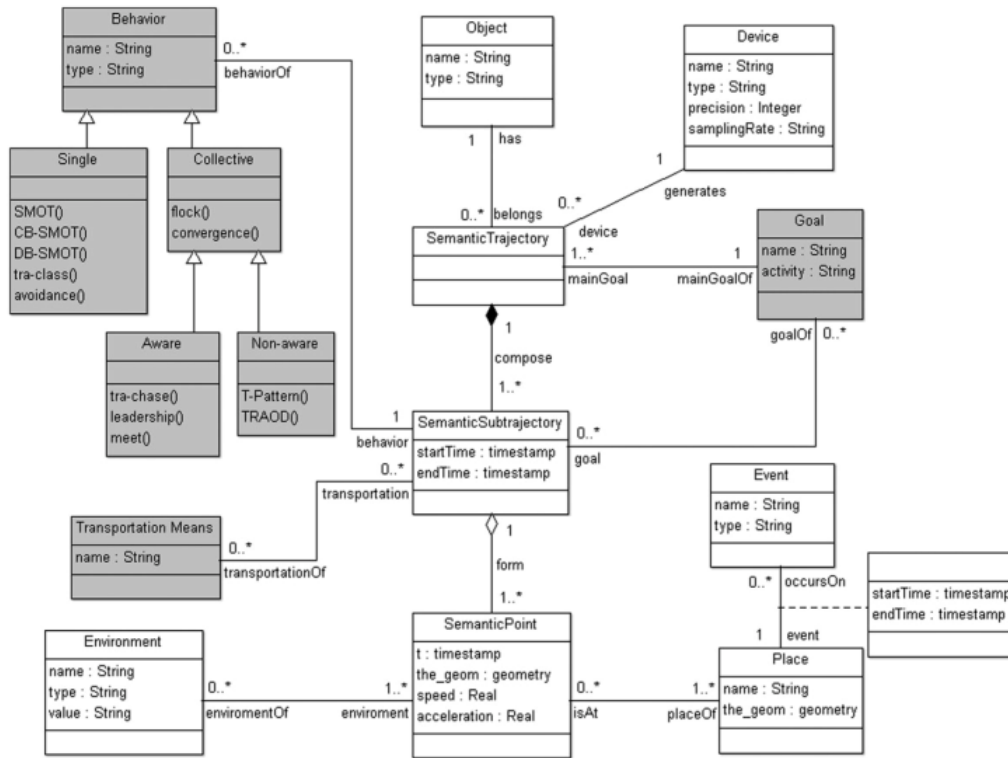


Figure 2.10: The **CONSTAnT** conceptual model for semantic trajectories [Bogorny et al. 2014]

ving object, the device that generated the trajectory, the semantic trajectory, eventual semantic subtrajectories, semantic points, the environment, the place and events. Also, goals, transportation means and behaviors of trajectories and subtrajectories are important parts of **CONSTAnT**, which is shown in Figure 2.10.

The authors of **CONSTAnT** also show some examples of instances of their model. For example, it is possible to represent a trajectory of a person that is visiting Rome with the goal of attending an event and makes a visit to the Coliseum to see a photographic exhibition. In this case, **CONSTAnT** allows representing a semantic trajectory with a subtrajectory corresponding to the visit in a touristic place and link it to an event that might be happening there. Besides, the authors also show in the example the possibility of linking

weather information to specific points of the subtrajectory.

In section 3.1 we are going to revisit some of these models to put them into the perspective of the ontology that we propose in this thesis. More specifically, we are going to compare related works in terms of segmentation support, granularity, context, and extent representation.

2.2 Trajectory Annotation

In this section, we review relevant works dedicated to the annotation of moving objects' trajectories that are directly related to the algorithms that we propose in this thesis. The term "annotation" is rather general and can be the objective of many steps in trajectory analysis. We give an overview of the main tasks on trajectory analysis and focus on the ones that propose segmentation and compression methods. Some time series compression algorithms are also briefly explored as many trajectory attributes can be expressed as the evolution of values in time (e.g. speed, direction, acceleration).

Hägerstrand [1970] was one of the first works that have dealt with the specification of paths of individuals in time-space. His ideas have originated the concept of space-time cube, a 3D representation of movement where space is usually depicted in the x and z axis and time is represented in the y vertical axis, space-time prism, a visual estimative of an individual's mobility given its constraints, and ultimately giving the basis for the Time Geography field. Thanks to the advances in computational representation of space-time cubes and prisms that followed, Miller [2005] proposed a measurement theory for Time Geography that formalized many of these concepts, which provided foundations for querying and analyzing movement data.

Some basic characteristics of movement have been elicited by Andrienko et al. [2008], that divided attributes according to whether they can be expressed in time instants or intervals. At a given moment (i.e. instant in time)

we can measure the time stamp itself, the position of the entity in space, the movement's direction, the speed of the movement, the moving object's change of the direction (turning angle), the acceleration, among other examples. As *overall characteristics*, [Andrienko et al. \[2008\]](#) cite attributes that can be computed over the whole trajectory of a part of it, such as its shape, traveled distance, duration, and various statistics over these characteristics. It is also of interest the relationships among trajectories. Being from the same moving object or from distinct entities, possible analysis include similarity measures, spatial and temporal relations like co-incidence in space and/or time in many forms (ordered and unordered colocation, full or partial co-existence in time, full or lagged co-incidence in space-time). A similar classification has also been explored by [Dodge et al. \[2008\]](#) as primitive movement parameters that can be used to calculate primary and secondary derivatives.

Movement traces and their derivate attributes are the main inputs for a series of processes that constitute a trajectory enrichment workflow. Many processing phases are needed to transform raw trajectories into useful records for applications. Figure 2.11 shows an overview of the process from raw sensor measurements, passing through trajectory reconstruction, aggregation, transformation, data mining, among other tasks. This chain generates knowledge about the data that can be used by applications.

A number of trajectory definitions can be used to express the many steps that are needed to fulfill the semantic enrichment pipeline. [Spaccapietra et al. \[2013\]](#) argue that usually there is a *movement track* representing the whole lifespan of an object that may originate *trajectories*, which are the segments of an object's movement that are of interest for a given application. Still according to [Spaccapietra et al. \[2013\]](#), among the possible representations of a trajectory, one can have a continuous, discrete, or stepwise representation, being the latter the result of the implementation of a step function that maps a

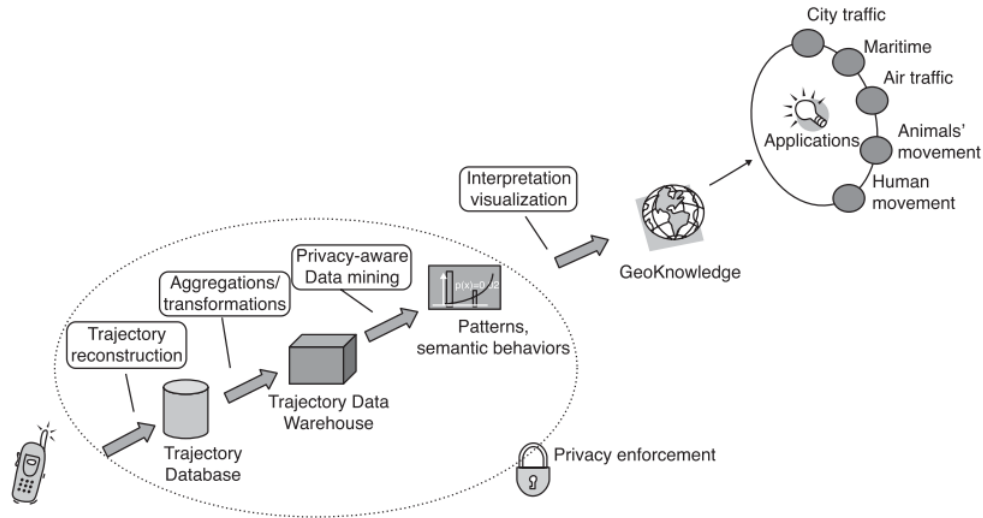


Figure 2.11: Overview of a trajectory enriching process [Marketos et al. 2013]

time interval to a finite set of annotations. This is equivalent to the definitions of episode previously discussed in this thesis.

Another possible view of this process allows the definition of different types of trajectories, e.g. *Spatio-temporal Trajectory*, *Structured Trajectory*, *Semantic Trajectory*. Yan et al. [2012] have defined these terms to detail the various transformations that lead to the enrichment of trajectory data with semantic information. Such computing platform is depicted in Figure 2.12.

As has been previously discussed, episodes have constantly appear in the literature as a reasonable concept to represent relevant parts of trajectories and their corresponding annotations [Mountain and Raper 2001, Guc et al. 2008, Yan 2011, Parent et al. 2013, Spaccapietra et al. 2013, Fileto et al. 2015a]. Episodes can play roles as either the output of annotation algorithms or as the input of more complex processes. Trajectory preprocessing tasks such as noise filtering, stay point detection, trajectory compression, and trajectory segmentation [Zheng 2015] are examples of steps that may have episodes as the resulting outputs of algorithms.

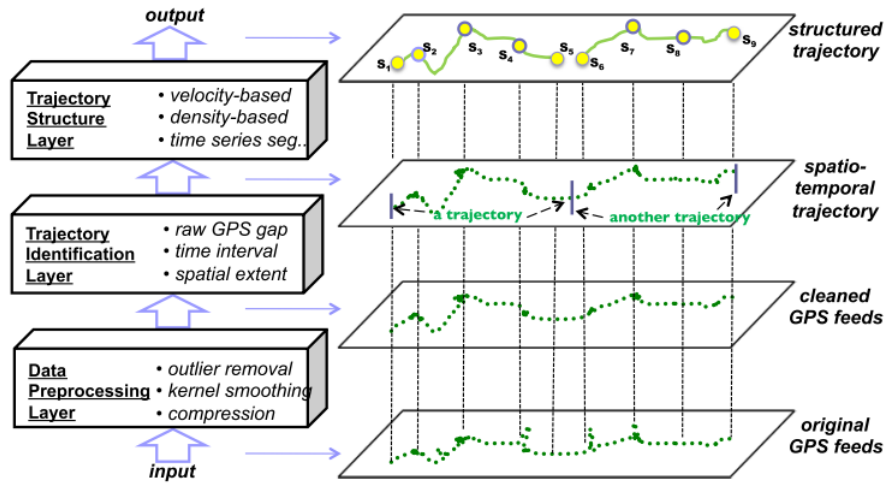


Figure 2.12: Trajectory computing platform [Yan et al. 2012]

Most compression methods concentrate on the geometric features of a trajectory, e.g. the Ramer–Douglas–Peucker algorithm [Ramer 1972], by iteratively removing trajectory points and verifying if some error threshold is not violated. The aim of such methods is representing the general shape of the original trace with a small amount of points.

Other compression methods are focused on trajectory attributes such as speed and direction. Potamias et al. [2006], for instance, have proposed approximation techniques for online compression by constructing *safe areas* for the last updated point and then checking if new points fall within the safe area. The size and shape of the safe area vary according to predefined thresholds for speed and direction.

The transportation network has been explored for building compressed representations of trajectories as proposed by Schmid et al. [2009], Richter et al. [2012]. In their work, reference points of the transportation network replace trajectory points after they pass through a map matching process. These reference points can be streets, public transport stations and lines, among other elements of spatial context as well as references to the movement pattern on the network expressed in natural language, e.g. “straight”.

An important task similar to compression in its methods but not necessarily with the objective of reducing the storage requirements of trajectories is the segmentation task, that refers to the process of partitioning movement data into multiple episodes, with the goal of simplifying or changing the representation of the trajectory into something that is more meaningful or easier to analyze [Laube 2014]. By this definition, we can conclude that segmentation is a concept that encompasses compression but may serve to other tasks such as clustering and classification. For this reason, we consider that trajectory annotation is in most cases a segmentation process being the exceptions the cases where the annotation concerns the whole trajectory.

Relevant works in this direction have been recently presented, for instance, by Buchin et al. [2009; 2013], Alewijnse et al. [2014]. Zheng [2015] identifies three types of segmentation: time interval-based, shape-based, and semantic-based. The first one refers to breaking down trajectories according to time gaps among them. The second one focuses on spatial characteristics, e.g. direction change, or line simplification methods. The third one is more flexible according to the application, e.g. by transportation means, stay points, among other criteria.

In the following subsections, we review some methods directly related to the algorithms proposed in this thesis. The focus of these methods are the semantic segmentation of trajectories based on stops, moves, and noisy data, generic time series segmentation that can be applied to the numeric temporal evolution of any trajectory attribute, and the annotation of segments of trajectories with spatial context data.

2.2.1 Stops identification

We can observe that many of the proposed segmentation strategies rely on some assumptions about the gathered data and, in some cases, additional

external data is used to assist the algorithms. The Stops and Moves of Trajectories (SMoT) method presented by [Alvares et al. \[2007\]](#) define stops as the trajectory points that intersect “candidate stops”, i.e. a previously defined set of polygons and minimum time durations. A major disadvantage of this approach is the need for manually selecting candidate stop polygons as well as the minimal time duration needed to consider each region as a stop. Putting a hard threshold on the duration of stop may cause the algorithm to miss important stops that have a time duration close to the threshold.

The Point-of-Interest Extraction (PIE) algorithm [[de Graaff and Keulen 2016](#)] uses the underlying geography polygons but it also considers reductions in speed, changes in direction and the accuracy of each GPS point. Whereas speed and direction can be easily computed from trajectory points, the availability of accuracy, while very important to assess the signal quality, is not commonly stored by most applications. This factor imposes an important obstacle to use this method with trajectories captured by third-party applications.

[Palma et al. \[2008\]](#) proposed the Clustering-Based Stops and Moves of Trajectories (CB-SMoT) algorithm. The premise of CB-SMoT is that the moving object’s speed decreases significantly when an interesting place is being visited (therefore, it is a stop). However, another important assumption is that the recording device keeps storing points even when the object is stopped so the stops can be characterized as regions where there is a greater spatial density of points.

[Yan et al. \[2010\]](#) propose a model and computing platform for abstracting trajectories at different levels starting with basic abstractions (e.g. stop, moves) to enriched higher-level abstractions (e.g. office, shop). In the first layer of their computing platform, trajectories are smoothed with a Gaussian kernel-based local regression model and outliers are identified by velocities thresholds

according to domain knowledge (e.g. car, human, cycle etc.).

In their Trajectory Structure Layer, the identification of stops and moves is performed by the means of a method for determining a speed threshold based on the type of moving object and the underlying movement area. For each GPS point of a given moving object, the speed threshold is determined by a function of the moving object’s average speed and the average speed of other moving objects in that position. For calculating the latter, the space is divided into a grid and an average speed is calculated for each cell. The dynamic speed threshold function is shown in Equation 2.1.

$$\Delta_{speed} = \min\{\delta_1 \times \overline{objectAvgSpeed}, \delta_2 \times \overline{positionAvgSpeed}\} \quad (2.1)$$

Their algorithm receives two coefficients of speed thresholds (δ_1 and δ_2), which are used in Equation 2.1, and a minimal stop duration (τ) as parameters. We can identify some differences from our method. First, using the non-robust average speed may difficult correct identification of stops if there is a large range of speeds in a single trajectory. Second, while there were an effort to dynamically set speed thresholds, this has not been done to the stop duration, which is still defined as an absolute value (e.g. 15 minutes). Third, although not discussed in their work, maintaining the spatial grid updated with the average speed of each cell may be a computational expensive procedure and the granularity of this grid has also not been explored. Furthermore, the authors follow the assumption that GPS data is sampled with uniform frequency along trajectories.

2.2.2 Time Series Segmentation

Considering the spatio-temporal nature of trajectories, it is easy to notice that many aspects of a moving object’s trajectory are derived from the relationship of these factors and can be expressed as a series of values ordered by time, i.e. a

time series. These trajectories characteristics such as speed, acceleration, and elevation are essential to the analysis of the movement dynamics. Therefore, methods for abstracting time series constitute an important task in trajectory modeling and analysis. Usually, the output of time series compression methods can be easily adapted to construct trajectory episodes.

The task of finding homogeneous periods in temporal data can be differentiated according to either the analysis happen with a group of series (constituting in dimensionality reduction task) or if it is performed with only a single data stream (leading to compression and filtering) [Papadimitriou et al. 2013]. Time series compression can also be considered as a one-dimensional clustering problem and are often referred to as segmentation as it can be used for creating an accurate approximation of time series, by reducing its dimensionality while retaining its essential features [Esling and Agon 2012].

Diverse possibilities of time series representations have been proposed. This task plays the important role of preserving the most relevant features of a given time series at the same time that it prioritizes using the least amount of storage as possible. Basically, the types of representation can be divided into two categories: non-data adaptive and data adaptive [Wang et al. 2013]. The main difference between the two is that in non-data adaptive methods a common representation is selected for every time series in a set in order to minimize the global reconstruction error, whereas in data adaptive the structure of each individual series is considered. While the first one is usually more efficient in terms of computational time, the second gives results that are more similar to the input signal in most cases.

In the following, we briefly explain some of the representation methods that are later compared in the evaluation of our algorithm ANBA. Piecewise Aggregated Approximation (PAA) [Yi and Faloutsos 2000, Keogh et al. 2001a] is one of the most basic and fast segmentation algorithms. It consists in

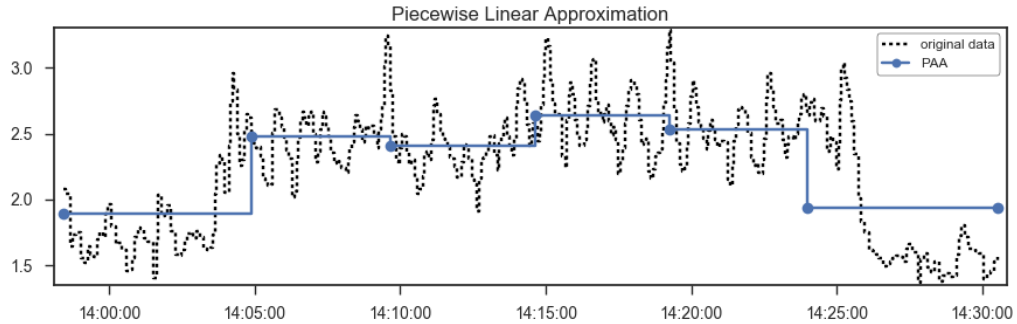


Figure 2.13: Example of Piecewise Aggregated Approximation of a time series

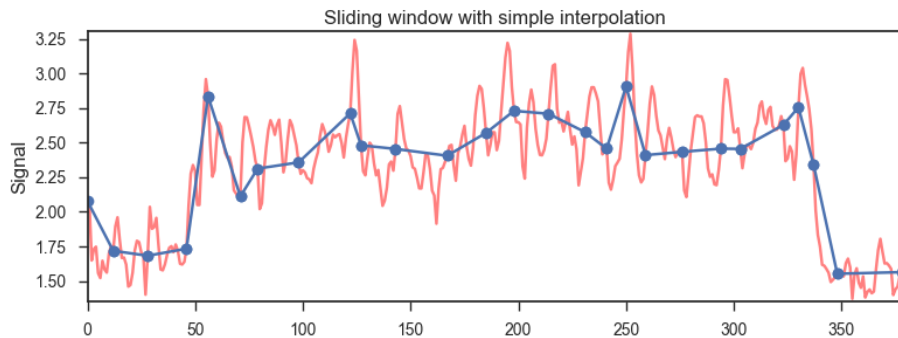
dividing the series frames of equal length previously defined as a parameter. Then, the mean value of the data falling within a frame is calculated and a vector of these values becomes the data reduced representation. Figure 2.13 shows one example of this segmentation method.

Piecewise Linear Segmentation (PLS) [Shatkay and Zdonik 1996] is a widely used method for representing time series. Three important approaches to achieve such representation are constantly cited in the literature. Keogh et al. [2001b] give a concise definition of these three strategies:

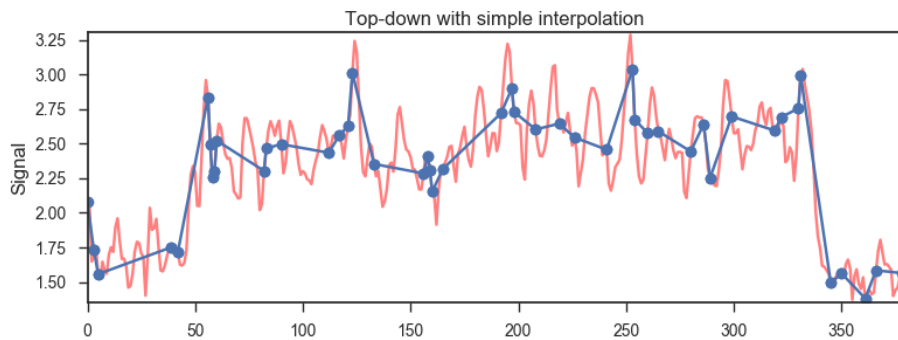
- Sliding Window: The segment is grown until it exceeds some error bound. The process repeats with the next data point not included in the newly approximated segment.
- Top-Down: The time series is recursively partitioned until some stopping criteria is met.
- Bottom-Up: Starting from the finest possible approximation, segments are merged until some stopping criteria is met.

In order to find the approximate value between two extreme points of each segment, two main strategies can be followed: linear interpolation or linear regression. While the first one just connects a line from the first to the last

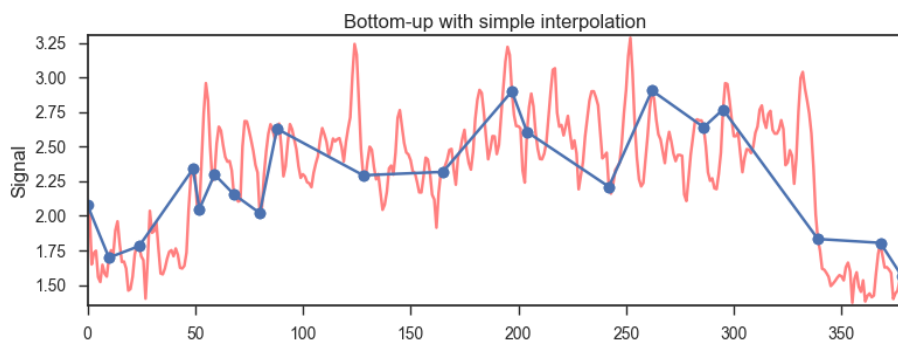
point, the later calculates the best fitting line that minimizes the least squared error for the intermediate points, which is more computationally expensive than the first option. Figure 2.14 shows an example of the same time series segmented with the three strategies.



(a) Sliding Window



(b) Top-down



(c) Bottom-up

Figure 2.14: Different approaches of Piecewise Linear Segmentation

In a recent survey, [Aghabozorgi et al. \[2015\]](#) have reviewed these methods

among other data representation proposals. The authors highlight that PAA is extremely fast, running in $O(n)$ time, but it is not data adaptive. PLS have a $O(n \log n)$ complexity for the bottom-up approach (the most efficient of the three approaches [Esling and Agon 2012]). The disadvantage of PLS in the perspective of this thesis is the fact that the generated segments are not a constant value, actually the output of PLS methods are a series of pairs of points in the form (x_1, y_1, x_2, y_2) , which need further adaptation of its output for defining trajectory episodes. In the comparison performed in Section 3.3.2, we take the median value of each segment to mitigate this issue.

2.2.3 Spatial Annotation

Annotating trajectories with its spatial features is a basic and important task. For many applications, this allows building higher level abstractions that describe the spatial context of the path besides the simple latitude and longitude information.

Yan et al. [2012] have proposed a classification of spatial annotation types according to the geometric shape of the element to be annotated. They differentiate Region of Interest (RoI), Line of Interest (LoI), and Point of Interest (PoI) as possible entities that can be annotated.

The PoIs are stop episodes that are annotated with the help of an algorithm that computes the most probable type of PoI involved in that stop, e.g. restaurant, service, home. To achieve that, a Hidden Markov Model (HMM) based method was devised, being tailored to trajectories that take place at locations with high density of PoIs.

Many research about identifying visited places do not mention the annotation task as its objective, but these works can be used to this end. Andrienko et al. [2007] is one example where significant places are discovered through clustering of trajectories. In this case, trajectories are used to annotate a PoI.

In other research, [Furletti et al. \[2013\]](#) have proposed an algorithm for automatic annotating of trajectories with the user activities. PoIs at stops are inferred based on categories, opening hours, characteristics of the moving object (e.g. maximum walking distance), among other parameters. Then, a probability of the PoI being visited is computed.

Annotation of Lines of Interest in the work of [Yan et al. \[2012\]](#) concern basically in identifying correct road segments for constrained network movement of cars, and classifying modes of transportation for mixed movement (e.g. walking, public transport, cycling). The first challenge is handled with map matching techniques and part of its output is used along with average velocity and acceleration to infer transportation modes.

Regions of interest, or semantic regions, are identified in their framework as the result of the spatial join between trajectory and land use data. As a result of this process, it is possible to have a general coarse-grained view of the movement in terms of visited places.

Their *Trajectory annotation with RoIs* algorithm works by receiving as input a trajectory and a set of semantic regions. These regions can be either in free form (e.g. OpenStreetMap polygons), or in a grid-like format, which is the case of the [Swisstopo²⁸](#) dataset used in their framework. The algorithm then proceeds to compute the intersection among the inputted trajectory and all episodes and create tuples in the form $\langle region, entering\ time, leaving\ time, type\ of\ region \rangle$ that is after attached to the trajectory.

The authors highlight that the spatial join can be computed only for selected episodes depending on requirements. However, this aspect is not further explored. Moreover, it is implicit that all intersections of the trajectory and polygons or grid cells are computed regardless of the type of geographic feature of metadata associated to the external data, which may increase pro-

²⁸<https://www.swisstopo.admin.ch/>

cessing time. We note that while this can be useful to annotate episodes in a parameter-free setting and to discover information previously unknown, it is desirable to allow some kind of filtering for the cases of applications that are only concerned with a well-defined set of types of regions.

Contributions

Contents

3.1	STEP Ontology	50
3.1.1	Requirements	50
3.1.2	The QualiTraj ontology	54
3.1.3	STEP: S emantic T rajectory E pisodes	56
3.1.4	Updates on the STEP ontology	66
3.1.5	Structuring Trajectories with STEP	68
3.1.6	Querying Episodes	70
3.1.7	Evaluation	75
3.2	STEP Framework	81
3.3	Automatic Annotation Algorithms	85
3.3.1	MSN: M ove- S top- N oise Classification	85
3.3.1.1	Exploratory Data Analysis	86
3.3.1.2	Outlier Labeling Rule	91
3.3.1.3	The MSN algorithm	93
3.3.1.4	Evaluation	99
3.3.2	ANBA: A daptive N atural B reaks A pproximation	102
3.3.2.1	Fisher-Jenks Natural Breaks	103
3.3.2.2	The ANBA algorithm	104
3.3.2.3	Evaluation	110
3.3.3	ISA: I ntersection-based S patial A notation	117

In this chapter, we explain in detail the **STEP** ontology and framework. First, each ontology entity is described as well as their possible relationships. Then, the general architecture of the **STEP** framework is shown followed by three main algorithms for trajectory automatic annotation, namely the Stop-Move-Noise, Adaptive Natural Breaks Approximation, and Intersection-based Spatial Annotation with OpenStreetMap data.

3.1 STEP Ontology

Trajectory acquisition, management, and processing are important tasks for many applications that deal with spatiotemporal data. In order to perform these tasks effectively, it is important to rely on flexible structures. Many data models have been proposed for representing spatiotemporal traces. However, modeling trajectory characteristics and context information is still a challenge.

In this section, we introduce the **STEP** ontology (Semantic Trajectory Episodes) for trajectory enrichment. In order to model this domain, we structure trajectories and related contextual data in terms of semantic episodes that allow describing various characteristics of the traces and context along time and space dimensions. We demonstrate the usage of the **STEP** ontology for enriching raw trajectories and show how the proposed model may be useful for trajectory analysis tasks. We focus on the task of structuring moving object data taking into consideration its internal characteristics as well as the external context that surrounds the trajectory.

3.1.1 Requirements

In the development of ontology design patterns [Gangemi 2005], it is prescribed the description of Generic Use Cases (GUC) to guide pattern creation. The

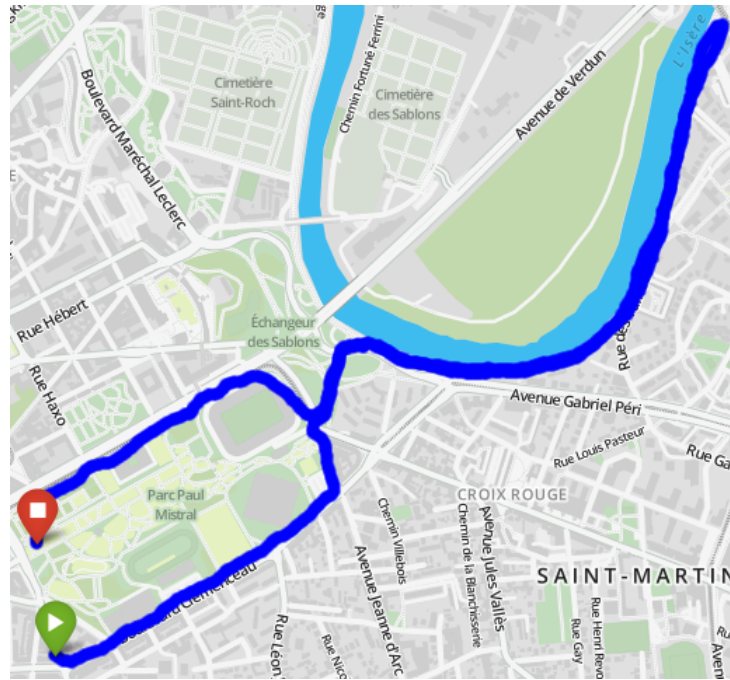


Figure 3.1: Trajectory of a runner. The person started the workout in a park, run next to a river, and then returned to the park.

concept of GUCs is linked to the idea of formulating competency questions, which are queries that a domain expert may run in a knowledge base and have a satisfactory answer if the data is structured following the ontology that is being designed. Although we have not developed an ontology design pattern, we have chosen this approach as starting point for building our ontology. In the following, we present the main competency questions that guided the development. For this exercise, we take the example of a trajectory collected during a running workout as depicted in Figure 3.1.

- How was the evolution of a given trajectory characteristic (e.g. speed, acceleration)?
- What was the runner's top speed?
- What was the trajectory's general shape? Does the person run in circles or was it a "back and forth" run?

- How was the weather during the workout?
- What relevant geographic features were nearby?
- What are the topological relationship between the trajectory and a given geographic feature (a park or a river)?
- Did the runner passed by some of its friends?

From the above competency questions, we can better define the scope of the desired data model. First, we need to represent what is “inside” the trajectory, i.e. what can be extracted from the elements that form the trace (for the case when a [GPS](#) is the source of information, these elements are latitude, longitude, and timestamps). It is also important to store information that spans some time interval (e.g. moments of high speed and moments of low speed) as well as events that might be represented as a single instant (e.g. the top speed is a single moment of the trajectory).

At the same degree of importance, we also notice the need for structuring information about features located in the surroundings of the trajectory as there may be strong links among them and the moving object’s path. Making these links are important to move beyond the trajectory [[Laube 2015](#)] in analysis tasks. The types of possible context data are very diverse, ranging from the kind of activity that was performed (e.g. running, commuting), the transportation means (e.g. car, public transport, bike, foot), the weather conditions, which Points or Regions of Interest were visited and how they were visited (what was the topological relationship between the Point or Region and the trajectory, e.g. cross, next, around).

To sum up the requirements about context data, we can identify two kinds of contextual information. First, the ones that are independent from the trajectory, e.g. the temperature, atmospheric pressure, the day of the week, the city where the trajectory took place. Second, information that relates

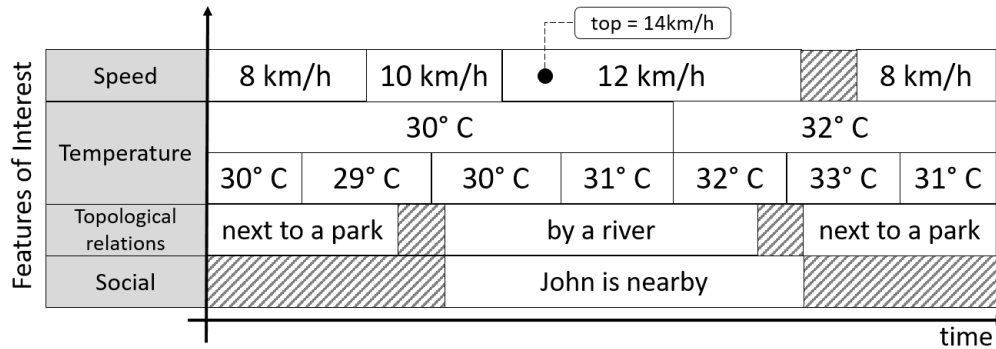


Figure 3.2: An example of multi-layered representation of trajectory and context based on episodes. Features of Interest are represented in the y axis and time is in x axis.

external objects (being these objects static or dynamic – a building or another trajectory) with the trajectory, e.g. the topological relationships between the trajectory and geographic features of a point, line or region of interest.

All of this information can have different levels of details. For some applications, simply expressing that the weather was sunny or rainy is enough, but for others, a more precise information may be important. For this reason, the ontology should allow the representation of data in different levels of granularity. Ilarri et al. [2015], for instance, argue that the semantic information attached to a trajectory should be represented at different levels of detail and it should be possible to easily commute from one to another based on the query requirements and existing privacy constraints.

Although not among practical queries for end users, additional metadata that may help in the preprocessing tasks can also be of interest for algorithms. For instance, data quality aspects like accuracy and sampling rate should be able to be represented with our ontology, for instance, to give an indication about the quality of other trajectory characteristics calculated from raw measurements.

In Figure 3.2, we show an example of features that can be expressed by

episodes with temporal extents, namely speed, temperature, topological relations, and direction. Although not exhaustive, this arrangement of information shows the usefulness of a representation that allows multiple layers of information to be combined. We can notice the need for representing both interval-based and instant-based episodes (e.g. the top speed information is an instant-based episode opposed to the remaining episodes). Another important aspect is the presence of qualitative and quantitative data, e.g. direction versus temperature. Also, some slots with diagonal lines represent periods of time without information about the feature in question. This can happen due to low quality or absence of positional data, or may of uninteresting for the application.

3.1.2 The QualiTraj ontology

Our first efforts in building an ontology for trajectory representation was done in the QualiTraj ontology [Nogueira and Martin 2014, Nogueira et al. 2014], which was built to express qualitative data of time-based characteristics of trajectories. The main QualiTraj entities are shown in Figure 3.3. Basically, the ontology is able to describe a trajectory that may have many profiles. Each **Profile** represents a dynamic characteristic of a **Trajectory** (e.g. speed, acceleration, direction) and is the entry point to the qualitative representation of the evolution of this characteristic over time. It was also added the **Global Attribute** entity linked to **Profile** in order to represent relevant information about the whole evolution of any trajectory or contextual element characteristic (e.g. the average speed of a trajectory).

Each **Profile** is composed by a sequence of **Segments**, which are qualitative representations of relevant changes of the characteristic over time. The kind of change is stored in the **Qualitative Value** (e.g. “Increase”, “Decrease”, “Steady”). The **Coefficient** serves to store the angle of the segment

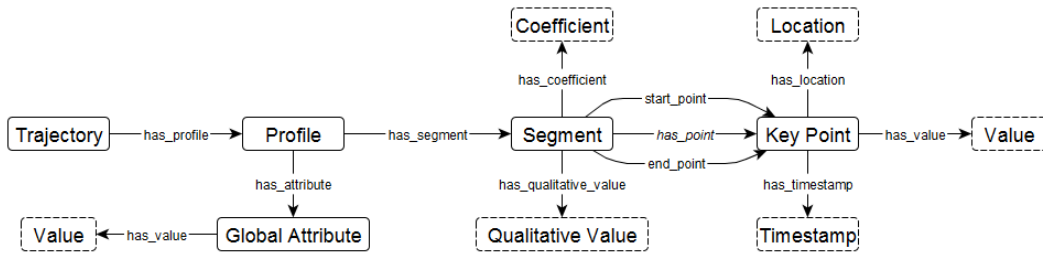


Figure 3.3: The QualiTraj ontology

line related to the x axis, as shown in Figure 3.4, the coefficient corresponds to the angle α . This coefficient can be useful if one wants to store the slope of the line and infer approximate values along the segment or to reconstruct the approximate evolution of a trajectory characteristic. Taking the example of Figure 3.4, if this speed evolution would have to be stored following the QualiTraj ontology, seven *Segment* instances would be created. The third one, for instance, would have a **Qualitative Value** attribute set to “Increase”, the value in degrees of the α angle would be the **Coefficient**, and two **Key Points** instances would be created having 7 seconds as **start time**, and 10 seconds as **end time** (or their respective timestamps).

This ontology have also introduced the concept of **Key Points**. These points are identified in space and time through **Location** and **Timestamp** attributes and may be used to represent important application-specific features of segments. For instance, due to the inherent loss of information, caused by qualitative transformation of raw data, a developer could store the highest and lowest quantitative value of a given characteristic for one or more segments. In this case, there would be two additional points for each **Segment** with labels such as **Highest speed** and **Lowest speed** and their respective values. These points would be represented by the optional relationship **has_point**. Although the **Key Point** entity gives a great flexibility to the user regarding which kind of quantitative information she wants to store, each segment must be linked to at least two points represented by the relationships **start_point**

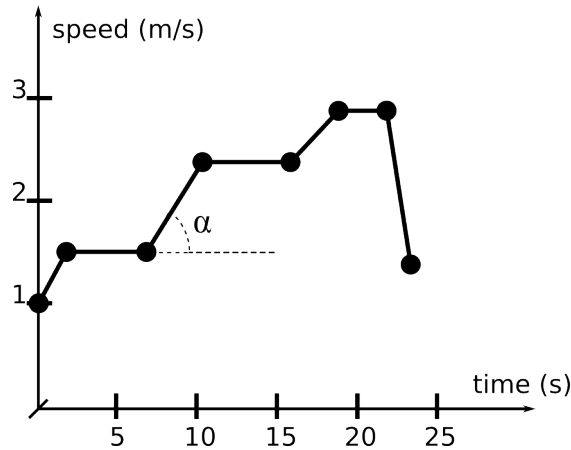


Figure 3.4: Example of abstract speed evolution of a trajectory.

and `end_point` in order to specify the times at which the segment starts and ends.

The QualiTraj model is focused on the description of qualitative aspects of trajectories. However, many of the requirements presented in section 3.1.1 are not addressed by this ontology. For instance, there is no support for a multilevel description of movement characteristics, as well as not entities to represent contextual elements. Moreover, the `Coefficient` entity can be considered as too restrictive, limiting the application possibilities to domain of time series representation. Another issue that we can highlight from QualiTraj concerns its lack of reuse of other available vocabularies for time and space concepts, for instance. These deficiencies have been addressed in the development of the `STEP` ontology, presented in the next section.

3.1.3 STEP: Semantic Trajectory Episodes

Based on the requirements presented in Section 3.1.1 and the previous work on semantic trajectories, we introduce in this section the `STEP` ontology for structuring trajectory and context into spatiotemporal episodes. A graphical representation of `STEP` is shown in Figure 3.8. We have used the Graffoo notation [Falco et al. 2014] and then transformed it into an `OWL` file. The offi-

cial namespace for the STEP ontology is <http://purl.org/net/step>, where eventual future updates of the ontology will be available. The first version of this ontology has been introduced first in our work [Nogueira and Martin 2015], shown in Figure 3.5. In this section, we explain the concepts that make part of this version and some changes that have been made since then, introducing a new version of the model.

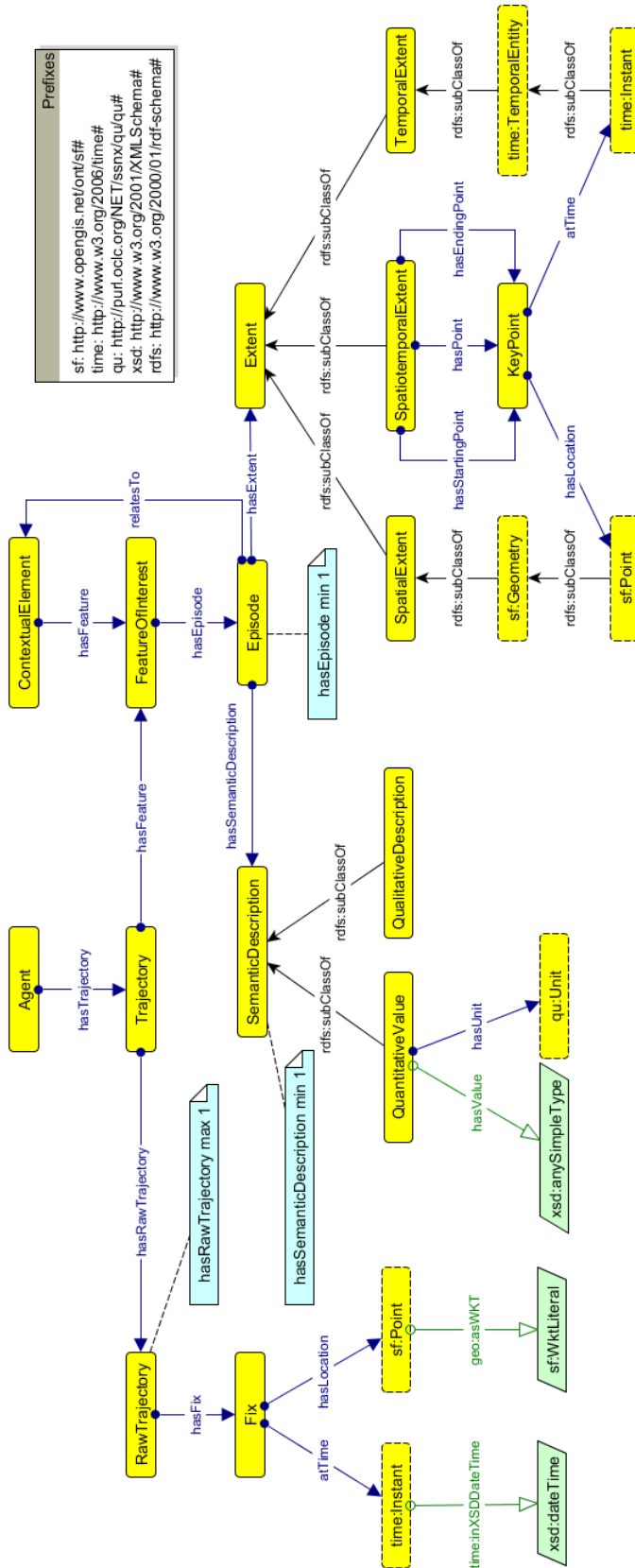


Figure 3.5: First version of the STEP ontology [Nogueira and Martin 2015]

The `Agent` class represents the moving object that performs the trajectory. For instance, in the case of human mobility, this entity could be an instance of `foaf:Person` from the Friend of a Friend ontology²⁹. Any `Trajectory` instance may have any number of `FeatureOfInterest` and only one `RawTrajectory`. It is important to notice that, unless otherwise explicitly declared, the cardinality of the relationships are “many to many”. In the case of `hasRawTrajectory` property, we clearly state that one `Trajectory` can have at most one raw trajectory representation through the `max 1` restriction. Having a `RawTrajectory` can be helpful for visualization purposes or if one wants to retrieve all trajectory fixes, but it may also lead to a heavy usage of storage space. Although we have left this possibility opened in the model, we have used the `RawTrajectory` class in practice only as a way of creating key points that are later explained for spatiotemporal extents. As a solution for the problem of representing all trajectory points in the `RawTrajectory`, one can store only significant points (after having computed all episodes that depend on all raw points) for plotting the spatial footprint of the moving object. This can be achieved by trajectory simplification methods, e.g. the Ramer–Douglas–Peucker algorithm [Ramer 1972].

We used a terminology similar to the ontology design pattern for semantic trajectories proposed by Hu et al. [2013] for representing spatiotemporal points. Basically, a `RawTrajectory` may have a series of fixes and each `Fix` is composed by a `Point` and a time `Instant`. We have reused the Simple Features ontology³⁰ from the Open Geospatial Consortium for general geometries and the W3C’s OWL:Time³¹ for referring to instants and intervals of time.

The other relationship that a `Trajectory` has is through the `hasFeature` property, which has a `FeatureOfInterest` as domain. This concept repre-

²⁹<http://xmlns.com/foaf/spec>

³⁰<http://www.opengis.net/ont/sf>

³¹<http://www.w3.org/2006/time>

sents the relevant aspects that an application using the **STEP** ontology wants to observe from two different sources of information: the agent's trajectories and the context elements. In the current version, **Trajectory** and **ContextualElement** were transformed into subclasses of the class **SpatiotemporalElement** as shown in Figure 3.8. By also linking **ContextualElements** to **FeatureOfInterest** through the **hasFeature** property, we simplify the ontology and make sure that the same structures (**Episodes**) can be used for both sources. In summary, one **Trajectory** or **ContextualElement** may have multiple **FeaturesOfInterest**, and each **FeatureOfInterest** may have many **Episodes**.

Episodes are the smallest semantic unity of the **STEP** ontology. This concept encapsulates values that a Feature of Interest may assume during the observed trajectory or contextual element together with an optional extent that delimits where and/or when that observation is valid. An **Episode** should have at least one **SemanticDescription**, which can be a quantitative value or a qualitative description. **QuantitativeValue** can be associated with different datatypes by **xsd:anySimpleType** (e.g. boolean, int, float) defined in the XML Schema³². However, letting only literal values to be associated are not enough to give rich semantics to **Episodes**. For this reason, a **qu:Unit** from the Library for Quantity Kinds and Units³³ or an equivalent ontology may be linked to **QuantitativeValue** instances in order to represent the measurement unit that have been used to express the value. This also avoids constructions like "50 km/h" as a value of a quantitative value, which would make queries harder to be constructed as the value would be mixed with the units in natural language.

Not all episodes can be described with literal values. Therefore, another option to give a semantic description to an **Episode** is through the **Qualita-**

³²<http://www.w3.org/TR/xmlschema-2>

³³<http://purl.oclc.org/NET/ssnx/qu/qu>

`tiveDescription` class. For instance, one can use this class to represent that, for a given episode, there were an increase of the speed. This is not a quantitative information, so it is not suitable for a `QuantitativeValue` instance. In this case, it is advisable to create an extension of `QualitativeDescription` that could better represent this data. Additionally, other qualitative data could be created for this case, e.g. `Decrease`, `Steady`. This kind of modeling allows the extension of the ontology to user-defined datatypes. Figure 3.8 shows examples of such extension with a small ontology that classifies stops and moves with subclasses of modes of transportation.

The `Extent` class allows to specify spatial and/or temporal limits for an `Episode`. Three subclasses can be used for representing extents, namely `SpatialExtent`, `TemporalExtent`, and `SpatiotemporalExtent`.

Spatial extents are used in cases where only the geometric aspect of the represented episode matters to the application. For instance, one could create episodes to describe land use data where the semantic description would be the type of usage and the extent would be a `SpatialExtent` specifying the limits where that usage is valid. We made `SpatialExtent` as superclass of `sf:Geometry` (see Figure 3.6), thus allowing that any Geometry subclass can be used to represent spatial extents, e.g. *Point*, *Line String*, *Surface*, etc.

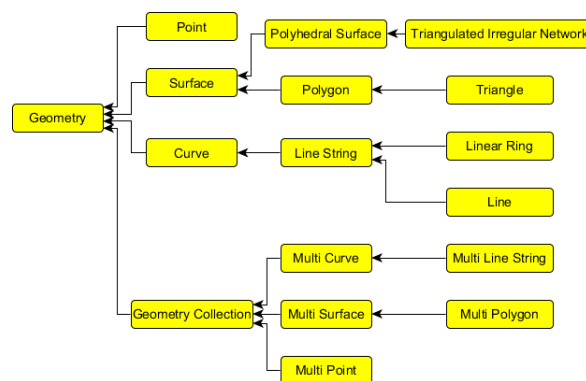


Figure 3.6: Part of the Simple Features ontology that defines Geometry subclasses.

Temporal extents are useful for cases where the temporal aspect of the modeled feature is important, e.g. a `ContextualElement` representing a commercial building may have a `FeatureOfInterest` with a label “Opening hours” having only a temporal extent describing when it is opened for clients. Time series is another example of a kind of data that fits well with this concept. For instance, an application can store temperature variations with episodes having only temporal extents. Possible temporal extents are instants and intervals, as can be seen in Figure 3.7.

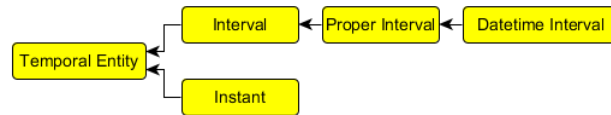


Figure 3.7: Part of the OWL:Time ontology that defines `TemporalEntity` subclasses.

The separation of spatial and temporal extents does not prevent the existence of both extents linked to the same `Episode`. Taking the above examples of land cover and temperature time series, there is no impediment in adding a temporal interval to describe for which period that geospatial location assumed the land usage description present on the corresponding `SemanticDescription`. This addition of temporal extent would allow the representation of land usage evolution over time. Correspondingly, in the second example, the spatial dimension of temperature variation (the changes in temperature in different geographic regions) may be interesting for some applications. Thus, a `SpatialExtent` (e.g. a polygon) can be linked to the `Episode` instance.

The `SpatiotemporalExtent` class holds a more complex structured compared to the other two `Extent` subclasses and is meant to delimit episodes in terms of points present in the trajectory itself. This allows delimiting trajectory segments or subtrajectories. An extent of this type is characterized by having one or more *KeyPoints*, a concept that we introduced in QualiTraj [No-

gueira et al. 2014] and slightly modified in the STEP ontology. Similarly to QualiTraj, we connect KeyPoints to Episodes in three possible ways: the properties hasStartingPoint, hasEndingPoint, and hasPoint.

Due to the strong association of the QualiTraj ontology for representing time series data, it enforces a constraint of always connecting each Segment (concept equivalent to Episodes) to a starting and to an ending KeyPoints. In the STEP ontology, we do not have this restriction, therefore allowing an Episode to have a SpatiotemporalExtent with only one KeyPoint through a hasPoint relation or even having a KeyPoint as the whole SpatiotemporalExtent of an Episode (this last possibility was included only in the current version of STEP). Therefore, we have the possibility to model events that span for some duration and distance and also events that happen instantly and in a single spatial point.

Despite the fact that the STEP ontology has been developed with flexibility in mind, some restrictions are still needed. First, the starting KeyPoint must precede the ending KeyPoint (similar to the OWL:Time's ProperInterval constraint). Second, there cannot be a starting KeyPoint without an ending KeyPoint and vice-versa. This has been modeled as SWRL rules, shown in the next sections. Another important restriction applies to TemporalExtent as the temporal boundaries of each episode must be within the global temporal duration of the corresponding Trajectory or ContextualElement.

Usually, different extents cannot overlap regarding episodes of the same FeatureOfInterest. Situations like defining one Episode during which the speed was $30km/h$ for the first 5 minutes, and another Episode stating that the speed was $40km/h$ for the first 3 minutes, for instance, does not make sense in the real world. However, it would be normal to have two temporal extents with different specializations (interval and instant). For example, an episode that represents the top speed of a moving object can overlap an interval-based

episode that represent the average speed over a time interval (see Figure 3.2).

The property `relatesTo` was added to make possible relating an episode to a contextual element. This is especially important for representing topological relations of trajectories. For instance, an episode of passing next to a park would have the topological relation “next to” as a semantic description and the `relatesTo` property would link this episode to an instance of the park where the moving object has passed by. This property can also be used to model relations among one trajectory with trajectories of other moving objects, as these can be considered as contextual elements.

It can be useful for some applications to store global values that are valid for an entire trajectory or for the entirety of a contextual element. In the first version of `STEP` (Figure 3.5), it has been suggested that a `FeatureOfInterest` was created and linked to an `Episode` without a `hasExtent` relating it to any `Extent`. Then, the global attribute value would be normally represented by one of the `SemanticDescription` subclasses. This was done because the property `hasSemanticDescription` had a cardinality restriction (`min 1`), which would force each `FeatureOfInterest` to have a `SemanticDescription`. However, this constraint was removed in order to simplify the addition of descriptions that range the entire trajectory. Now, a `SpatiotemporaElement` can have a `FeatureOfInterest` (e.g. “Average speed”) having a direct link to a description.

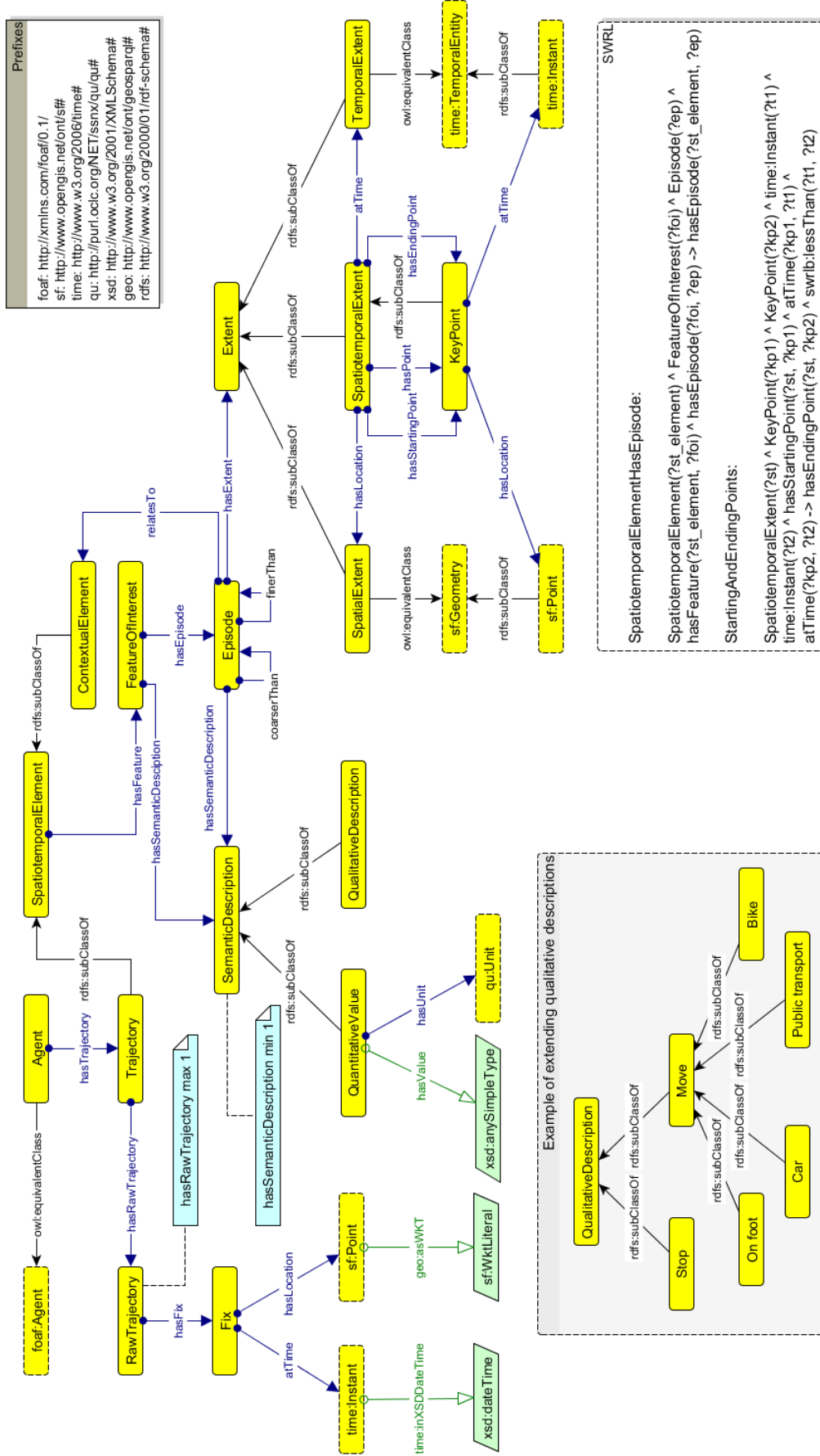


Figure 3.8: Overview of the STEP ontology. Concepts from other ontologies are indicated by their namespaces and dotted rectangles, diamond-shaped nodes are datatypes, notes represent class restrictions.

3.1.4 Updates on the STEP ontology

As we can notice from the description of the entities and Figure 3.5, the first version of the **STEP** ontology is general enough to represent a large variety of spatiotemporal information from several domains. However, it still did not cover some desired requirements for trajectory modeling. Its main deficiency is the lack of explicit support for granularity levels. This has been corrected in the version shown in Figure 3.8 and are detailed in the following.

KeyPoint is now a subclass of **SpatiotemporalExtent**, allowing a more direct definition of point-like events. In the previous version, it was necessary to create an instance of **SpatiotemporalExtent** that would have **KeyPoint** through the **hasPoint** property. This case is still useful for representing relevant information inside an episode (e.g. the top speed during an interval).

The new object properties **coarserThan** and **finerThan** allows the creation of hierarchies of episodes. Both are transitive, meaning that if *Episode A* is coarser than *Episode B* which is coarser than *Episode C*, then it can be inferred that *Episode A* is coarser than *Episode C*. Additionally, **coarserThan** is the inverse of **finerThan** (`owl:inverseOf`).

SpatiotemporalElement is introduced. It has subclasses **Trajectory** and **ContextualElement** and concentrates the **hasFeature** property domain. The **relatesTo** property changed its range to **SpatiotemporalElement**, which allows linking a trajectory to other trajectories easily (e.g. expressing that a trajectory crosses another one).

The property **hasSemanticDescription** can link a **SpatiotemporalElement** directly to a qualitative description of quantitative value without needing the creation of an **Episode**. This is useful for representing attributes that are valid for the entire span of the spatiotemporal element. This change removes the restriction of creating at least one episode with the same extent of the trajectory as has been discussed in the previous section.

We have defined our own classes for `Geometry` and `TemporalEntity` and then we have aligned them to Simple Features and OWL:Time ontologies, respectively, through the `owl:equivalentClass` property, which links a class description to another class description in a way that the two class descriptions involved have the same class extension (i.e., both class extensions contain exactly the same set of individuals) [W3C 2004]. This removed the subclassing of entities pertaining to external ontologies. This conceptual alignment was also performed in order to specify that an `Agent` is equivalent to entities of the class `foaf:Agent` that is present in the FOAF vocabulary. Although FOAF subclasses `Agent` with only `Person`, `Organization`, and `Group`, it is intended to cover a larger group of entities as it is evidenced in its official documentation³⁴ in the following comment about the `Agent` class: “*An agent (eg. person, group, software or physical artifact).*”

Finally, SWRL rules have been added to allow inferencing about some facts in query time. First, when a `SpatiotemporalElement` has a `FeatureOfInterest` and this feature has an `Episode`, a reasoner can retrieve graph patterns as if the spatiotemporal element had a direct link to the episode. This rule is shown in Definition 1.

Definition 1. *SWRL Rule SpatiotemporalElementHasEpisode*

$$\text{SpatiotemporalElement}(?st) \wedge \text{FeatureOfInterest}(?foi) \wedge \text{Episode}(?ep) \wedge \text{hasFeature}(?st, ?foi) \wedge \text{hasEpisode}(?foi, ?ep) \Rightarrow \text{hasEpisode}(?st, ?ep)$$

Another rule that has been added refers to the constraint that we mentioned in the previous section about enforcing that if a `SpatiotemporalExtent` has a starting point, it must have an ending point. This rule also assures that the timestamp of the starting point should have a value that is earlier than the one of the ending point. The formalization in SWRL of this rule can be seen in Definition 2.

³⁴<http://xmlns.com/foaf/spec/20140114.rdf>

Definition 2. SWRL Rule StartingAndEndingPoint

$$\text{SpatiotemporalExtent}(?st) \wedge \text{KeyPoint}(?kp1) \wedge \text{KeyPoint}(?kp2) \wedge \text{time:Instant}(?t1) \\ \wedge \text{time:Instant}(?t2) \wedge \text{hasStartingPoint}(?st, ?kp1) \wedge \text{atTime}(?kp1, ?t1) \wedge \text{atTime}(?kp2, ?t2) \Rightarrow \text{hasEndingPoint}(?st, ?kp2) \wedge \text{swrlb:lessThan}(?t1, ?t2)$$

These rules can help developers to write simpler queries and assure model consistency if a SWRL-compatible reasoner is available to the application and/or triple store. For instance, a SPARQL query may need additional graph patterns to first retrieve the features of interest of a trajectory that matches some label (e.g. “Stop/Move”) and then retrieve episodes with the desired condition. However, thanks to the introduction of the rule described in Definition 1, a developer can write a query as if a `SpatiotemporalElement` had episodes directly linked to it. This functions as a ‘shortcut’ for smaller queries while keeping the requirements on domain and range of `hasFeature` and `hasEpisode` untouched. In the following section, we focus on practical usage of the ontology in SPARQL queries and instantiation examples.

3.1.5 Structuring Trajectories with STEP

Having defined our ontology, we now show some possibilities to instantiate it in order to represent trajectories and contextual information.

Creating episodes depends on what Features of Interest are important for each application. Usually, episodes of a Feature of Interest will be constructed by following some algorithm that takes the trajectory and/or external context data, does some computation based on some criteria, and records the episodes in the triple store following the STEP ontology specification, as it is the case of the second part of this thesis that deals with automatic annotation algorithms.

In the following, we exemplify the creation of episodes that represent the speed evolution of a moving object and some geographic features located around its trajectory. These Features of Interest are good examples of des-

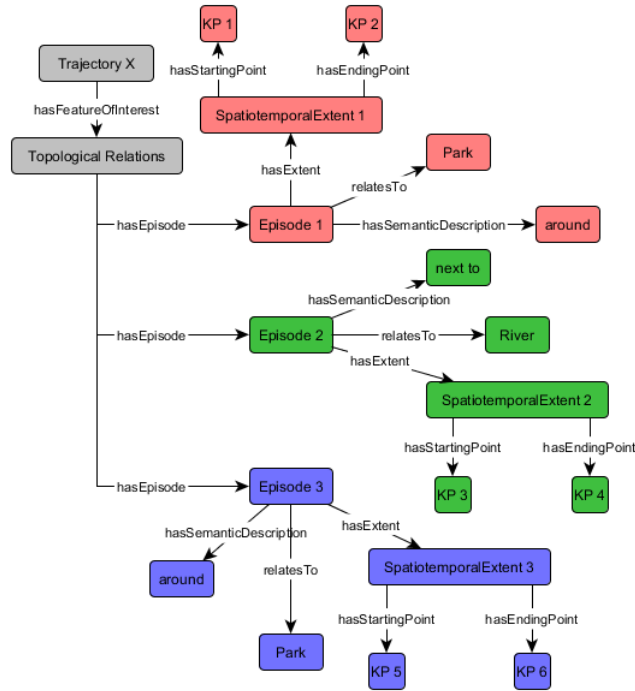


Figure 3.9: An excerpt example of STEP ontology instances. The Feature of Interest in this case is the topological relations of the trajectory and nearby geographic features.

cribing internal aspects of trajectories as well as contextual data that comes from other data sources. At this point, we show a rather naive approach to speed segmentation as our focus in this section to explore STEP episodes. A more effective solution will be presented in the following sections of the thesis.

To create episodes that represent speed, one can calculate the speed at each point of the trajectory by computing the approximate distance between two pairs of latitude and longitude using the Haversine formula to calculate distance and the sampling interval between points to calculate the elapsed time. Then, speed data can be distributed into bins of equal size ranging from the minimum speed to the maximum speed. Then, each speed value can be classified into its respective bin.

To give a simple example, consider that we have the following speeds (in

meters per second): [4, 1, 0.5, 2.9, 1, 1.7, 1.5, 1.5]. The minimum speed is $0.5m/s$ and the maximum speed is $4m/s$. If we want to have five equally spaced bins, we take the range (difference between the maximum and minimum values), which is 3.5, and divide by 5, which gives an increment of 0.7 (the width of each bin). Therefore, each speed value is going to fit into one of the following bins: [0.5 to 1.2), [1.2 to 1.9), [1.9 to 2.6), [2.6 to 3.3), and [3.3 to 4]. This allows the clustering of similar speeds, which causes a reduction in the number of episodes. While this is an illustrative example of a quite simple method, a better procedure, more data adaptive, is presented later in this thesis.

The discretization also allows the qualitative representation of data. For instance, considering that speed values are normalized among trajectories, one could assign labels like “slow”, “average”, and “fast” for the bins.

In another example, consider that we have an algorithm that identifies the topological relationship between a trajectory and geographic features (e.g. intersect, touch). The output of this method could be used to create episodes as shown in Figure 3.9, which uses the trajectory of Figure 3.1 to express possible instances of **STEP** that represent geographic relations between a trajectory and its context. A more elaborated discretization method and contextual annotation procedure are presented in our framework in later chapters.

3.1.6 Querying Episodes

In this part, we demonstrate how a graph containing instances of the **STEP** ontology can be queried in order to answer questions related to those that have been described in the requirements of section 3.1.1. For instance, Listing 3.1 shows a query that identifies if a runner is following a specific program called Interval training or fartlek. In this kind of workout, the runner has to alternate periods of slow and very high speeds for a predefined number of

times.

```

1 PREFIX : <purl.org/net/step#> .
2 SELECT ?trajectory
3 WHERE
4 {
5     ?trajectory :hasFeature ?foi .
6     ?foi rdfs:label "Speed"@en ;
7         :hasEpisode ?ep1 ;
8         :hasEpisode ?ep2 ;
9         :hasEpisode ?ep3 ;
10        :hasEpisode ?ep4 .
11
12    ?ep1 :hasSemanticDescription "slow" ;
13         :hasSpatiotemporalExtent ?extent1 .
14
15    ?ep2 :hasSemanticDescription "very high" ;
16         :hasSpatiotemporalExtent ?extent2 .
17
18    ?ep3 :hasSemanticDescription "slow" .
19         :hasSpatiotemporalExtent ?extent3 .
20
21    ?ep4 :hasSemanticDescription "very high" ;
22         :hasSpatiotemporalExtent ?extent4 .
23
24    ?extent1 :hasEndingPoint ?end1 .
25    ?extent2 :hasStartingPoint ?start2 .
26    ?extent2 :hasEndingPoint ?end2 .
27    ?extent3 :hasStartingPoint ?start3 .
28    ?extent3 :hasEndingPoint ?end3 .
29    ?extent4 :hasStartingPoint ?start4 .
30
31    FILTER(?end1 < ?start2 && ?end2 < ?start3 && ?end3 < ?start4)
32 }
```

Listing 3.1: SPARQL query to detect a speed pattern

The query in Listing 3.1 assumes that the episodes have been labeled in a range between “very high” and “slow”. In this example, we search for trajectories that present a pattern of slowing down and going very fast twice. We enforce temporal sequence with the *filter* expression, which restrict solutions

to only those that evaluate to true.

Another query example is shown in Listing 3.2 and it is related to contextual data. In this example, we ask the following question: “*How much of people’s workouts takes place near parks?*”. To answer this, a query similar to Listing 3.2 could be formulated. Taking as example the instances of Figure 3.9, this query would return the following pairs of *KeyPoints*: (*KP1*, *KP2*) and (*KP3*, *KP4*) as starting and ending points, respectively.

```

1 PREFIX : <purl.org/net/step#> .
2 SELECT ?startingPoint, ?endingPoint
3 WHERE
4 {
5     ?trajectory :hasFeature ?foi .
6
7     ?foi rdfs:label "Topological relations"@en ;
8         :hasEpisode ?episode .
9
10    ?park rdfs:label "Park" .
11
12    ?episode :relatesTo ?park ;
13            :hasSemanticDescription "nearby" ;
14            :hasStartingPoint ?startingPoint ;
15            :hasEndingPoint ?endingPoint .
16 }
```

Listing 3.2: Query for retrieving the points of the trajectory during which a park was nearby.

Then, a simple algorithm would be able to calculate how much time was spent around parks. It is important to notice that we have used *rdfs:label* to match nodes. Other approach could be using domain ontologies (e.g. types of geographic features) instead of matching against the “park” string. These types of queries can reveal some preferences of a person regarding the places where he or she generally goes to exercise, for example.

In the following example, we compare the same query written using the

Baquara² and STEP ontologies (see Listing 3.3). The query is the same presented in Fileto et al. [2015b] described as *“Select the social media user’s trails with at least one stop to visit a mountain called Corcovado in the city of Rio, followed by one stop in a marketplace, where he/she does at least one finer stop in a restaurant”*.

This example shows the differences in representing the granularity of the two models, which can be considered equivalent. The main difference lies in the fact that Baquara² uses an order attribute that should be attached to an entity. While this can be automatically computed at annotation time, we believe that our approach of using the timestamps of each episodes is more efficient as it uses an information that is intrinsic to the episodes in question. If dates are stored as XSD datetime format, direct comparison is supported by the SPARQL standard. On the other hand, using a literal value such as “order” may raise problems if episodes should be reorganized due to the inclusion of a new episode, for instance.

In Listing 3.4, we show a query that has been described by Bogorny et al. [2014] as *“How many tourist trajectories stayed at the Colosseum when it was raining and there was a photographic exhibition?”*. We verify that, despite the differences between CONSTAnT and STEP the query can be equally answered by both models.

However, our ontological approach allows us to reuse concepts from other ontologies and enables referencing real world entities in a unique way. For instance, pointing to `dbo:Colosseum` assures that we are referencing the tourist attraction in Rome, and not some restaurant or night club that may have the same name. It can also be argued that the SPARQL query is more readable, although this is a rather subjective judgment.

```

PREFIX bq: <http://www.seek-project.eu/Baquara02>
PREFIX step: <http://purl.org/net/step>
PREFIX schema: <http://schema.org/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX lgdo: <http://linkedgedata.org/ontology/>

SELECT ?t WHERE {
  ?t a bq:Trail .
  ?sc bq:father ?t ;
    a bq:Stop;
    bq:ord ?sc_ord;
    bq:visits ?corcovado .
  ?corcovado a schema:Mountain ;
    rdfs:label "Corcovado" ;
    dbp:location ?rio .
  ?rio a dbp:City ;
    rdfs:label "Rio de Janeiro" .
  ?sm bq:father ?t ;
    a bq:Stop;
    bq:ord ?sm_ord;
    bq:visits ?mp .
  ?mp a lgdo:Marketplace .
  ?sr bq:father ?sm ;
    a bq:Stop;
    bq:visits ?r .
  ?r a lgdo:Restaurant .

  FILTER ?sc_ord < ?sm_ord
}

SELECT ?t WHERE {
  ?t a step:Trajectory .
  step:hasEpisode ?sc ;
  step:hasEpisode ?sm .
  ?sc step:hasQualitativeDescription "Stop" ;
  step:relatesTo ?corcovado ;
  step:hasExtent ?sc_extent .
  ?corcovado a schema:Mountain ;
  rdfs:label "Corcovado" ;
  dbp:location ?rio .
  ?rio a dbp:City ;
  rdfs:label "Rio de Janeiro" .
  ?sc_extent step:hasEndingPoint ?sc_end.
  ?sc_end step:atTime ?sc_end_time .
  ?sm step:hasQualitativeDescription "Stop" ;
  step:relatesTo ?mp ;
  step:hasExtent ?sm_extent .
  ?mp a lgdo:Marketplace .
  ?sm_extent step:hasStartingPoint ?sm_start .
  ?sm_start step:atTime ?sm_start_time .
  ?sr step:hasQualitativeDescription "Stop" ;
  step:relatesTo ?r ;
  step:finerThan ?sm .
  ?r a lgdo:Restaurant .

  FILTER ?sm_start_time > ?sc_end_time
}

```

Listing 3.3: Equivalent queries in Baquara² (left) and STEP (right)

```

SELECT count(sp.tid)
FROM SemanticPoint sp JOIN Environment e,
SemanticPoint sp JOIN Place p,
Place p JOIN Event v
WHERE e.name = "weather" AND
e.value = "raining" AND
p.name = "Colosseum" AND
v.name= "photographic exhibition"

```

```

PREFIX step: <http://purl.org/net/step>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT COUNT(?t)
WHERE {
?t step:hasEpisode ?ep .
?ep step:relatesTo dbo:Colosseum ;
   step:relatesTo ?weather
   step:relatesTo ?photoExhibition .
?weather a step:ContextualElement ;
   rdfs:label "Weather" ;
   step:hasFeatureOfInterest ?condition .
?condition step:hasQualitativeDescription "Rainy" .
}

```

Listing 3.4: Equivalent queries in CONSTAnT (top) and STEP (bottom)

3.1.7 Evaluation

Comparing models can be a subjective task because some modeling solutions may seem simplistic but, at the same time, they can be sufficient for a range of applications, while other solutions may be complex and suited for specific application needs. We have selected some criteria to compare trajectory conceptual models and ontologies based on the features that our model has that differs most from others. While this should not be used to rank the STEP ontology above other proposals, it serves to show that our model gathers a

set of features that is only partially covered by other models. Furthermore, some of the criteria cannot be categorized as a Yes/No question, but in a more detailed scale. This comparison is summarized in Table 3.1 and we detail each criteria in the following.

Table 3.1: Comparative table of trajectory models

Model	Main Segmentation Concept	Support for Granularity	Support for Generic Extents	Support for Generic Contextual Elements	Type	Online Availability
Spaccapietra et al. [2008]	Stop-Move	No	No	No	Conceptual	N/A
Baglioni et al. [2008; 2009]	Stop-Move	No	No	No	Ontology	No
Hu et al. [2013]	Segment	No	No	No	Ontology design pattern	No
Fileto et al. [2015b] (Baquara ²)	Episodes	Yes	No	Yes	Ontology	No
van Hage et al. [2012] (SEM)	Events	Yes	No	Yes	Ontology	Yes
Yan et al. [2008]	Stop-Move	No	Yes	No	Ontology	No
Andrienko et al. [2011]	Events	No	Yes	Yes	Conceptual	N/A
Bogorny et al. [2014] (CONSTAnT)	Subtrajectory	No	No	Yes	Conceptual	N/A
STEP	Episodes	Yes	Yes	Yes	Ontology	Yes

Segmentation

This criteria refers to the ways of expressing parts of trajectories that are provided by the model to the user. The chosen term per se is not important (i.e. either events, episodes, subtrajectories can be considered semantically similar), but how the model allows a user to select specific trajectory segments to add some semantic information to it.

We can observe that some models have a limited power of expressing trajectory segments, namely the ones that provide only the fundamental two concepts of Stop and Move. While these concepts form the basis of trajectory analysis, this limitation is useful for application that are more concerned with the periods of stop. On the other hand, models that provide more generic entities like Events, Episodes, and Subtrajectories can be considered more

adaptable to new application needs.

In the case of **CONSTAnT**, subtrajectories have a disadvantage of limiting this kind of segmenting only for trajectory related attributes. Therefore, contextual elements must always be linked to some subtrajectory, while it could be more interesting to give the opportunity of representing this information independently.

STEP provides a higher level of organization with features of interest that is not present in **SEM**. This also provides a way of expressing global attributes that are harder to represent with **SEM**. For instance, how one could represent the average speed of a trajectory in **SEM**? While this is not impossible, this shows that **SEM** is more tailored to represent qualitative aspects like “speeding up” or “slowing down”, for example.

Granularity

As defined by **Hornsby and Egenhofer [2002]**, “granularity refers to the notion that the world is perceived at different grain sizes”. In our context, granularity refers to the power of expressing different levels of detail of episodes both in time and space and it is advisable that the trajectories’ semantic information should be represented at different levels of detail [**Ilarri et al. 2015**]. The representation of diverse granules of information is then an important requirement for trajectory modeling and spatiotemporal phenomena in general [**Silva et al. 2015**].

The Baquara² and **SEM** ontologies are the only current models that provide constructs for representing hierarchies of segments. In Baquara², a trajectory, trail or episode may have a *father* and one or more *children*. Additionally, movement segments of Baquara² should contain numeric information regarding their level (the distance between the segment and its farthest father). In the case of **SEM**, an Event may have sub-events. In **STEP**, we also support hierarchy of episodes through the **coarser** and **finer** properties, that

are transitive and inverse.

Extent

This criteria is related to the support of representing different kinds of temporal and spatial extents and their possible combinations. The most important type of extent in trajectory modeling is the spatiotemporal one, i.e. the ability of defining the locations and timestamps in the trajectory where some homogeneous criteria is met. However, a model that separates these two components of spatiotemporal extent can benefit of expanding the possibilities of representation that do not involve one of temporal or spatial aspects.

The ontology of [Baglioni et al. \[2008; 2009\]](#) focuses entirely in stops, thus, it is only possible to associate a place (through the `is_at` property). Also, a `Stop` may be associated to a `Time` entity, but this is not the case of a `Move`. The Spatiotemporal Ontology of [Yan et al. \[2008\]](#) have expanded the possibilities of defining extents by combining spatial and temporal ontologies to provide concepts for describing phenomena with time-varying geometry. While this proposal opens up a wide range of possibilities, when they integrate this ontology with their Trajectory Ontology, it is specified that a stop is limited to a combination of point-like geometry and time interval ($\subseteq \exists hasGeometry.Point \cap \exists hasTime.Interval$) and a move is a list of Time-Points (point and instant combination), therefore, it is not possible to express more complex extents for these episodes (e.g. defining a stop as a region instead of a point).

In Baquara², we have that a Movement Segment is a tuple containing geometry, initial and final positions of the corresponding subsequence, and a time span. It is not clear from their work how this is modeled into the ontology as the authors present these attributes in an abstract way. If these properties can be instantiated independently in the annotation process, then their approach is similar to ours for this criteria.

The extent of Subtrajectories are delimited by start and end time in `CONSTANT`, thus, defining the extent of this entity as a temporal interval. The model also allows linking a Subtrajectory to one or more Semantic Points. This idea is close to the `KeyPoint` concept of `STEP` and allows a finer representation of trajectory events. A drawback of this conceptual model is the difficulty of expressing aggregated data, e.g. average speed during a given segment.

Our `STEP` model allows the representation of episodes in a way that each episode can have an extent that can be of three different main types: spatial-only, temporal-only, or spatiotemporal, as described in subsection 3.1.3. Moreover, we have followed an approach similar to Yan et al. [2008] in terms of reusing other ontologies and composing the extent of episodes based on a combination of concepts. However, we have left the possibilities opened for any combination of GeoSPARQL's `Geometry` and OWL:Time's `TemporalEntity`.

Context

It is natural that the spatial context takes greater attention in trajectory modeling. The conceptual model of Spaccapietra et al. [2008] already proposed to link begin, end and stops to spatial objects representing the corresponding location in terms of application objects. Nevertheless, contextual information may span a wider range of possibilities as the spatio-temporal context consists of the space, time and other objects positioned in the space and/or time [Andrienko et al. 2011].

It is important to note that, for the matter of the comparison in Table 3.1, we consider that a model supports context representation if there is some concept beyond locating trajectory points in time and space because, in some cases, we verify that some extension of the ontology is needed to represent the relationship of the trajectory with contextual elements.

In the work of Yan et al. [2008], the representation of contextual elements

is delegated to Geography and Application ontologies that can vary according to the domain. While the modularization is an important characteristic, we believe that it is possible to provide generic entities to enrich trajectories with contextual information. These generic entities can also serve as an extension point if more complex ontologies are needed, but they can be sufficient in some cases.

The **CONSTANT** model counts with a more advanced support for contextual elements when compared to other trajectory-specific models as it provides an Event class that is able to express more information about a place. However, this link to a place instance limits its extent to a spatial entity.

One advantage of the **STEP** model is that **ContextualElement** can have episodes with their corresponding extents that are completely independent from any trajectory instance. This allows the representation of spatiotemporal phenomena that can be later linked to trajectories or can coexist in a decoupled manner but can be matched in querying time. A similar feature is only identified in **SEM** due to its lightweight characteristic. Moreover, diverse contextual element can be represented, e.g. weather, places of interest, other trajectories, having the structure of episodes with flexible extents and semantic descriptions.

Type and availability

We have also classified each mode either as a conceptual model, ontology or design pattern in order to identify the status of implementation of the ideas of their respective authors. This is also important because the models that are modeled as ontologies count with a controlled standardized formal vocabulary. Lastly, it is important to notice that, due to the power of extensibility of ontologies, a simple ontology extension or new version can solve some model deficiencies.

Another point that can speed up adoption of a model by an applica-

tion is its online availability. This applies only for ontological models in Table 3.1. At the time of this writing, only the **SEM** and **STEP** ontologies are readily available, i.e. they can be found and downloaded, for instance, from the Linked Open Vocabularies [Vandenbussche et al. 2016] repository (<http://lov.okfn.org/dataset/lov/vocabs/sem> and <http://lov.okfn.org/dataset/lov/vocabs/step>) besides their respective websites.

3.2 STEP Framework

The **STEP** framework proposed in this thesis serves as a bridge between the **STEP** ontology and the automatic annotation algorithms. This layer consists basically into a mapping of the ontology entities into classes that follow the object oriented paradigm and a series of general purpose utility methods that help the application of annotation algorithms. An overview of the framework can be viewed in Figure 3.10 where the highlighted modules represent the parts that have been implemented in this thesis.

Our framework contains a preprocessing layer that handles raw **GPS** data, i.e. a sequence of tuples consisting in latitude, longitude, and timestamp records. The current version of the framework works with the popular GPS Exchange Format (**GPX**)³⁵ file format but it can easily adapted to consume other formats like **TCX**, **CSV**, etc. From this simple sequence of tuples, our preprocessing layer computes the following attributes of each point:

- time duration from previous point in seconds by the simple difference of timestamps between the two points.
- distance from previous point in meters by the haversine formula, a function that computes the distance between two points in a sphere, as defined in Equation 3.1.

³⁵<http://www.topografix.com/gpx.asp>

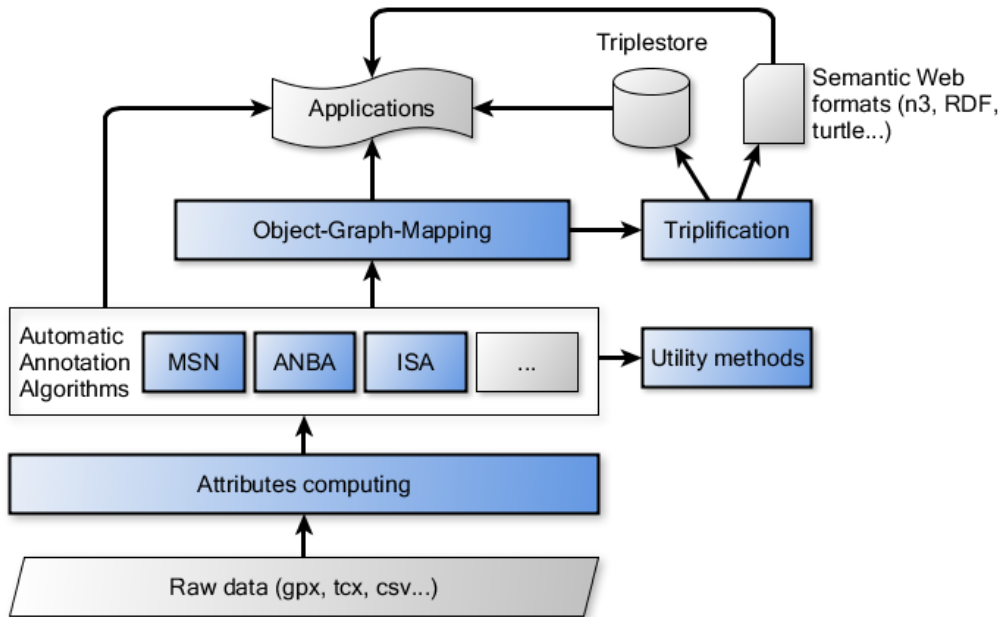


Figure 3.10: Overview of the STEP framework. The colored blocks were implemented during this thesis.

$$d = 2r \arcsin \left(\sqrt{\sin^2 \left(\frac{\varphi_2 - \varphi_1}{2} \right) + \cos(\varphi_1) \cos(\varphi_2) \sin^2 \left(\frac{\lambda_2 - \lambda_1}{2} \right)} \right) \quad (3.1)$$

- elevation in meters of each point retrieved by the Shuttle Radar Topography Mission (SRTM)³⁶, as a way to provide a more accurate elevation data.
- speed in meters per second by the relation between the traveled distance and the time spent to traverse two points.
- acceleration in meters per second squared by the difference of speed between each point and its previous neighbor.

³⁶<http://www2.jpl.nasa.gov/srtm/>

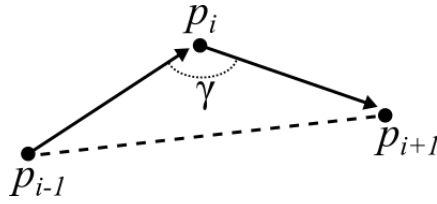


Figure 3.11: For each three points of a trajectory, we calculate the angle formed by them to get the turning angle.

- heading in degrees, the direction towards the object is theoretically moving at each timestamp, its values ranges from 0 to 360.
- turning angle in degrees. We have that, for a triad of successive points p_{i-1}, p_i, p_{i+1} , the turning angle consists on the angle formed by these three points where γ is the angle having p_i as vertex (see Figure 3.11). The triangle edges are equivalent to the distances $\overline{p_{i-1} p_i}$, $\overline{p_i p_{i+1}}$, and $\overline{p_{i-1} p_{i+1}}$. We have applied the Rule of Cosines for unknown angles (Equation 3.2) to find the angle between two successive GPS points.

$$\gamma = \arccos \left(\frac{\overline{p_{i-1} p_i}^2 + \overline{p_i p_{i+1}}^2 - \overline{p_{i-1} p_{i+1}}^2}{2 \times \overline{p_{i-1} p_i} \times \overline{p_i p_{i+1}}} \right) \quad (3.2)$$

Other utility methods of the framework include, for instance, array manipulation procedures to group sequential values, e.g. given an array like [1, 2, 3, 4, 15, 16, 17, 18, 50, 51, 52, 53, 54, 55], the method returns the a series of pairs containing the limits of contiguous groups, e.g. [[1, 4], [15, 18], [50, 55]]. This procedure is essential to find which points of the trajectory delimit the episodes created by the framework.

Another important auxiliary procedure of the STEP framework consists in creating intermediate instances of object-oriented representation of the trajectory points in the form of the `RawTrajectory` concept and its components. This method takes a raw GPX representation and create a `RawTrajectory` object that serves as a container for the points that delimit the episodes crea-

ted by the framework. Similarly, our framework also have the ability of taking a time series and creating a `FeatureOfInterest` concept. The framework has been implemented in the Python 3.5 language and the following main packages: `pandas`³⁷, `numpy`³⁸, `gpxpy`³⁹, `srtm`⁴⁰, `shapely`⁴¹, `overpy`⁴², `rdflib`⁴³, among others.

In the Object-Graph-Mapping layer (see Figure 3.10), each class representing a `STEP` ontology entity contains a method for outputting a JSON representation of it as well as a `triplify()` function that returns an intermediate representation of the graph of triples that can be then serialized to a Semantic Web compatible file format such as `n3`, `RDF/XML`, `turtle`, etc. or loaded to a triple store. It is interesting to notice that the nested structure of object is returned for both of these methods, i.e. if the `triplify()` function of an `Episode` is called, all of its components (semantic descriptions and extents) are returned by triggering its respective `triplify()` methods. That way, only one call to a `FeatureOfInterest` object is enough to retrieve a graph of all the episodes that compose that object. That way, applications have multiple options of consuming the output of automatic annotation algorithms.

The `STEP` framework can be compared to `SeMiTri` (Semantic Middleware for Trajectories) [Yan et al. 2011] as both have been devised to serve as a single system that concentrates different algorithms that may work together eventually. One important difference is that we provide an object-graph interface that maps the `STEP` ontology to classes that can be used by applications.

³⁷<http://www.pandas.pydata.org>

³⁸<http://www.numpy.org/>

³⁹<https://github.com/tkrajina/gpxpy>

⁴⁰<https://github.com/tkrajina/srtm.py>

⁴¹<http://toblerity.org/shapely>

⁴²<https://github.com/DinoTools/python-overpy>

⁴³<https://github.com/RDFLib/rdflib>

3.3 Automatic Annotation Algorithms

3.3.1 MSN: Move-Stop-Noise Classification

In this section, we describe a way of creating episodes based on the detection of stops and moves during a single trajectory. Detecting periods of movement and its absence is a fundamental segmentation criteria and it has been vastly explored [Alvares et al. 2007, Palma et al. 2008, Yan et al. 2010, Rocha et al. 2010, de Graaff and Keulen 2016]. Applications dealing with real world data also have to deal with noisy measurements which, in some cases, makes it impossible to determine the actual state of the moving object. Although some related work have considered the presence of noise in trajectories, they usually handle this by previous smoothing or by using additional data that is not always available.

The characteristics of trajectory data can vary broadly according to sensor's physical components, sampling rate, algorithmic post-processing, environmental conditions, among other factors. The conjunction of these elements may result in trajectories with different levels of quality even for traces captured by the same device. Therefore, this facet of spatio-temporal data research disfavor the possibility of proposing a universal method for detecting stops and moves as well as trajectory segmentations based on other criteria.

As can be observed from related work, it is thus necessary to make assumptions about the collected data before proposing an approach to trajectory segmentation. The main assumption of our method is that points are not sampled with the same frequency during the entirety of the trajectory. Therefore, we consider the existence of a spatial filter that discards redundant points or stops recording points when the object is not moving. Also, we consider that the sampling rate is approximately constant when the object is moving, i.e. new points are recorded at near equally spaced interval of times. This has been noted in our previous work [Nogueira et al. 2014], but we did

not consider noisy trajectory segments nor used robust statistic measures.

An example of the spatial filter can be seen in Listing 3.5 where a location listener is created to get updates from the GPS sensor each 10 meters, or 35 seconds.

```
1 LocationListener locationListener = new MyLocationListener();
2 LocationManager lm =
3     (LocationManager) getSystemService(Context.LOCATION_SERVICE);
4
5 lm.requestLocationUpdates(LocationManager.GPS_PROVIDER,
6     35000, 10, this.locationListener);
```

Listing 3.5: Java code for getting position updates only if the moving object has moved for at least 10 meters.

Another important difference in our method is that the notion of stop in other works is usually related to the identification of Regions of Interest allowing the classification of a point as a stop even when there is movement, while in our work we aim to identify moments where an actual stop took place. Moreover, in this stage we do not consider using external contextual data (e.g. the polygons of adjacent geographic features) or additional sensor data (e.g. GPS accuracy) as we do not link stops to Points or Regions of Interest and, in the second case, we favor to use trajectory attributes that can be computed only from the raw spatiotemporal data, i.e. latitude, longitude, and timestamps.

3.3.1.1 Exploratory Data Analysis

An important initial task when analyzing a dataset is verifying the relationships among the variables. The output of this analysis allows to highlight the relationships of variables that tends to better explain the dataset variability. A useful statistic method for such kind of analysis is the verification of correlation strength among variables.

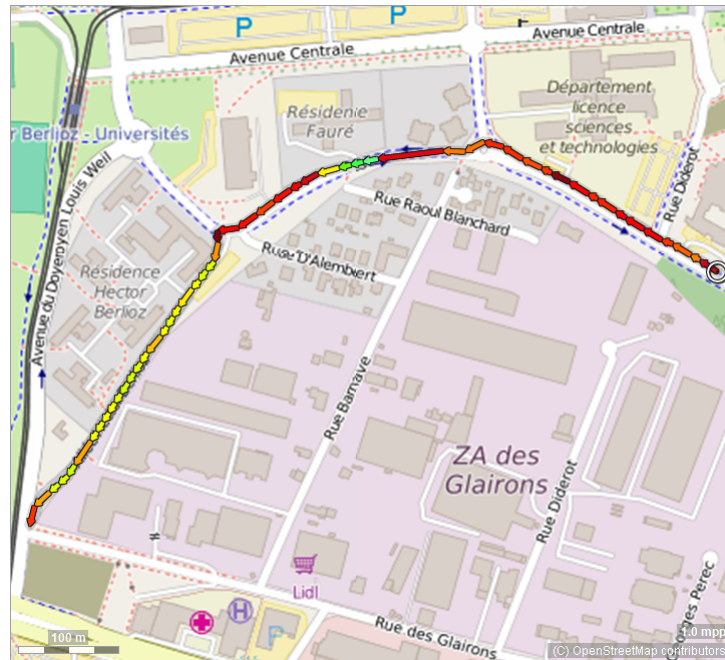


Figure 3.12: An example trajectory.

A popular method to calculate the correlation coefficient between two variables is the Pearson product-moment correlation coefficient. This method consists in inferring a linear regression line of best fit through the 2D projection of the two variables. The resulting Pearson correlation coefficient (r) is an indication of how close the data points are to the projected line.

We are going to use the trajectory shown in Figure 3.12 as illustrative example of the [MSN](#) method for detection of moves, speed and noise segments. This trajectory was recorded in a controlled manner in order to have two stops of a few seconds and two periods of noise that have been simulated by turning off the smartphone [GPS](#) for a few seconds.

For each pair of sequential points in the trajectory of Figure 3.12, we can calculate the speed, distance and duration between points as shown in Figure 3.13. In this Figure, we can observe some interesting characteristics based on previous knowledge about this particular trajectory. The trajectory starts with distances of about 10 meters between points, durations of 5 to 7

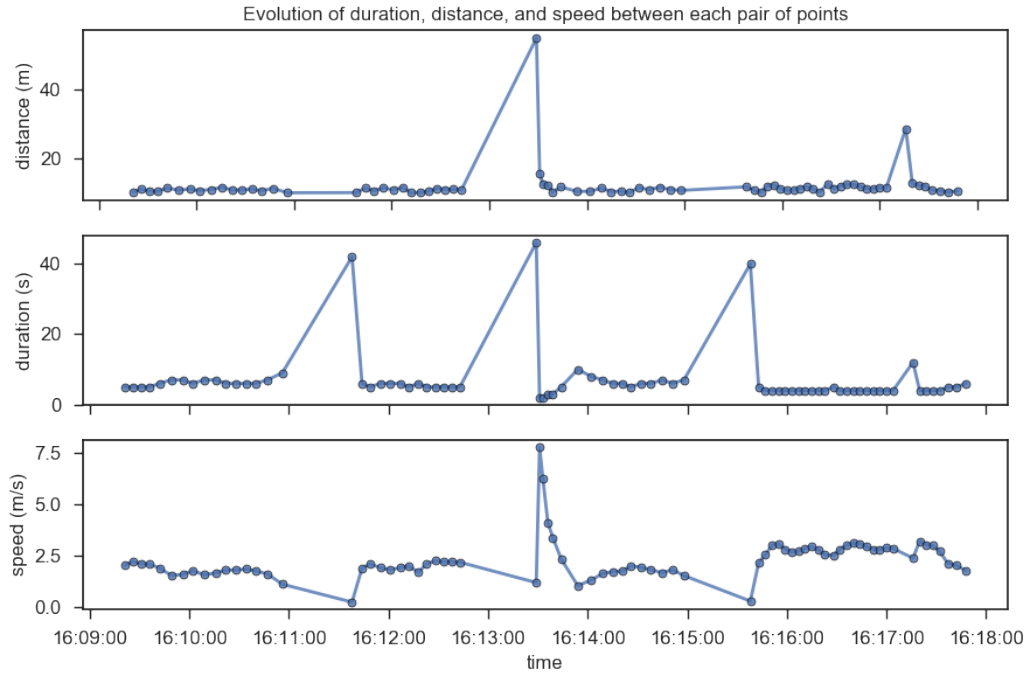


Figure 3.13: Speed, distance, and duration between points of the example trajectory.

seconds, and a fairly constant speed until there is a peak of 42 seconds in the duration series. At the same moment, we can observe that the speed drops to a value near zero while the distance remains unchanged. This characterizes a stop. Some seconds later in the example trajectory, it is possible to notice another peak in duration at the same time of a peak in the distance between points that are not followed by a decrease in speed. This characterizes a period of noise. In the remaining of the trajectory, another stop and another noise period can be noticed.

The conclusions achieved in the example have been confirmed by an exploratory study of the Spearman correlation of some trajectory characteristics (see Equation 3.3). We have chosen this measure because it is more resistant than the Pearson coefficient as it diminishes the importance of extreme scores by first ranking the two variables and then correlating the ranks instead of the

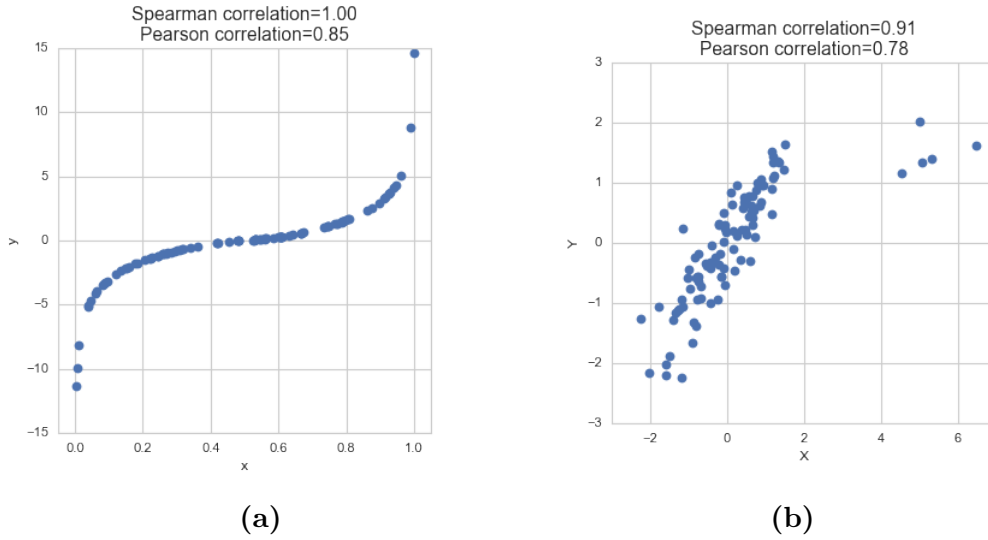


Figure 3.14: Two cases demonstrating the better performance of Spearman correlation over Pearson correlation in (a) non-linearity of relationship and (b) presence of outliers.

actual values [Myers and Well 2003]. The advantage of Spearman correlation over Pearson’s is shown in Figure 3.14.

$$r_s = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \quad (3.3)$$

Tables 3.2, 3.3, and 3.4 show aggregated statistics of the Spearman correlation among some movement attributes of 2226 trajectories, which were collected from a widely used third-party mobile application. Walking and running activities ranging from 2 to 42 kilometers in the cities of Grenoble (France) and Barcelona (Spain) were randomly selected. From this data, we can observe that the pairing between speed and duration is the one that presents the strongest correlation. In this case, a strong negative correlation indicates that when the values of duration increase, the values of speed tend to decrease and vice-versa, which is what one can expect given the previously explained assumptions about the data.

From the experiments performed with the described trajectories, we can

	duration	distance	speed	acceleration
duration	1	0.167612	-0.862789	0.345232
distance	0.167612	1	0.289103	-0.0314847
speed	-0.862789	0.289103	1	-0.361087
acceleration	0.345232	-0.0314847	-0.361087	1

Table 3.2: Median of Spearman correlations among movement attributes of 2226 trajectories

	duration	distance	speed	acceleration
duration	1	0.191846	-0.827017	0.342137
distance	0.191846	1	0.268344	-0.0303889
speed	-0.827017	0.268344	1	-0.35647
acceleration	0.342137	-0.0303889	-0.35647	1

Table 3.3: Mean of Spearman correlations among movement attributes of 2226 trajectories

	duration	distance	speed	acceleration
duration	0	0.255567	0.127464	0.11003
distance	0.255567	0	0.191583	0.0647744
speed	0.127464	0.191583	0	0.103414
acceleration	0.11003	0.0647744	0.103414	0

Table 3.4: Standard deviation of Spearman correlations among movement attributes of 2226 trajectories

conclude that there is a negative correlation between the values of speed and duration that characterizes a stop. For the noisy cases, there is no pair of variables that helps the classification. Thus, we make use of the assumption that the points are recorded at near constant distance intervals most of the time. Additionally, we propose a subprocedure to identify consecutive points showing unrealistic movement behavior based on the angle between points.

3.3.1.2 Outlier Labeling Rule

Based on the exploratory data study, we can approach the classification of moves, stops and noise as an outlier detection problem. In order to identify the outliers in time series, we use the modified z-score proposed by [Iglewicz and Hoaglin \[1993\]](#). The usage of this method is motivated by the poor performance of other popular measures like the standard deviation and the mean in the presence of outliers.

An indicator of the robustness of a statistic is its breakdown point, i.e. the maximum proportion of outlier data points that can be inserted into the dataset before the statistic gives a wrong result. The *mean* is known to have a breakdown point of 0% because, for instance, if just one value of a given series is set to infinity, the series' mean goes to infinity. On the other hand, the *median* (the central point of a series) is a statistic measure with a high breakdown point. Using the same concept of the previous example, the median value of a series is only affected if more than 50% of the data would be set to infinity.

Another estimator that is easily modified in the presence of outliers is the standard deviation, as it takes into consideration the squared distance from the mean for each value. According to [Huber and Ronchetti \[2009\]](#), the most useful ancillary estimate of scale is the Median Absolute Deviation (**MAD**) (see Equation 3.4), which is the median of absolute distances from the series median. The constant scale factor 1.4826 makes the **MAD** unbiased at the

normal distribution [Rousseeuw and Hubert 2011].

$$MAD = 1.4826 \times \text{median}(|Y_i - \tilde{Y}|) \quad (3.4)$$

Besides its superiority, the MAD is not yet largely used in some fields, as noticed by Leys et al. [2013]. In the same work, the authors recommend using “the median plus or minus 2.5 times the MAD method for outlier detection”. In our work, we use the modified z-score, which also uses the MAD and it is based on the z-score (see Equation 3.5), which uses the non-robust statistics mean (\bar{Y}) and standard deviation (s).

$$Z_i = \frac{Y_i - \bar{Y}}{s} \quad (3.5)$$

Iglewicz and Hoaglin [1993] recommend using the modified z-score shown in Equation 3.6 where each element of a series is subtracted from the median (\tilde{x}), multiplied by a factor to make the MAD consistent at the normal distribution (0.6745). As a final recommendation from the authors, points having modified z-scores with an absolute value greater than 3.5 have a high probability of being outliers [NIST/SEMATECH 2012]. Besides the advantage of using the MAD statistic in what concerns it being more robust to outliers than the standard deviation, it is also adequate for application in populations that do not fit perfectly a Gaussian distribution [Gorard 2005], which is the case for real world GPS track datasets.

$$M_i = \frac{0.6745(x_i - \tilde{x})}{MAD} \quad (3.6)$$

As an example of the advantage of using MAD, consider the following set without any outlier $x = \{6.27, 6.34, 6.25, 6.31, 6.28\}$ with mean $\bar{x} = 6.29$, median $\tilde{x} = 6.28$, and standard deviation $\sigma_x = 0.03$. Now, considering the introduction of an outlier, the set becomes $x' = \{6.27, 6.34, 6.25, 63.1, 6.28\}$ with mean $\bar{x}' = 17.64$, median $\tilde{x}' = 6.28$, and standard deviation $\sigma_{x'} = 22.72$.

We can compute the ordinary z-scores for the two sets as defined in Equation 3.5 and get $Z_x = \{-0.6, 1.58, -1.2, 0.63, -0.31\}$ and $Z_{x'} = \{-0.5, -0.5, -0.5, 1.99, -0.5\}$. Using the popular threshold of 2.5 to label a value as an outlier would not identify the outlier introduced in x' . Both sets have the same $MAD = 0.04$ and if we compute the absolute distance (AD) from the median, we get the following results: $AD_x = \{0.01, 0.06, 0.03, 0.03, 0\}$ and $AD_{x'} = \{0.01, 0.06, 0.03, 56.8, 0\}$. Finally, applying Equation 3.6 to both sets yields: $M_x = \{-0.22, 1.34, -0.67, 0.67, 0\}$ and $M_{x'} = \{-0.22, 1.34, -0.67, 1277, 0\}$. We can notice that the outlier in the second set highly exceeds the recommended threshold of 3.5.

3.3.1.3 The MSN algorithm

Considering a trajectory $\tau = \{(s_1, t_1), (s_2, t_2), \dots, (s_n, t_n)\}$, where each position $s_i = (lat, lon)$ is a pair of latitude and longitude coordinates, and each time instant t_i is represented by a timestamp, for each pair of points $(s_i, t_i), (s_{i+1}, t_{i+1})$, we compute its spatial distance, duration, speed, and turning angle. Then, we store these values in their respective time series $S_\tau, T_\tau, V_\tau, A_\tau$, having as index the same timestamps of the raw trajectory τ . For A_τ , a turning angle consists on the angle formed by three neighboring points as has been described in Equation 3.2 and Figure 3.11.

Considering the above computed time series, we are able to formulate an algorithm (see Algorithm 1) for determining which instants of the trajectory are likely to be stops or moves, as well as which segments have a degree of uncertainty that makes this classification impossible. The algorithm's input are the time series representing the spatial distances (S_τ), the durations (T_τ), the speed (V_τ), and the turning angle (A_τ) between points besides the thresholds ϵ_s, ϵ_t , and ϵ_v representing the modified z-score limits for distance, duration, and speed, respectively. Additionally, we have included a minimum turning angle parameter (θ) to improve the noise detection following the intuition that

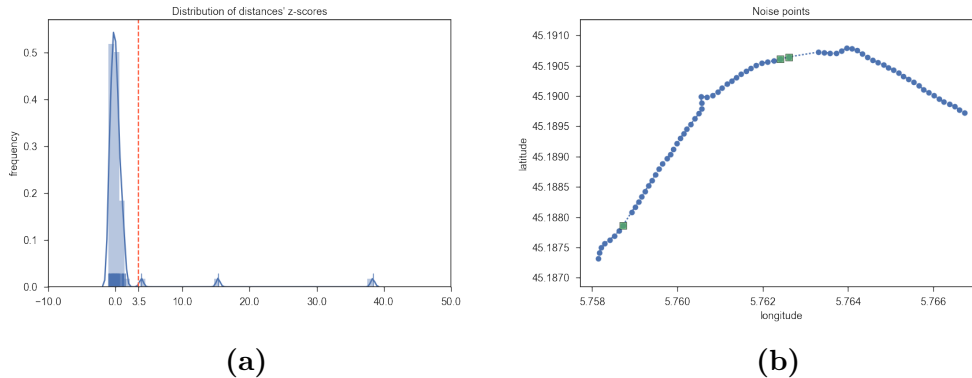


Figure 3.15: The density plot of distances in (a) and the three trajectory points with long distances in (b)

it is improbable for a moving object to take successive turns with small angles, and a random uniform jitter (ρ) to avoid the MAD breakdown point.

As the trajectory’s sampling rate is assumed to be performed at nearly constant distance only while the object is moving and locations are not retained while the object is stopped, the problem can be summarized into searching for outliers into the time series as they are expected to have relevant gaps characterizing periods of either stops or noise. These gaps in time coincide with decreases in speed for the cases of stops as can be observed by the negative correlation between these two variables.

In order to better explain the MSN algorithm, we consider the example trajectory of Figure 3.12. We have set the following values for the parameters in this example: $\epsilon_s = \epsilon_v = 3.5$, $\epsilon_t = 5$, $\theta = 45$.

We can see in Algorithm 1 that the first part in MSN is to identify potentially erroneous points, for which we use the general term “noise”. This first classification, shown in lines 2-8, identifies the points having relative long distances. In the example (Figure 3.15), three points are identified in this phase. They have distances of about 17, 28, and 55 meters, while the median distance of all pairs of sequential trajectory points is 11 meters.

Algorithm 1 Move-Stop-Noise classification algorithm

```

1: procedure MOVESTOPNOISE( $S_\tau, T_\tau, V_\tau, A_\tau, \epsilon_s, \epsilon_t, \epsilon_v, \theta, \rho$ )
2:    $distance\_outliers = []$ 
3:    $M_s = \text{MODIFIEDZSCORE}(S_\tau, MAD_s, \tilde{s})$  ▷ Equation 3.6
4:   for  $i = 0$  to  $length(M_s)$  do:
5:     if  $M_s[i] > \epsilon_s$  then ▷ Get long distances
6:       Append  $i$  to  $distance\_outliers$ 
7:     end if
8:   end for
9:    $direction\_outliers = []$ 
10:  for  $i = 0$  to  $length(A_\tau)$  do:
11:    if  $A_\tau[i] < \theta$  and  $A_\tau[i + 1] < \theta$  then ▷ Get sharp turning angles
12:      Append  $i$  and  $i + 1$  to  $direction\_outliers$ 
13:       $i++$ 
14:    end if
15:  end for
16:   $noise\_indexes = distance\_outliers + direction\_outliers$ 
17:   $clean\_indexes = \tau - \tau[noise\_indexes]$ 
18:   $\tau = \tau[clean\_indexes]$  ▷ Removing noisy points
19:   $T_\tau = T_\tau + \rho$  ▷ Adding small random uniform noise
20:   $duration\_outliers = []$ 
21:   $M_t = \text{MODIFIEDZSCORE}(T_\tau, MAD_t, \tilde{t})$ 
22:  for  $i = 0$  to  $length(M_t)$  do:
23:    if  $M_t[i] > \epsilon_t$  then ▷ Get long durations
24:      Append  $i$  to  $duration\_outliers$ 
25:    end if
26:  end for
27:   $V_\tau = \ln V_\tau$  ▷ Natural log of speed
28:   $speed\_outliers = []$ 
29:   $M_v = \text{MODIFIEDZSCORE}(V_\tau, MAD_v, \tilde{v})$ 
30:  for  $i = 0$  to  $length(M_v)$  do:
31:    if  $M_v[i] < -\epsilon_v$  then ▷ Get slow speeds
32:      Append  $i$  to  $speed\_outliers$ 
33:    end if
34:  end for
35:   $stop\_indexes = duration\_outliers \cap speed\_outliers$ 
36:   $move\_indexes = clean\_indexes - stop\_indexes$ 
37:  return  $noise\_indexes, stop\_indexes, move\_indexes$ 
38: end procedure

```

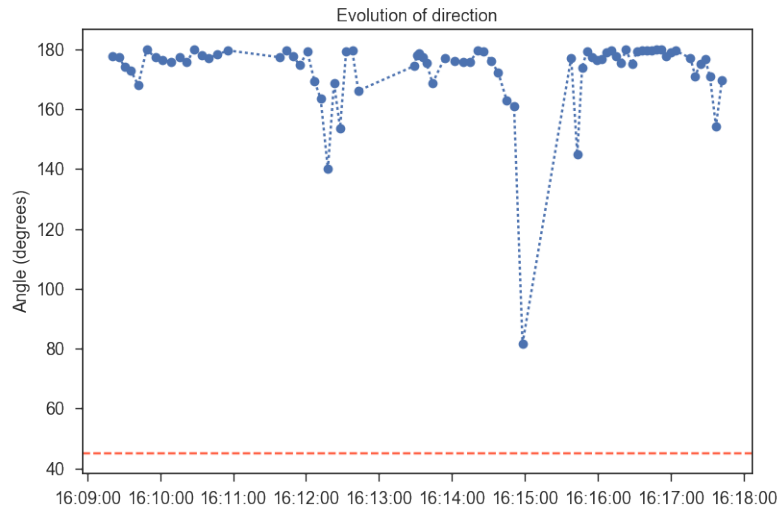


Figure 3.16: Time series of turning angles. There are no group of consecutive angles with less than 45 degrees, therefore no direction outliers are identified.

The second step of noise classification consists in verifying the turning angles (lines 9-15). We account for the fact that only one sharp angle in a trajectory may represent a movement of “turning back”, while two consecutive sharp angles is less likely to happen and can be considered as a potential noisy segment. For that reason, we test for neighbor points and move the iteration one step further if at least a pair like this is found. This case is not present in the example trajectory as can be seen in Figure 3.16. There is no group of points as vertices of angles of less than 45 degrees.

Once the noisy points are identified (the union of distance and direction outliers), the second part of the MSN algorithm consists in labeling potential stops by removing the noisy points and analyzing modified z-scores. Lines 19-26 show the first step, which is designed to identify long time gaps.

We have observed that the time series of duration between points may contain repeated values in more than 50% of the data. In these cases, the MAD value is always equal to zero (Equation 3.4), which causes a division

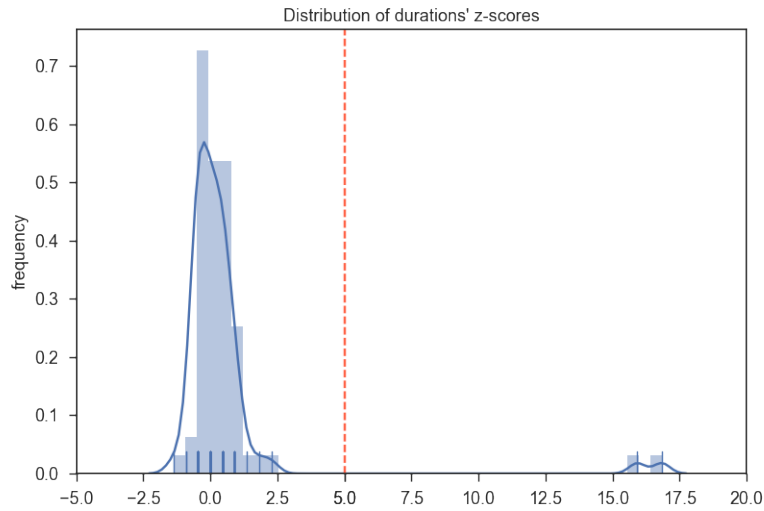


Figure 3.17: Distribution plot of slightly “jittered” durations where two outliers are identified.

by zero in the modified z-score computation (Equation 3.6). To avoid this, we add a small amount of random uniform noise to the duration series (line 19). The value to be added is randomly selected from a interval, e.g. $[-0.5, 0.5)$, corresponding to the ρ variable in Algorithm 1. For instance, the time series $\{4, 5, 5, 6, 6\}$ may become $\{4.02, 5.19, 4.97, 6.37, 5.97\}$, which is enough to avoid the MAD being set to zero and does not have an impact on the determination of stops as a half-second change can be considered as negligible in our context.

After adding the small jitter to durations, we apply the same procedure as the one applied to find the distance outliers. However, we have set the modified z-score threshold to 5 in order to avoid false positives. Figure 3.17 shows the distribution of durations for the example trajectory. Two long durations are identified having 40 and 42 seconds while the median duration of the entire trajectory is around 5 seconds.

The complement of stop identification (lines 27-34) regards the analysis of the trajectory speed. The Outlier Labeling Rule of Iglewicz and Hoaglin [1993]

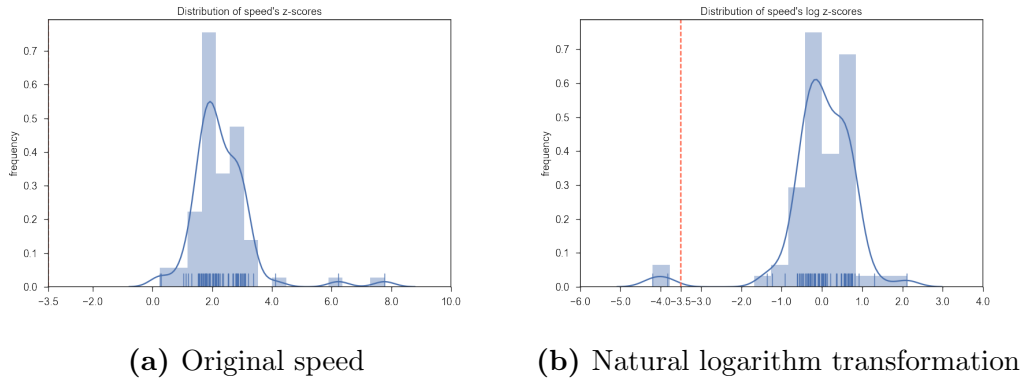


Figure 3.18: Difference of speed data before and after natural logarithmic transformation. Also, the outlier threshold is shown in (b)

should be applied to approximately normally distributed datasets, however, for the trajectories considered in this work, speed data has demonstrated to be positively skewed in general. In order to normalize speed data, the natural logarithm was applied as taking the log of the data can restore symmetry in this case [Zumel and Mount 2014]. Figure 3.18 shows the importance of this transformation to find slow speed outliers. In Figure 3.18b, it is possible to see that two points have speeds relatively slower ($0.23m/s$ and $0.29m/s$) while the median speed value for the example trajectory is $2.1m/s$.

Successive points having speeds lower than the specified threshold ϵ_v are special cases because they can be classified either as stops or moves depending on the interpretation of who analyses the data. In the first case, considering these points as stops brings the risk of classifying a long period of slow movement as a stop, which would make the spatial extent of the stop greater than normal. On the other hand, due to the accuracy of GPS acquired positions, the sequence may actually represent a period where the object was stationary. A possible approach for these cases could involve analyzing more variables. For instance, one could define a threshold to the number of contiguous points with low speed where sequences having more points than the threshold are

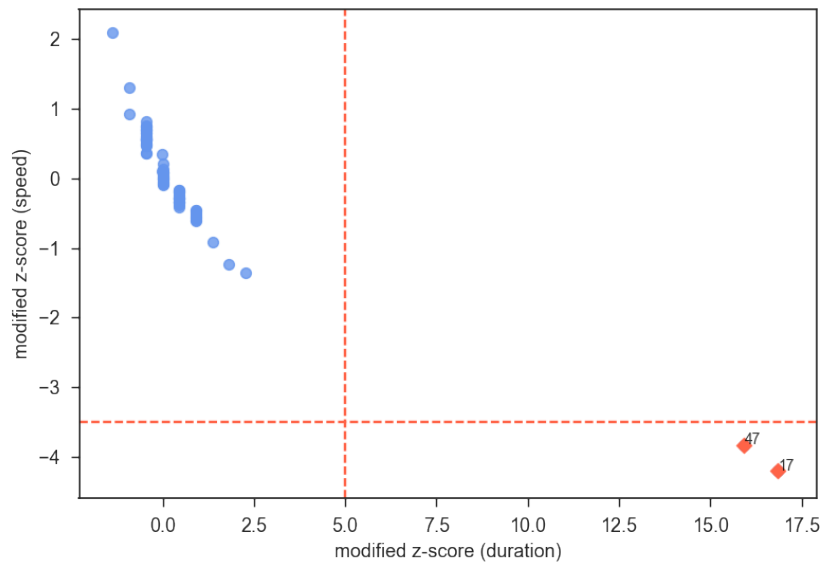


Figure 3.19: Modified z-scores of speed and duration with their thresholds.

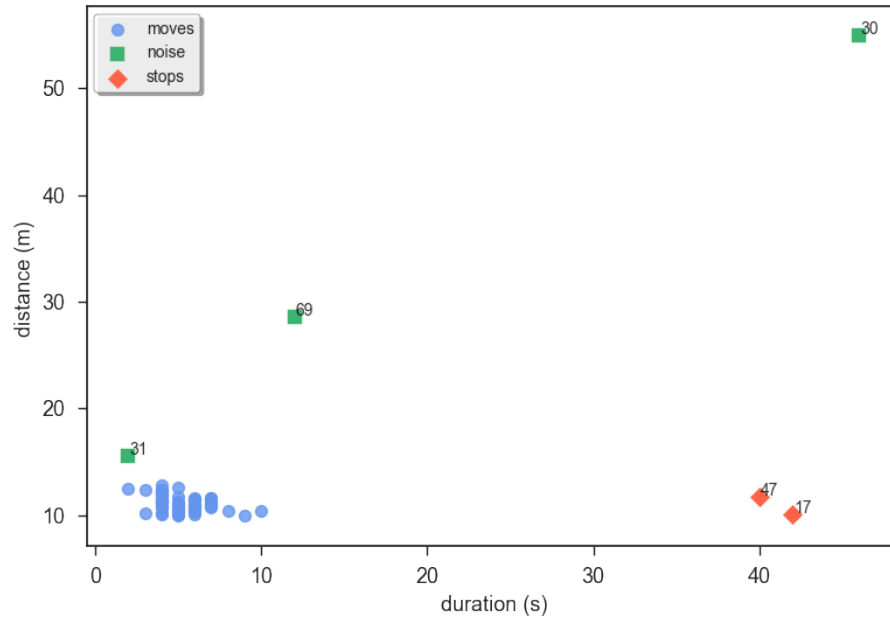
considered as moves. We left this task as a future work.

Finally, we define as stops the points that have present both slow speed and long durations. Figure 3.19 shows these two variables with their respective thresholds. According to our algorithm, the points located in the lower right quadrant have a high probability of being stops. Figure 3.20 shows all points with their classification as either move, stop, or noise.

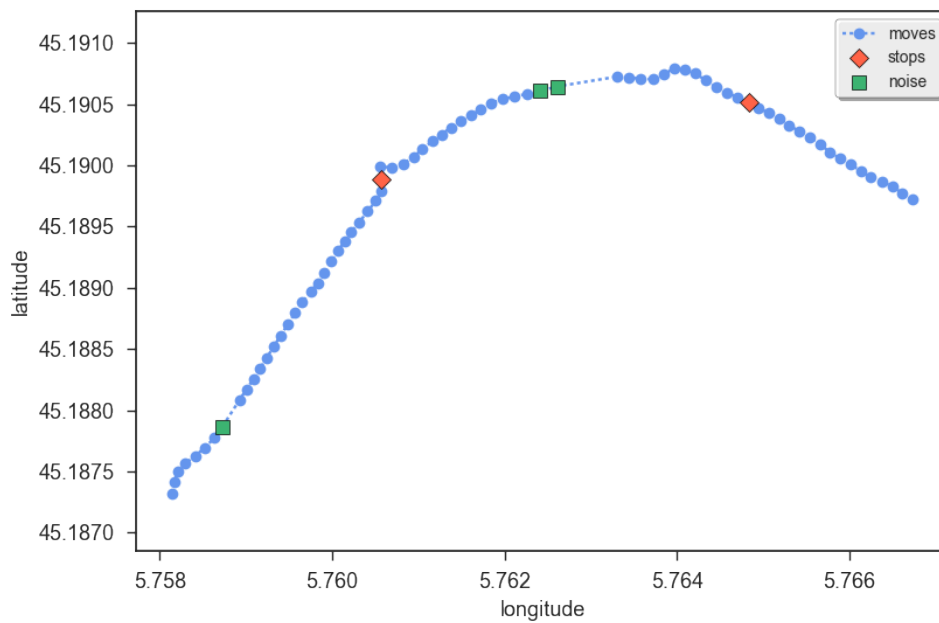
3.3.1.4 Evaluation

Due to the assumptions about the trajectory sampling strategies (constant sampling versus spatial filtering), it is difficult to make a comparison with related work by running all algorithms with the same set of trajectories. In this evaluation, we focus on analyzing the theoretical performance of other stop classification methods and we highlight the main differences from our work.

Table 3.5 shows a general comparison of the main algorithms for stop



(a) Outliers identified as noise and stops. Normal points are considered as periods of movement.



(b) The outliers marked in the trajectory geometry.

Figure 3.20: Final classification of the MSN algorithm.

Table 3.5: Comparison of stop detection algorithms

	Parameters	Noise Handling	Spatial Filter Support	External Data Independent
SMoT [Alvares et al. 2007]	Polygons, minimum duration for each polygon	No	No	No
CB-SMoT [Palma et al. 2008]	Polygons, area, minimum duration	No	No	No
DB-SMoT [Rocha et al. 2010]	Minimum direction change (degrees), minimum duration (hours), maximum tolerance (number of points)	No	No	Yes
Velocity-based trajectory structure [Yan et al. 2010]	Minimum stop duration, Object speed threshold coefficient, Cell speed threshold coefficient	No	Yes	No
CandidateStops [Nogueira et al. 2014]	Minimum speed (m/s)	No	Yes	Yes
PIE [de Graaff and Keulen 2016]	Polygons, maximum inaccuracy (meters), minimum staypoint distance (meters), minimum staypoint time (seconds), minimum direction change (degrees), maximum projection distance (meters)	Yes	No	No
MSN	Distance outlier threshold, duration outlier threshold, speed outlier threshold, minimum direction change (degrees)	Yes	Yes	Yes

detection in the literature. As advantages of our method, we can point out the independence of external data, the usage of characteristics that can be completely extracted from the trajectory points, the robustness of statistic methods involved, and the handling of noise.

By not relying on the polygons of the underlying geography, our method is adequate to trajectories that are not in a constrained space, being able to identify stops also in free space. Also, apart from the minimum turning angle in degrees, an important aspect of MSN is that the other threshold parameters are not based on metric quantities, e.g. distance in meters or duration in seconds, instead, the outlier labeling rule allows the usage of more

abstract thresholds.

A drawback of [MSN](#) is the fact that it relies on the comparison of data points relatively to the rest of the dataset. For instance, to identify a large time gap correctly, it is necessary to the majority of other time gaps to have a short time, which is not surprising because we base our method on outlier detection for approximately normally distributed data. However, if the trajectory contains a large quantity of noise, the method may fail in recognizing stops. This can be avoided by a preprocessing step to assess the level of noise before applying the [MSN](#) algorithm and tries to alleviate the noise or diagnose that the level of erroneous data makes further analysis inviable.

Regarding algorithm complexity, the [MSN](#) method can be implemented in $O(n + m)$ considering a raw trajectory with n points before noise removal and m points after noise removal. In the worst case, $n = m$ (no points are discarded in the noise classification phase), thus, the complexity is $O(2n)$. It is important to notice that we have not shown in [Algorithm 1](#) the most concise and efficient implementation for the sake of clarity, but the noise handling could be summarized in one *for* loop as well as the stop classification. A Python implementation of the [MSN](#) algorithm is also available in [Appendix A](#).

3.3.2 ANBA: Adaptive Natural Breaks Approximation

Once trajectory preprocessing steps are done (e.g. cleaning, map matching, stops extraction), the features of interest like speed, direction, acceleration (i.e. features that consist into numerical or categorical data indexed by time) can be represented into our framework. However, these features can hold a large amount of data that would require a considerable storage space. Instead, this data can be compressed without losing its most important structures. This compression is beneficial both for reducing storage requirements as well as for

facilitating further analysis. A myriad of approaches can be used to transform time series into smaller compressed abstractions of observations [Aghabozorgi et al. 2015]. In this work, we propose a segmentation strategy based on variance changes of a unidimensional set of values where the objective is to identify intervals with near homogeneous values. With the method presented in this section, we focus rather on the classification of a time series in few classes instead of finding linear segments that better fit the time series as it has been done in our previous work [Nogueira et al. 2014, Nogueira and Martin 2014] where we have used PLS and represented the time series segments in the QualiTraj ontology.

3.3.2.1 Fisher-Jenks Natural Breaks

In order to achieve the classification of a time series into a set of classes, we propose using the Fisher-Jenks Natural Breaks algorithm with an optional step to automatically determining the number of classes and two additional steps that helps in avoiding overfitting the approximated series and therefore avoids creating more segments. The Fisher-Jenks algorithm was proposed by Jenks [1977] and is an evolution of the Jenks-Caspall algorithm [Jenks and Caspall 1971] as the first one did not guarantee an optimal solution. Although Fisher [1958] was the responsible of the method’s mathematical soundness, this algorithm is usually refereed as “Jenks Natural Breaks” [Slocum et al. 2009]. The Fisher-Jenks algorithm benefits from the fact that any optimal partition is equal to the sum of optimal partitions of subsets of the data, as demonstrated by Fisher [1958], to reduce the search space and not have to search through all possible combinations.

Intended for choosing different colors for choropleth maps [Andrienko et al. 2001], the objective of the Fisher-Jenks algorithm is to find break points in a univariate dataset with the goal of minimizing intraclass variance while maximizing inter-class variance. This is done by introducing a measure of error,

e.g. the Absolute Distance from the Class Median (**ADCM**), and calculating the sum of **ADCM** for subsets of the data. For instance, given the set $\{2, 3, 4, 10, 12, 15, 20, 21\}$ and a number of classes equals to 3, the Fisher-Jenks algorithm would return the break points 2, 10, 20, 21, which are enough to infer the subset intervals: $\{\{2, 3, 4\}, \{10, 12, 15\}, \{20, 21\}\}$, being this configuration the one that has the smallest intraclass variation and greatest inter-class variability.

3.3.2.2 The ANBA algorithm

The name **ANBA** is inspired by the Natural Breaks found by the Fisher-Jenks method. A disadvantage of the Fisher-Jenks algorithm is the need of previously informing the number of classes, which hinders its generalization to the automatic segmentation of potentially diverse datasets. To overcome this drawback of the Fisher-Jenks algorithm, we have included in our method an optional initial step for finding a reasonable number of classes for an individual time series based on a more abstract parameter. Once the optimal breaks are found, the output of Fisher-Jenks algorithm is used to segment the time series into classes. After that, we use each class median as a Candidate Approximate Value (**CAV**) for the corresponding class and then possibly merge **CAVs** to define the final Approximate Values (**AV**). At last, each original input value is approximated to its nearest **AV**. The two additional steps of merging **CAVs** and approximate the original points to **AVs** are influenced by two parameters of **ANBA**. As a result, we are able to discretize any series into a reduced set of segments without losing its general shape. A high level representation of the **ANBA** algorithm is presented in Algorithm 2. A Python implementation of the algorithm is also available in Appendix B.

The first optional part of **ANBA** consists in setting the number of classes to 1 (the minimum possible value), running the Fisher-Jenks algorithm iteratively and assessing the output's Goodness of Variance Fit (**GVF**), increasing

the number of classes by 1 in the next iteration if needed (case the minimum **GVF** informed by the user has not been reached with the current number of classes). Thus, a target **GVF** can substitute the original parameter of the Fisher-Jenks algorithm (number of classes) and this procedure is not executed if a target **GVF** is not informed. This way, instead of receiving the number of classes, the algorithm receives a real number in the interval $[0, 1]$ where 0 represents the least resemblance with the original signal and 1 means that the final output should be identical to the original time series. The **GVF** value is defined in Equation 3.7 where **SDCM** stands for Squared Deviations from the Class Mean and **SDAM** stands for Squared Deviations from the Array Mean.

$$GVF = \frac{SDCM}{SDAM} \quad (3.7)$$

In the evaluation shown in the next subsection, we differentiate the two ways of using **ANBA** by **ANBA-n** when the number of classes are informed beforehand, hence discarding the need of the **GVF** verification procedure, and **ANBA-gvf** when the number of classes are not inputted and the multiple calls to Fisher-Jenks may be executed.

In Algorithm 2, we consider *ts* a given time series, *min_gvf* the minimum goodness of variance fit, *n_classes* the number of classes, *min_dist* the minimum distance between two **AVs**, and *min_time* the minimum duration of a segment. The only obligatory parameters are *ts* and one of either *min_gvf* or *n_classes*. If *min_gvf* is inputted by the user, the algorithm executes the lines from 3 to 10 in order to find the number of classes that satisfies the target **GVF** as previously explained.

Having defined the number of classes, the Fisher-Jenks algorithm is executed (line 10), which returns the natural breaks of values. The minimum and maximum values are also returned along with the breaks, allowing the creation of ranges that delimit the classes.

Algorithm 2 ANBA: Adaptive Natural Breaks Approximation

```

1: procedure ANBA(ts, min_gvf, n_classes, min_dist, min_time)
2:   if min_gvf is defined then
3:     gvf = 0
4:     n_classes = 1
5:     while gvf < min_gvf do
6:       n_classes = n_classes + 1
7:       gvf = GOODNESS_OF_VARIANCE_FIT(ts, n_classes)
8:     end while
9:   end if
10:  breaks = FISHER-JENKS(ts, n_classes)
11:  CAV = Calculate median points of each class
12:  AV = MERGE_CANDIDATE_APPROXIMATIONS(CAV, min_dist)
13:  approximated_points = Approximate each ts point to its nearest AV
14:  for point in approximated_points do
15:    if point duration <= min_time then
16:      Change point class to the class of a neighbor approximated point
17:    end if
18:  end for
19:  return segments
20: end procedure

```

The next step (line 11) consists in calculating the **CAV** for each class and then verifying if the **CAVs** have distances greater than min_dist from each other. The output of the subprocedure **MERGE CANDIDATE APPROXIMATIONS** may be the same as the medians if all median values are distant from each other by a distance equal or greater than min_dist . Otherwise, there may exist two cases for median values that are too close: a pair, or three or more median values. For the first case, the mean between the two values are taken into consideration as approximate value. In the second case, a range starting from the minimum **CAV** to the maximum **CAV** of the subset is generated with the min_dist serving as step. In this case, the last **CAV** is not always present in the selected values.

For instance, consider that the list of **CAVs** is $\{2, 2.4, 3.5\}$ and $min_dist = 0.5$. Then, as it is a case of a pair of close values, the final **AV** is set to $\{2.2, 3.5\}$. Now, suppose that $CAV = \{2, 2.4, 2.7, 3.5\}$ and $min_dist = 0.5$. The first three elements from **CAV** violates the minimum specified distance, therefore they are substituted by $\{2, 2.5\}$. The final **AV** is $\{2, 2.5, 3.5\}$.

The last step of **ANBA** is approximating the points to **AVs** (lines 13 to 18). For each point of the time series, its distance from all **AVs** is computed and the shortest one is chosen. In this part, the minimum segment time constraint is taken into consideration, i.e. if a segment would have a duration shorter than min_time , and it is part of a different class compared to its right and left neighbors, we approximate this segment to the preceding point's class instead. This avoids isolated segments with negligible durations. For instance, if in a given point of time series the values drops enough to be classified by another class and then rises again, it will not be relevant enough to create a new segment.

Introducing min_dist and min_time has the potential to influence the **GVF** of the final result. As these two parameters may cause the displacement of

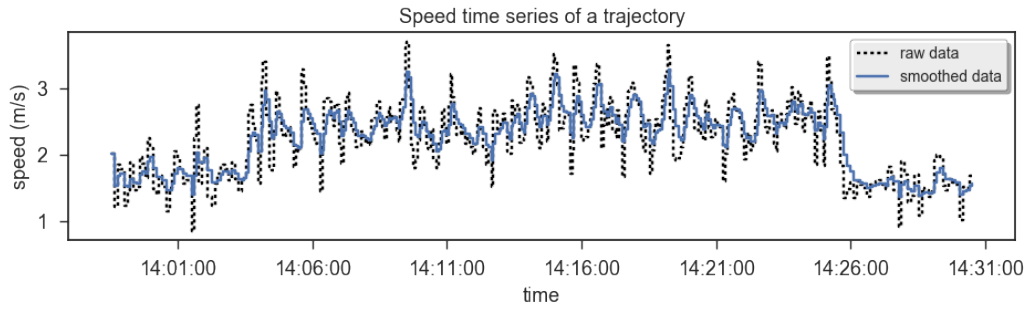
points from the median of its corresponding class, causing a decrease of the correlation between the original data and the output.

Considering that a time series have n points to be divided into k classes, there are known algorithms to compute the natural breaks in $O(k \times n \times \log(n))$ [Hilferink 2013]. In the cases of ANBA-gvf, this procedure can significantly increase running time due to the multiple calls of Fisher-Jenks. The steps of merging the CAVs is proportional to the number of classes k , $O(k)$, and approximating AVs is done in $O(n^2)$ as first the values are approximated and then reevaluated to find short duration segments. Specifically for this last step, there is still room for improvement that can be subject of future work.

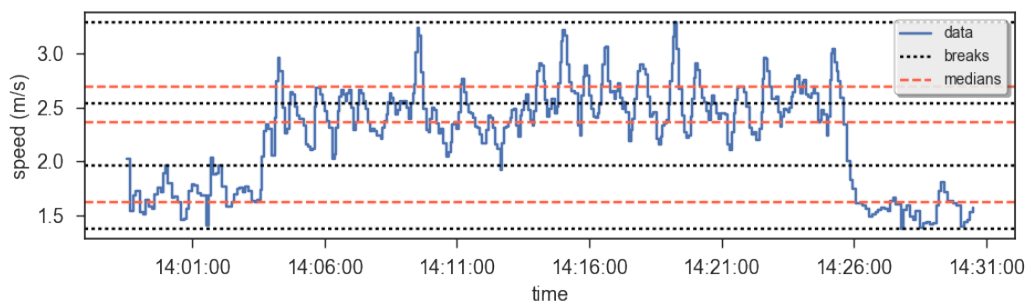
Figure 3.21 shows the main steps of the ANBA process in an example time series computed from the speed evolution of a real trajectory. The following parameters have been used for this example: $min_gvf = 0.5$, $min_dist = 0.5$, $min_time = 7$ seconds. While the first parameter is the most abstract, the other ones heavily depends on the application domain. We considered that variations of less than 0.5m/s were not interesting for this specific case. For the min_time argument, we have selected the 90th percentile of the whole time series (in this case, 7 seconds) to avoid choosing a too arbitrary value.

In Figure 3.21a, we can see the original data used in the example and the smoothed version that is considered as actual input for the ANBA algorithm. Smoothing input signals is not required, therefore we do not enforce the application of any specific method. However, it is advisable to smooth out values in order to eliminate peaks according to a previous analysis of the dataset. As can be noticed in Figure 3.21a, the original data has been changed just enough for discard abrupt changes of values. In this case, a exponential weighted moving average with a span of 5 data points has been applied to smooth out the peaks in the signal.

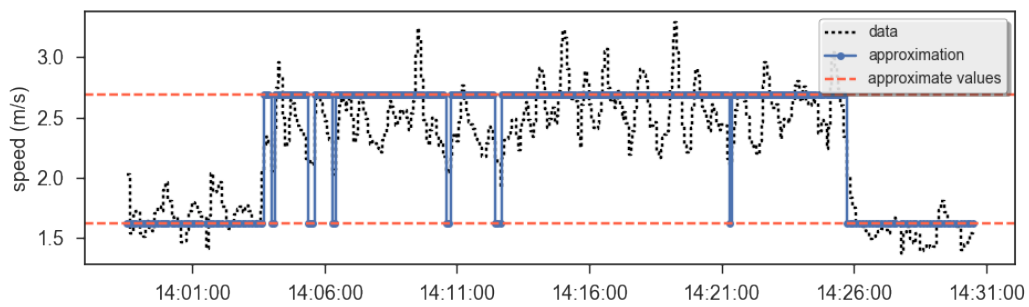
The first part of ANBA algorithm, which defines the number of classes



(a) Raw speed data and its smoothed version.



(b) Breaks and medians (CAVs).



(c) Final AV and a first approximation.

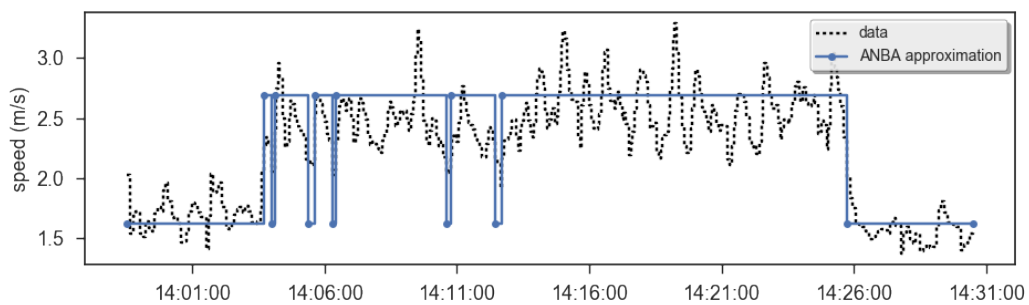
(d) Final representation after eliminating points with duration less than min_dist .

Figure 3.21: Stages of the ANBA process from the raw data to the approximation in an example speed time series extracted from a trajectory.

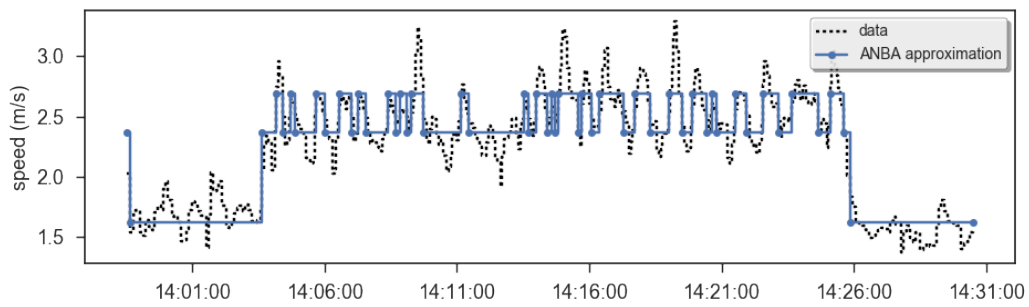
based on a minimum *GVF*, returns 3 as the optimal number of classes for achieving a *GVF* of 0.5 or higher. In this case, the Fisher-Jenks algorithm return the following breaks: [1.39471441, 2.01465982, 2.58280389, 3.31918128]. These values are shown in Figure 3.21b as dotted lines. The median value of each class is also shown (dashed lines) and correspond to [1.64301596, 2.40163605, 2.74171959]. The medians constitute the *CAV* in our method.

In Figure 3.21c, we show a stage where the *MERGE CANDIDATE APPROXIMATIONS* procedure has already been applied to *CAV*, merging the last two *CAVs*, returning the following *AVs*: [1.64301596, 2.57167782]. It is also possible to see the first approximation to *AVs*. The final result is shown in Figure 3.21d, where the points with less than seven seconds have been repositioned to reduce the number of generated segments. The two small segments have 8 and 10 seconds each, and were left as is according to the *min_time* parameter in the example (7 seconds).

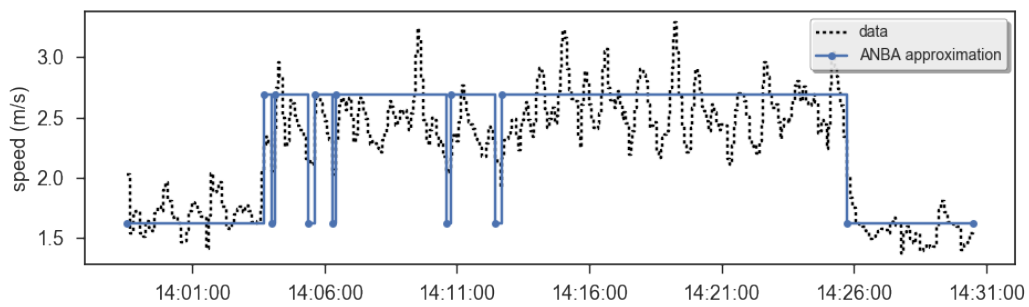
It is important to highlight that, in the example above, the parameters have been set in a way to have a representation of the input signal in a higher level of granularity. Therefore, the small variability in speed that can be observed mostly in the middle of the trajectory have been ignored in the *ANBA* output. In another configuration, changing the *min_dist* parameter to a higher value (or omitting it) would not lead to merging the two upper *CAVs* and some of the speed variations in this part could have been represented, for instance. Figure 3.22 illustrates this difference, where we can also notice the impact in the number of segments (48 vs 7).

3.3.2.3 Evaluation

We have evaluated our method with a real dataset acquired from a third-party mobile application. The subset used for the experiments comprise 106 time series of the speed of trajectories captured between July 2015 and August 2016. Table 3.6 summarizes the dataset characteristics: duration in seconds,



(a) More detailed representation



(b) Less detailed representation

Figure 3.22: Differences in level of detail with two and three approximate values

Table 3.6: Dataset summary

	duration	length	number of points
mean	1938	3738.84	323.48
standard deviation	742	651.84	48.39
minimum	949	2219.94	207
25% percentile	1526	3278.85	293.5
50% percentile	1860	3772.37	320.5
75% percentile	2073	4152.92	366.75
maximum	5340	6961.6	399

length in meters, and number of points.

It is hard to define equivalent parameters for a fair comparison through all algorithms as either the quantity and meaning of arguments varies across the considered methods. Moreover, we show the results for a singular set of parameters for each method. Therefore, the comparison made in this section has the objective of validating the viability of the [ANBA](#) method for time series clustering but it is not exhaustive in terms of variation of parameters. For the testing dataset used in this thesis, we have set the following parameters:

- PLS
 - *max_error*: 1m/s
- PAA
 - *paa_size*: the number of segments is defined as a function of the total duration of the trajectory to result in one segment per minute.
- ANBA
 - *min_dist*: 1m/s

- *min_time*: 90% percentile of the time between points
- *min_gvf*: 50% (only for ANBA-gvf)
- *n_classes*: 3 (only for ANBA-n)

Before applying the ANBA algorithm, the data was smoothed by an exponentially weighted moving average [Roberts 1959] with span = 5 (the same as in the example of Figure 3.21) in order to remove abnormal peaks caused by GPS inaccuracies. As a final step for all methods, the stops calculated by the MSN algorithm have been reincorporated to the approximated signal, but the time for computing the move, stop and noise segments of trajectories have not been taken into consideration.

For the testing dataset, we have observed that the number of classes chosen to achieve a minimum GVF of 0.5 was 3 classes in most cases. More specifically, when *min_gvf* was set to 0.5, the chosen number of classes was equal to 3 in 85% of cases. To determine this number, running the ANBA algorithm for one trajectory would make 4 calls (3 for finding the number of classes and another one to run the algorithm with the supplementary steps). Four classes was enough to achieve GVF of 0.5 in only 3%, five classes in 9%, and only one case 6 classes were needed for the classification. This is the reason for selecting 3 as the number of classes for the cases when GVF was not used. Running the ANBA algorithm with these two possible set-ups allow us to analyze the impact of the GOODNESS_OF_VARIANCE_FIT procedure that constitutes the first step of ANBA when *min_gvf* is preferred over *n_classes*.

In order to compare the different algorithms, we have select the following metrics: the Pearson’s Correlation, the Root Mean Squared Error (RMSE), the running time, and the number of generated segments. The running time has been taken as the best of three consecutive runs of each algorithm for each trajectory.

The Pearson’s correlation, also known as Pearson product-moment corre-

lation coefficient (Equation 3.8), takes values in the range $[-1, 1]$ and measures how close two series are to each other where the stronger the correlation, the closer the values is to 1. As the values are ranked, it actually determines the level to which two variables are proportional to each other.

$$r(Y, \hat{Y})^2 = \frac{\sum_i (\hat{Y}_i - \bar{Y})^2}{\sum_i (Y_i - \bar{Y})^2} \quad (3.8)$$

The Root Mean Squared Error (**RMSE**) is defined in Equation 3.9 and it gives an information similar to the correlation. It measures the similarity between two series by calculating the mean difference point by point. It considers the square of each difference in order to punish more the bigger errors. The measure unit that it outputs is the same as the series in question, e.g. meters/seconds. Therefore, it complements the comparison given by the Pearson's Correlation. The **RMSE** is defined in Equation 3.9.

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (3.9)$$

Figure 3.23 shows a summary of the tests performed. **PAA** is the baseline in terms of running time (being far superior to other methods) and number of segments (as it is a function of its parameters). However, **PAA** has the worst performance regarding the correlation and an average performance in **RMSE**.

The Top-Down algorithms have shown the best results for correlation and **RMSE**, however, they generate a high number of segments and also had a bad performance in running time. This clearly shows the trade-off between correlation and **RMSE** versus the number of segments and running time, which is valid for all methods.

The Sliding-Window algorithms had their worst performance in correlation and number of segments, being average in terms of **RMSE** and showed a good performance in running time. The Bottom-Up strategies also had mostly

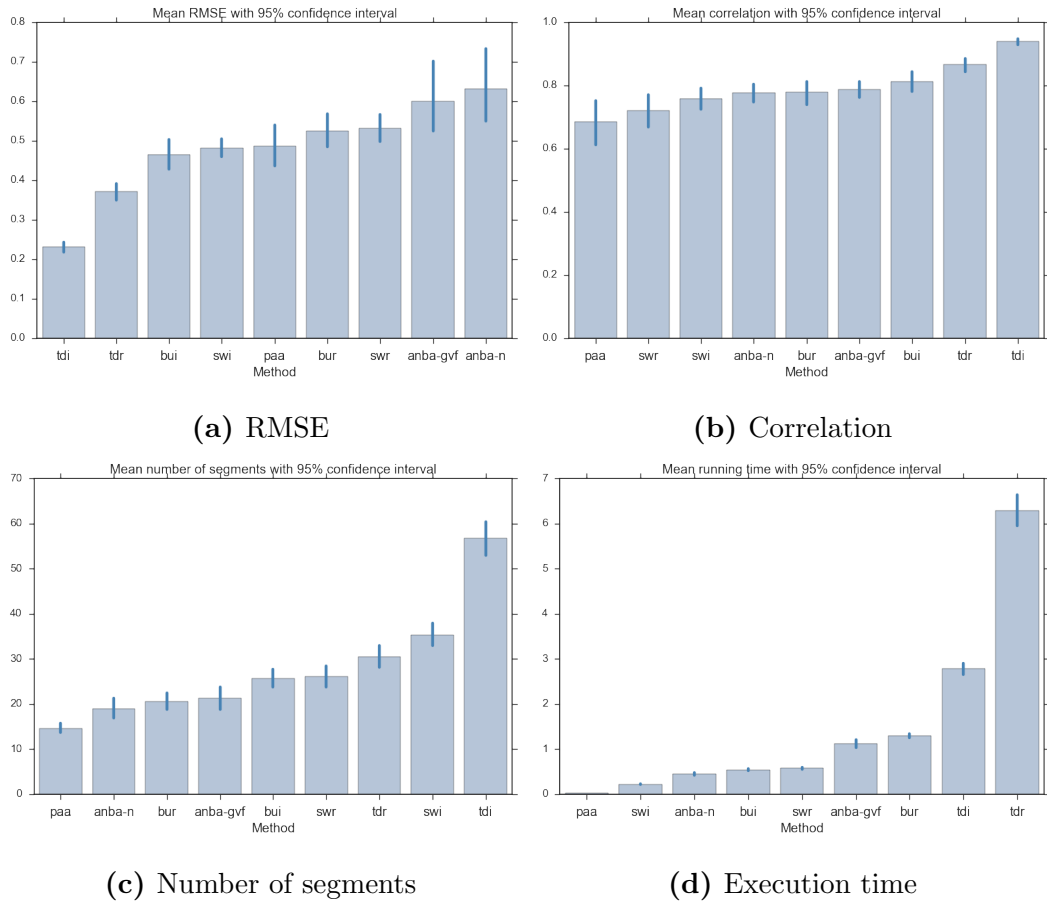


Figure 3.23: Methods considered in the evaluation. **paa**: Piecewise Aggregate Approximation, **tdi**: Top-Down with Interpolation, **tdr**: Top-Down with Regression, **bui**: Bottom-Up with Interpolation, **bur**: Bottom-Up with Regression, **swi**: Sliding Window with Interpolation, **swr**: Sliding Window with Regression, **anba-n**, **anba-gvf**: Adaptive Natural Breaks Approximation with number of classes and GVF, respectively. RMSE is in meters per seconds and running time is in seconds.

	rmse	correlation	time	segments
paa	0.486	0.685	0.016	14.594
anba-n	0.632	0.777	0.446	18.934
bur	0.525	0.779	1.294	20.509
anba-gvf	0.600	0.787	1.116	21.302
bui	0.464	0.813	0.540	25.689
swr	0.531	0.721	0.572	26.057
tdr	0.371	0.866	6.283	30.453
swi	0.482	0.758	0.216	35.321
tdi	0.230	0.939	2.777	56.708

Table 3.7: Mean value of metrics for the testing dataset.

average performance in all metrics.

As for the two versions of the **ANBA** algorithm, they have achieved the worst **RMSE** (also with the widest confidence intervals) with a mean of approximately $0.6m/s$. The mean values for all metrics and methods are shown in Table 3.7. Their correlation results showed a good performance, being near the mark of 80%. The small number of generated segments in an interesting aspect of the algorithm and it is even more important when put into perspective with the good correlation, as it is of interest to balance these two metrics).

The most different results of the two **ANBA** versions was observed in the execution time. This can be explained by the multiple calls to Fisher-Jenks algorithm that are made in the **ANBA-gvf** as opposed to just one call in **ANBA-n**. As all other metrics of the two versions did not suffered a high impact, a strategy that can be recommended for this dataset is to run the **ANBA-gvf** version with a small subset to find the most recurrent number of classes for the desired level of **GVF** and then run the **ANBA-n** version for all

dataset instances.

One advantage of **ANBA** is that it may save some storage space in a Semantic Web setting where the time series approximation should be saved in a triple store. For instance, if two approximate values are enough to discretize a time series of 20 segments, only two entities representing each value have to be created and only the extent of each segment have to be individually specified. The combination of the extent instances with the corresponding approximate value instance by relationships would form the final series representation in the triple store. To achieve this result, the output of other algorithms like **PAA** and **PLS** would have to be modified because they generally yield different values even for segments near to each other. The **ANBA** method provides this advantage out-of-the-box because of the characteristics of Fisher-Jenks algorithm and also because of the merging procedure of approximate values.

3.3.3 ISA: Intersection-based Spatial Annotation

Automatic annotation of trajectories with spatial features is highly dependent on the application domain. For instance, a public transport application is probably more concerned with entities like bus stops, street names, and railway lines than an application about tourism which by its turn may be more interested in the location of museums, landmarks, and hotels.

In most of the related literature, the efforts done with this kind of annotation aims the identification of Regions of Interest (RoIs) of a moving object when it is not moving. In the work of **Yan et al. [2011]**, the authors propose an algorithm for trajectory annotation with RoIs. However, no standard format of external data is mentioned, a rather limited dataset is used (Swisstopo from Switzerland) and, from what can be inferred of their work, all intersections are computed features disregarding the type of polygons. This last aspect can be specially time consuming for long trajectories.

An extensive data source of geographical information can be found in the OpenStreetMap⁴⁴ (OSM) project, which have as objective to build a free geographic database of the world [Bennett 2010]. It includes a numerous variety of geographic entities such as streets, footpaths, buildings, waterways, beaches, parks, administrative boundaries, land use, etc. Based on volunteered geographic information, the OpenStreetMap (OSM) databased can be changed by any individual and is constantly evolving, consisting in one of the largest collaborative database of geographic features on the web.

The OSM data model is based on three data structures – nodes, ways, and relations – and a tagging system. Nodes represent specific points characterized by latitude and longitude. Ways connect a set of nodes that constitute linear features and area boundaries. Relations can be used to group a set of elements (nodes, ways, and/or other relations) optionally attributing roles to the involved elements. Moreover, relations can be used to describe different relationships among elements, for instance, multiple ways representing streets can be members of a relation that constitutes a bus route, in another example, a relation can be used to model a complex feature composed by more than one way, constituting a multipolygon (e.g. an archipelago). Additionally, relations can be employed to specify features that are not rendered in usual maps, such as turn restrictions on streets that are more useful for routing services.

Besides the primitive data structures, another vital part of OSM's data model is its tagging system. Tags are key-value pairs that can be associated with any node, way or relation. There is no formal enforcement on which tags can be used and how they are used, meaning that anyone can create a new tag when needed. However, the community of contributors generally consults the OSM Wiki⁴⁵ where guidelines are described for the most popular tags. Examples of tags are: *highway = primary*, *amenity = parking*, *waterway =*

⁴⁴<http://www.openstreetmap.org>

⁴⁵https://wiki.openstreetmap.org/wiki/Map_Features

river.

There are many ways of accessing *OSM* data for usage in third-party applications. One can directly download XML files containing specific areas or even the entire “planet” file, which has 52GB at the time of this writing. There is also a REST API that allow retrieving and modifying the *OSM* database by authenticated users.

A number of other APIs are also available mainly with read-only access to *OSM* data. A prominent one is the Overpass API⁴⁶ along with its domain-specific language Overpass QL. This initiative also features Overpass Turbo⁴⁷, a web-based tool for querying and visualizing *OSM* data.

In the context of the Semantic Web, an important project is the LinkedGeoData (*LGD*). LinkedGeoData (*LGD*) consists in translating *OSM* data originally stored in a relational database into *RDF* triples following a lightweight ontology [Stadler et al. 2012]. The project proposes two different namespaces: *lgd*⁴⁸ and *lgdo*⁴⁹ in order to separate instances from the ontology. The mapping from *OSM* data to *RDF* is performed based on the unique IDs of *OSM* features. For instance, the Eiffel Tower has ID 5013364 in *OSM* and it is represented as the resource *lgd:way5013364* in *LGD*.

The LinkedGeoData ontology was built through a semi-automated process that consisted on the creation of *tag mappers* that match specified tag patterns to transformation functions written in Java for creating *RDF* descriptions. For instance, a tag mapper could specify that the *OSM* features containing the tag *{amenity: school}* had to be mapped to the *lgdo:School* concept or, in a more general case, any feature containing the tag key *tourism* had to be mapped to the *lgdo:TourismThing* concept in the *LGD* ontology.

Taking into consideration the large variety of tags in *OSM*, this mapping

⁴⁶<http://overpass-api.de>

⁴⁷<http://overpass-turbo.eu>

⁴⁸<http://linkedgeodata.org/triplify/>

⁴⁹<http://linkedgeodata.org/ontology/>

process can also be viewed as a filtering mechanism if we consider that only tags that hold important semantic value are selected. In this sense, features containing no tags or only informative tags like *name*, *source*, *comment*, *note*, etc. can be discarded as they have no important semantic information about the feature (e.g. whether it is a building, a park, a river, etc.).

At the time of this writing, there are two ways of directly querying the LGD database apart from direct downloads of data dumps. The first possibility is a Virtuoso endpoint⁵⁰ and the other one is a web-based interface⁵¹. Both LGD endpoints allows the submission of SPARQL queries, however, for spatial queries, Virtuoso built-in functions⁵² should be used as this triple store's version do not support OGC's GeoSPARQL standard for spatial queries yet.

Despite the focus of this thesis being the annotation of trajectories with technologies suited for the Semantic Web stack, not all use cases could be successfully tested with LGD services due to servers temporary unavailability and due to the fact that LGD data is based on periodic dumps of OSM data, i.e data on LGD does not always reflect the most up-to-date version of OSM. Although we have implemented the working solution using the Overpass API, we also show equivalent SPARQL queries on LGD that would achieve the same results. Another option for querying OSM is Linked Open Street Map (LOSM), proposed by Ragone et al. [2016] and in a beta version at the time of this writing.

In our automatic annotation algorithm, we enrich trajectories based on OSM geographic features with relevant tags. In order to achieve this, we have used the Open Semantic Network (OSN) RDF dataset proposed by Ballatore et al. [2012]. By exploring the hyperlinks among the pages that describe OSM

⁵⁰www.linkedgeodata.org/sparql

⁵¹www.linkedgeodata.org/snorql

⁵²<http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VirtGeoSPARQLEnhancementDocs>

tags in the OSM Wiki⁵³, the authors have conceptualized a directed graph containing OSM keys and tags, lexical descriptions, relationships between tags, general internal links, and links to Wikipedia pages. More importantly for our case, the OSN graph also contains links to equivalent classes in the LGD ontology.

By linking the spatial annotation episodes to OSN vocabulary, it is possible to benefit from the semantic similarity rankings between the tags created with the help of co-citation algorithms. For instance, it becomes possible to infer that different geographic features tagged with *waterway = river* and *natural = lake* have some level of similarity as both concepts are related to the concept *natural = water*.

In the STEP framework, we have implemented the ISA algorithm for retrieving OSM features based on the intersection of a trajectory with its geographic context. Additionally, the user has the option of filtering which types of features are selected based on OSM tag keys. The data flow of the algorithm is pictured in Figure 3.24 where the shaded parallelograms represent the parameters of the ISA algorithm. Please notice that some elements of the Figure have been duplicated to avoid edge overlapping. A Python implementation of the ISA algorithm is also available in Appendix C.

The ISA method starts with the verification of the presence of keys informed by the user. If one or more keys are inputted, we check if they should be expanded. In the positive case, a query in the OSN RDF graph is performed to retrieve all other terms that are related to each of the informed keys.

For instance, if the inputted keys variable contains only the *leisure* key and the expand option is enabled, this query would return 53 new keys. However, some of these new keys does not have a semantic value – e.g. *opening_hours*, *addr:housenumber*, *contact:website* – therefore, we have incremen-

⁵³http://wiki.openstreetmap.org/wiki/Map_Features

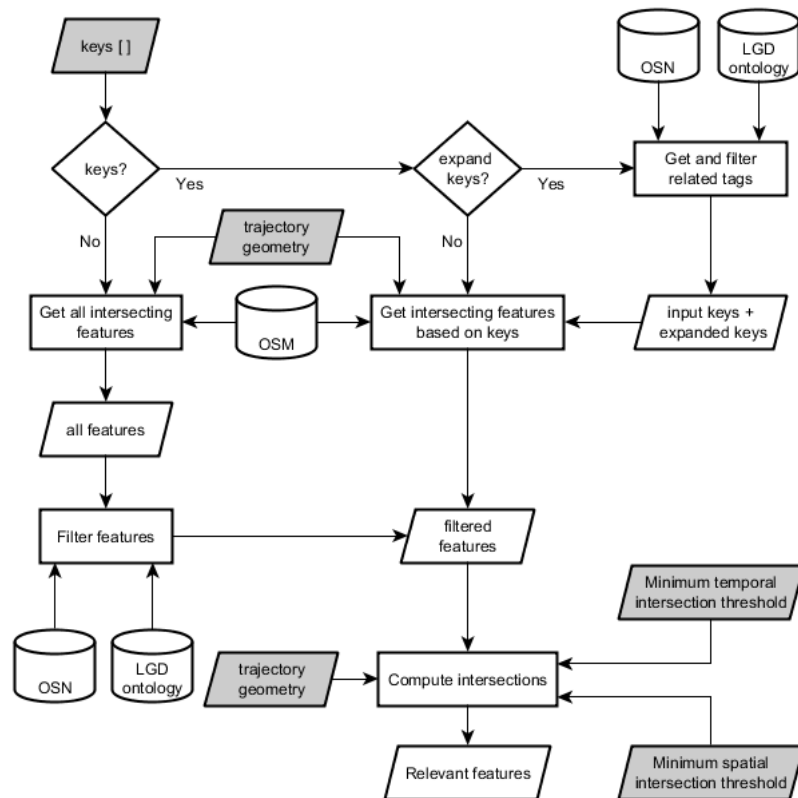


Figure 3.24: Data flow of the intersection-based spatial annotation algorithm with OSM features.

ted this query with a filter to include only OSN concepts that have an exact match with some concept from the LGD ontology. In the case of the example, the *leisure* key would be expanded with the following other keys that have a match in LGD: [*'highway', 'playground', 'route', 'tourism', 'man_made', 'sport', 'waterway', 'harbour', 'shelter', 'building', 'natural', 'fishing', 'landuse', 'boundary', 'amenity', 'barrier', 'shop'*], totaling 17 keys instead of the 53 without the LGD filter. The query that was built by the ISA algorithm for this example is shown in Listing 3.6.

```

1 PREFIX skos:<http://www.w3.org/2004/02/skos/core#>
2 PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3 PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
4 PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
5
6 SELECT ?subject ?relatedTerm ?lgdConcept
7 {
8   ?subject skos:related ?relatedTerm .
9   ?relatedTerm skos:exactMatch ?lgdConcept .
10
11  FILTER (REGEX(STR(?subject),
12              "http://spatial.ucd.ie/lod/osn/term/k:(leisure).*"))
13  FILTER (STRSTARTS(STR(?lgdConcept),
14                  "http://linkedgeodata.org/ontology"))
15 }

```

Listing 3.6: SPARQL query that finds all OSN concepts that are related to the *leisure* key. Additionally, only related terms that have an exact match in LGD ontology are returned.

The next step of our method consists in retrieving intersecting OSM features that have tags matching the keys informed by the user and the expanded keys if this option was chosen, otherwise only the explicit keys informed by the user are considered. If no keys are previously informed by the user, it means that there is no interest in narrowing down the types of features returned from OSM and the algorithm should make no distinction of retrieve features based

on their tags.

When no keys are specified in the parameters, all intersecting [OSM](#) features are retrieved. At the end of this process, an Overpass QL query is built according to the choices of the user, that reflect in the regular expression that is used in Overpass QL query. Listing 3.7 shows an example that matches [OSM](#) ways and relations that intersect the line defined with the `poly` keyword and have at least one of the keys *leisure*, *natural*, or *landuse*.

```

1 [out:json];
2 (way[~"leisure|natural|landuse"~"."]
3 (poly:"45.18210998 5.73425791 45.1824220976 5.7353382904 ...");
4 rel(bw);
5
6 relation[~"leisure|natural|landuse"~"."]
7 (poly:"45.18210998 5.73425791 45.1824220976 5.7353382904 ..."););
8 out geom;

```

Listing 3.7: Example of Overpass API query that retrieve all ways and relations intersecting the geometry specified in the *poly* field. The tags are selected by a regular expression.

An equivalent query, in [SPARQL](#), is shown in Listing 3.8 and can be sent to [LinkedGeoData](#) endpoints. It is worth noticing that this is the only procedure that depends on the availability of an external server. For the case of other queries shown in this section, only a [RDF](#) dump of the [OSN](#) and a serialization of the [LGD](#) ontology are sufficient to perform the filtering of relevant tags.

In the cases when no keys are given as parameters, all intersecting [OSM](#) features are returned. However, some filtering process is still necessary. Once the intersecting features are retrieved, each one is verified to check if at least one of its tags have a corresponding concept mapped in [OSN](#) and if that concept have an exact match in the [LGD](#) ontology. This process is similar to the one performed in the keys expansion procedure and helps in maintaining only features that have a defined semantic meaning in the context of the [LGD](#)

```

1 PREFIX ogc: <http://www.opengis.net/ont/geosparql#>
2 PREFIX geom: <http://geovocab.org/geometry#>
3 PREFIX lgdo: <http://linkedgeodata.org/ontology/>
4 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
5
6 SELECT ?subject, ?type, ?label
7 FROM <http://linkedgeodata.org> {
8     ?subject
9     a ?type ;
10    geom:geometry [
11        ogc:asWKT ?geometry
12    ] .
13    OPTIONAL {
14        ?subject rdfs:label ?label .
15    }
16    OPTIONAL {
17        ?subclass rdfs:subClassOf ?type .
18        ?subject a ?subclass .
19    }
20    OPTIONAL {
21        ?type rdfs:subClassOf ?superclass .
22    }
23    FILTER (?type IN (lgdo:Leisure, lgdo:Natural, lgdo:Landuse))
24    FILTER (bif:st_intersects(?geometry,
25                                bif:st_geomFromText("LINESTRING(...)",
26                                0.001))
27 }

```

Listing 3.8: SPARQL query to retrieve all subjects, types, geometries and labels of entities that intersect the area around 1 meter of a given line (the trajectory) and have one of the classes inside the “IN” filter.

ontology, which we adopted to measure the significance of OSM tags.

Once OSM features have been retrieved and filtered, the ISA algorithm computes the intersection between the trajectory and each feature polygon. As OSM ways are represented by nodes, we create polygons based on the latitude and longitude data. A variable buffer can be applied to the polygon

in order to both compensate GPS accuracy errors and consider features not directly crossed by the moving object. The default length of this buffer was set to 20 meters, but it can be freely changed.

The final result of the ISA method retrieves the OSM features that reach at least one of the two criteria represented by the Minimum Temporal Intersection Threshold and the Minimum Spatial Intersection Threshold. These are parameters that range from 0 to 1 and represent the percentage that each feature must meet to be considered as relevant. This avoids the selection of OSM features where the moving object has passed quickly or has barely touched.

Figure 3.25 shows an example of polygons returned by the Overpass API where the keys *leisure*, *natural*, and *landuse* have been used as filtering criteria (see Listings 3.7 and 3.8). For each of these features, it is then calculated how long the moving object has stayed in it both in terms of distance and duration.

The park (1) in Figure 3.25, has around 30% of coverage both in time and distance. All other features also does not show a big difference between distance and duration coverages. The building (2) has 4%, feature 3 has 2%, the grass field (4) has 4%, feature 5 has 1%, another park (6) has around 34% of coverage, a strip of wood (7) has 43%, the river (8) has 34%, and the residential block (9) covers about 32% of the trajectory space and time. Table 3.8 have the detailed proportions of each feature regarding the example trajectory.

In the example of Figure 3.25, with δ_t and δ_s both set to 20%, only features 1, 6, 7, 8, and 9 are considered as relevant. In this case, the relevant features correspond to two parks, a wood, a river, and a residential area.

With the output of the spatial annotation algorithm, it is possible to create instances of the STEP ontology. The OSM features can be seen as contextual

Table 3.8: Feature tags, duration ratio, and distance ratios regarding the example trajectory.

Feature	OSM tags	Duration ratio	Distance ratio
1	{'leisure': 'park', 'name': 'Parc Paul Mistral'}	29.11%	30.28%
2	{'leisure': 'sports_centre', 'name': 'Halle Clemenceau'}	4.29%	3.96%
3	{'landuse': 'forest'}	2.09%	1.74%
4	{'landuse': 'grass'}	4.19%	4.34%
5	{'leisure': 'sports_centre', 'name': 'Palais des Sports'}	0.82%	0.77%
6	{'leisure': 'park', 'name': "Parc des Berges de l'Isère"}	33.4%	35.3%
7	{'natural': 'wood'}	42.48%	43.82%
8	{'natural': 'water', 'name': "L'Isère"}	34.83%	34.29%
9	{'landuse': 'residential', 'name': 'Châtelet Abbaye Jouhaux'}	32.48%	32.66%

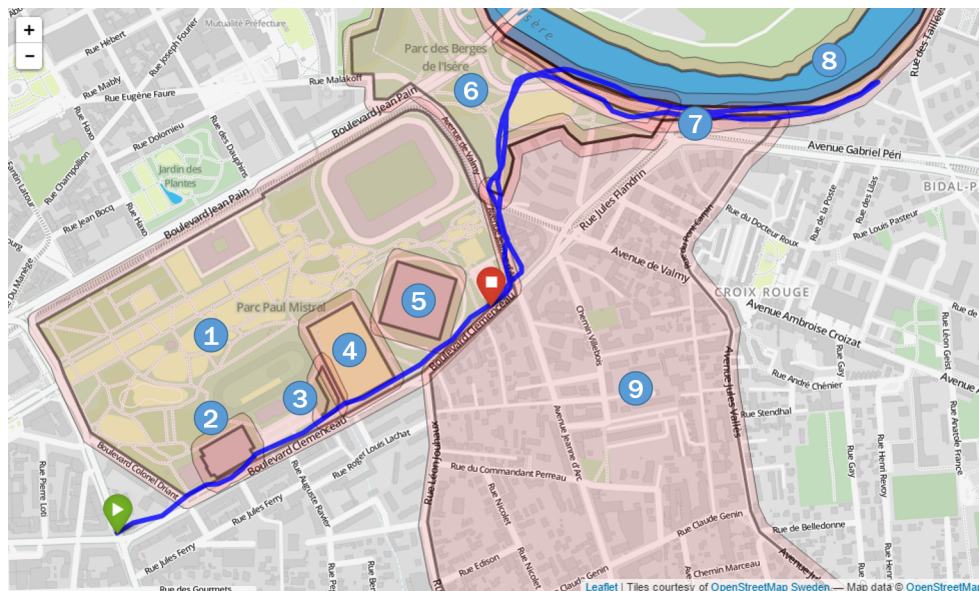


Figure 3.25: Example of polygons retrieved by queries on Overpass API

elements and OSN concepts can be easily linked to these elements as they follow a well-defined link structure, e.g. for the tag *waterway = riverbank*, the corresponding OSN link is <http://spatial.ucd.ie/lod/osn/term/k:waterway/v:riverbank>. An example of RDF code generated by the STEP framework can be seen in Listing 3.9.

```

1 <rdf:Description
2   rdf:about="http://example.com/resource/2njWC77BrXkr26ZXKxPZZA">
3   <owl:sameAs
4     rdf:resource="http://linkedgedata.org/triplify/way4014810"/>
5   <rdf:type
6     rdf:resource="http://purl.org/net/step#ContextualElement"/>
7   <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
8     Parc Paul Mistral
9   </rdfs:label>
10  <skos:related
11    rdf:resource="http://spatial.ucd.ie/lod/osn/term/k:leisure/v:park"/>
12 </rdf:Description>

```

Listing 3.9: Excerpt of RDF/XML file automatically generated by the STEP framework.

In Figure 3.26, it is possible to see some examples of OSM features that are selected in different situations. For these cases, no keys have been informed to the ISA algorithm and the spatial and temporal thresholds have both been set to 0.2, meaning that at least 20% of the trajectory’s duration and distance should be intersected by an OSM feature in order to that feature to be considered as relevant.

It is also shown the LGD ontology concepts that have been identified based on the OSM tags. It is possible to notice that the LGD concepts also helps to describe the spatial context, besides serving its role of filtering important categories of geographical features, such as parks, forests, beaches, farms, vineyards, etc.

Some aspects of these algorithms can be further improved in future work.



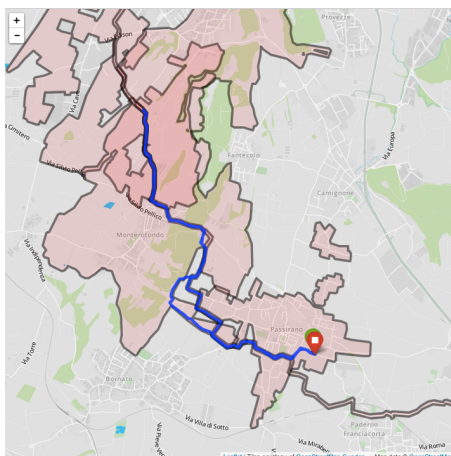
Cycleway, LanduseForest, Parking, Fence, Stream, HighwayResidential, Pedestrian, SportsCentre, Footway, Park, Bus, HighwayService, Steps, Secondary, NaturalWater, Grass, Path, LanduseResidential, NaturalWood, RouteBicycle

(a) A trajectory in Grenoble, France.



Cycleway, TrunkLink, NatureReserve, HighwayResidential, NaturalBeach, Pedestrian, Footway, HighwayTrack, Park, Bus, LivingStreet, HighwayService, Secondary, LanduseResidential, Trunk

(b) A trajectory in Rio de Janeiro, Brazil.



Cycleway, LanduseForest, HighwayResidential, Unclassified, Footway, HighwayTrack, Park, HighwayService, Farmland, LanduseIndustrial, Path, Vineyard, LanduseResidential, Tertiary, LanduseRetail, Mtb, RouteBicycle

(c) A trajectory in Brescia, Italy.

Figure 3.26: Some example trajectories along with the features selected by the ISA algorithm and the corresponding LinkedGeo-Data concepts.

First, it relies on the input of the tags keys of interest, excluding the tag values. Depending on the application, the semantic information can be more valuable in the tag value instead of the tag key. This has not been done due to limitations in the Overpass API, that do not accept complex regular expressions at the time of this writing.

Regarding the intersection among polygons and trajectory, the method has room for improvement if one considers the types of features by their semantic similarity. For instance, a group of three features with respective tags: *landuse = forest*, *landuse = grass*, and *landuse = wood* as the algorithm is concerned only with individual polygons, this group of semantic related features may not achieve the thresholds set by the user but they would still be important for describing the trajectory's spatial context.

Another aspect is the previous knowledge about the tag keys that should be informed by the user. Considering that this algorithm can be scaled to a large number of trajectories, it can be extended to infer which kinds of tag keys and values are more frequent and recommend tags that seems to be more relevant. However, this recommendation should be validated by domain experts. This is partially solved by the keys expansion mechanism aided by the filtering process that relies on the LGD ontology, but is can further studied how upper ontologies such as DOLCE⁵⁴ or OpenCyc⁵⁵ can help in identifying more adequate concepts for a particular domain.

⁵⁴<http://www.loa.istc.cnr.it/old/DOLCE.html>

⁵⁵<http://sw.opencyc.org>

Conclusions and Perspectives

Contents

4.1	Conclusions	131
4.2	Future Work	134

4.1 Conclusions

The trending increase in production of large volumes of spatiotemporal data calls for innovative methods of building representations of trajectory data in higher abstractions levels. Trajectory annotation is a critical task at the initial steps of any intelligent system for data mining and knowledge discovery about trajectory datasets. The selection, adaptation or creation of a data model is another important factor to ponder before implementing such systems.

In this thesis we have dealt initially with the challenge of representing spatiotemporal phenomena encompassing trajectories and its contextual elements in an ontological approach. The **STEP** ontology has been proposed and compared to relevant works in the area. We believe that the **STEP** ontology is flexible enough to represent trajectories in different applications and it concentrates a series of features that are only partially covered by other models up to this date. We have focused on building a model that is capable of representing generic episodes in a hierarchy composed by Features of Interest and episodes in different granularity levels, relating contextual element

to individual episodes, expressing qualitative and quantitative values for segments as well as for entire trajectories and contextual elements. Moreover, by allowing different types of extent representations, namely temporal, spatial, and spatiotemporal, we enforce the flexibility aspect of the ontology to adapt to different uses.

We argue that ontologies are similar to software artifacts in that they may have their own life cycle and are subject to evolution. This means that the distribution aspect of an ontology should not be overlooked. Therefore, we have made efforts to make the **STEP** ontology available not only in this thesis, but also in adequate channels like its namespace⁵⁶, website⁵⁷ and the Linked Open Vocabularies repository⁵⁸. Although none of these services are assured to be available every time, it is expected that the ontology continues to be publicly available.

Regarding the usage of Semantic Web technologies for spatiotemporal analysis, it is important to notice that the same advances of relational databases in terms of performance are not yet present in spatially enabled triple stores [Patroumpas et al. 2014]. This relevant issue should be taken into account for production-ready systems, where a hybrid solution can be a viable option.

In order to bridge the gap between raw trajectories, annotation algorithms, and semantic graphs based on our ontology, we have devised the **STEP** framework, which is composed by a graph-object mapping layer that enables the creation of **STEP** instances that can later be transformed and serialized into triples. Besides, the framework also allows a central point for accessing preprocessing and annotation algorithms that can reuse the output of other methods present in the **STEP** framework.

One method proposed in this thesis is the Move-Stop-Noise detection al-

⁵⁶<http://purl.org/net/step>

⁵⁷<http://talespaiva.github.io/step/>

⁵⁸<http://lov.okfn.org/dataset/lov/vocabs/step>

gorithm, that is tailored to trajectories that have been sampled at irregular intervals of time or have been preprocessed to eliminate redundant points at near locations. This characteristic violates a basic assumption made by state-of-the-art algorithms for stay point detection and have motivated our work to fill this gap. Besides, the [MSN](#) method has also been designed to be independent of external data (e.g. the underlying geographic features) and is free of parameters in terms of time and space, i.e. there is no need of defining hard thresholds such as specifying that a stop has to be equal of greater than 10 seconds, for instance. Conversely, the parameters used in our method are informed as absolute numbers as proposed by robust outlier discovery that can be adapted if needed.

Many of trajectory attributes can be expressed as time series. Based on this, we have proposed a new time series compression method that innovates in searching for a reduced set of values that can be used to reconstruct a previously compressed series with relevant correlation with the original signal.

As discussed by [Esling and Agon \[2012\]](#), it is desirable for a data representation proposal for time series to allow significant reduction of the data dimensionality, emphasize on fundamental shape characteristics on both local and global scales, have low computational cost for computing the representation, have a good reconstruction quality from the reduced representation, and be insensitive to noise or have implicit noise handling. We have presented the [ANBA](#) method that covers these requirements.

We argue that after removing noise from trajectories by using the [MSN](#) method, the time-based attributes of trajectories such as speed can be slightly smoothed to remove white noise. Then this signal can be inputted to the [ANBA](#) algorithm in order to construct episodes based on either an absolute number of classes or a relative parameter in the ranging from 0 to 1 related to the degree of similarity between the input series and the resulting episodes.

Although having compared the ANBA method with PAA and PLS, we cannot affirm that this method is better for all kinds of time series for two main reasons: first, it has been tested with datasets of limited size between 200 and 400 observations. Second, only datasets of one domain have been used (spatiotemporal time series). Further testing should be carried with classical datasets. e.g. electrocardiogram time series, if one wants to generalize the method for any kind of time series.

Complementing the contributions of this thesis, we have included the Intersection-based Spatial Annotation algorithm in our framework. This algorithm explores the relationships among OpenStreetMap tags issued from the Open Semantic Network Ballatore et al. [2012]. We have demonstrated that our method allows the discovery of new tags taking into consideration an initial set of keys informed by the user and is also capable of creating a high level description of the environment where the trajectory took place based on its spatial join with OSM polygons.

4.2 Future Work

The results accomplished in this thesis allows some future developments in both improving the current results and extending the proposed framework.

Ontologies, like software, are living artifacts subject to change. It is natural for ontologies to evolve to cover more use cases and to be linked to other ontologies. For instance, at the time of this writing, the W3C's Spatial Data on the Web Working Group (SDWWG)⁵⁹ is actively developing a new version of the OWL:Time ontology. Therefore, the STEP ontology is likely to be extended as new applications use it.

The STEP framework has the potential to evolve to contain more semantic trajectory related algorithms focused on segmentation and enrichment proce-

⁵⁹<http://www.opengeospatial.org/projects/groups/sdwwg>

dures. It can also be extended beyond annotation and include more complex methods of spatiotemporal data mining. Mobility profile building regarding dynamic aspects of trajectories and spatial relations with geographical features, for instance, is a possible activity that involves the evolution of both the data model and the algorithms presented in this thesis.

Regarding the detection of stop, move, and noise episodes, it can be envisaged the application of other algorithms, notably supervised learning ones, as the algorithm proposed in this thesis takes advantage only of statistical properties of individual trajectories. For the practical usage of the supervised approach, it is needed some amount of training data, which brings the necessity of previous manually annotate trajectory data with known labels as input for the algorithms. Therefore, a tool for visually annotate trajectories with stop, move, and noise segments can be an interesting contribution. This may improve results by specializing the algorithms for a heterogeneous scenario where various devices capture positional data using their own sampling rate. This idea can also be extended for building other types of episodes.

In the context of time series representation, the comparison with other state-of-the-art algorithms can be interesting. For instance, compression by Discrete Wavelet Transform with Haar wavelets and its derivation Adaptive Piecewise Constant Approximation (APCA). In fact, the final output of ANBA is more similar to APCA than the other compared methods, but due to APCA's complex implementation [Aghabozorgi et al. 2015], this has not been done yet. As discussed in its evaluation, further comparisons with other methods as well as with other datasets should be performed to better assess the ANBA approximation results. Moreover, the algorithm has been designed for off-line processing, therefore, further work with streaming scenarios can be envisaged. Furthermore, the step of approximating AVs, which is done in $O(n^2)$ can be improved for reducing its computational complexity.

Complex topological relations among trajectories and context may be a subject of interest for novel automatic annotation algorithms. For now, we have used only the 'nearby' keyword to express a relationship in the ISA procedure. Interesting extensions of this algorithm to support more complex annotations (e.g. 'crosses', 'around', 'next to') would be valuable to give more detailed insights about how the moving object interacts with the environment.

Publications

Some parts of this thesis are based on the following papers:

- Nogueira, T.P. and Martin, H. 2014. Qualitative Representation of Dynamic Attributes of Trajectories. Short paper. Proceedings of the AGILE'2014 International Conference on Geographic Information Science. (2014), 3–6.
- Nogueira, T.P., Braga, R. B., Martin, H. 2014. An Ontology-based Approach to Represent Trajectory Characteristics. Computing for Geospatial Research and Application (COM.Geo), 2014 Fifth International Conference on. (2014), 102–107.
- Nogueira, T.P. and Martin, H. 2015. Querying semantic trajectory episodes. Proceedings of the 4th ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems - MobiGIS '15 (Seattle, WA, USA, 2015), 23–30.

Bibliography

Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering – A decade review. *Information Systems*, 53:16–38, oct 2015. doi: 10.1016/j.is.2015.04.007. URL <http://dx.doi.org/10.1016/j.is.2015.04.007>. [Cited at pages 44, 103, and 135]

Sander P A Alewijnse, Kevin Buchin, Maike Buchin, Andrea Kölzsch, Helmut Kruckenberg, and Michel A Westenberg. A framework for trajectory segmentation by stable criteria. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - SIGSPATIAL '14*, pages 351–360, 2014. ISBN 9781450331319. doi: 10.1145/2666310.2666415. URL <http://dl.acm.org/citation.cfm?doid=2666310.2666415>. [Cited at pages 4 and 39]

James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, nov 1983. ISSN 00010782. doi: 10.1145/182.358434. URL <http://portal.acm.org/citation.cfm?doid=182.358434>. [Cited at page 22]

Luis Otavio Alvares, Vania Bogorny, Bart Kuijpers, Jose Antonio Fernandes de Macedo, Bart Moelans, and Alejandro Vaisman. A model for enriching trajectories with semantic geographical information. In *Proceedings of the 15th annual ACM International Symposium on Advances in Geographic Information Systems*, page 1, New York, New York, USA, 2007. ACM Press. ISBN 9781595939142. doi: 10.1145/1341012.1341041. URL <http://portal.acm.org/citation.cfm?doid=1341012.1341041>. [Cited at pages 40, 85, and 101]

Gennady Andrienko, Natalia Andrienko, and Alexandr Savinov. Choropleth

- Maps : Classification Revisited. *Proceedings ICA*, 2:1109–1119, 2001. URL <http://geoanalytics.net/and/papers/ica01.pdf>. [Cited at page 103]
- Gennady Andrienko, Natalia Andrienko, and Stefan Wrobel. Visual analytics tools for analysis of movement data. *ACM SIGKDD Explorations Newsletter*, 9(2):38, dec 2007. ISSN 19310145. doi: 10.1145/1345448.1345455. URL <http://portal.acm.org/citation.cfm?doid=1345448.1345455>. [Cited at page 45]
- Gennady Andrienko, Natalia Andrienko, and Marco Heurich. An event-based conceptual model for context-aware movement analysis. *International Journal of Geographical Information Science*, 25(9):1347–1370, sep 2011. ISSN 1365-8816. doi: 10.1080/13658816.2011.556120. URL <http://www.tandfonline.com/doi/abs/10.1080/13658816.2011.556120>. [Cited at pages 33, 76, and 79]
- Natalia Andrienko, Gennady Andrienko, Nikos Pelekis, and Stefano Spaccapietra. Basic Concepts of Movement Data. In Fosca Giannotti and Dino Pedreschi, editors, *Mobility, Data Mining and Privacy*, chapter 2, pages 15–38. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-75176-2. doi: 10.1007/978-3-540-75177-9_2. [Cited at pages 5, 35, and 36]
- Miriam Baglioni, Jose Antonio Fernandes de Macedo, Chiara Renso, and Monica Wachowicz. An Ontology-Based Approach for the Semantic Modeling and Reasoning on Trajectories. In *Advances in Conceptual Modeling – Challenges and Opportunities*, pages 344–353. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. doi: 10.1007/978-3-540-87991-6_41. URL http://link.springer.com/10.1007/978-3-540-87991-6_{_}41. [Cited at pages 4, 25, 76, and 78]
- Miriam Baglioni, Jose Antonio Fernandes de Macedo, Chiara Renso, Roberto Trasarti, and Monica Wachowicz. Towards Semantic Interpretation of Mo-

- vement Behavior. In Monika Sester, Lars Bernard, and Volker Paelke, editors, *Advances in GIScience*, Lecture Notes in Geoinformation and Cartography, pages 271–288. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-00317-2. doi: 10.1007/978-3-642-00318-9_14. URL <http://www.springerlink.com/index/10.1007/978-3-642-00318-9>. [Cited at pages 26, 76, and 78]
- Andrea Ballatore, Michela Bertolotto, and David C. Wilson. Geographic Knowledge Extraction and Semantic Similarity in OpenStreetMap. *Knowledge and Information Systems*, 37(1):61–81, 2012. doi: <http://dx.doi.org/10.1007/s10115-012-0571-0>. [Cited at pages 120 and 134]
- Jonathan Bennett. *OpenStreetMap*. Packt Publishing, Birmingham, UK, 2010. ISBN 9781847197504. [Cited at page 118]
- Tim Berners-Lee. Information Management: A Proposal. *Word Journal Of The International Linguistic Association*, February 2(May):1–10, 1989. URL <http://www.w3.org/History/1989/proposal.html>. [Cited at page 12]
- Tim Berners-Lee. Linked data-design issues, 2006. URL <https://www.w3.org/DesignIssues/LinkedData.html>. [Cited at page 13]
- Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, jan 2009. ISSN 1552-6283. doi: 10.4018/jswis.2009081901. URL <http://www.igi-global.com/article/linked-data-story-far/37496>. [Cited at page 13]
- Vania Bogorny, Chiara Renso, Artur Ribeiro de Aquino, Fernando de Lucca Siqueira, and Luis Otavio Alvares. CONSTAnT - A Conceptual Data Model for Semantic Trajectories of Moving Objects. *Transactions in GIS*, 18(1):

- 66–88, 2014. doi: 10.1111/tgis.12011. URL <http://doi.wiley.com/10.1111/tgis.12011>. [Cited at pages ix, 32, 33, 34, 73, and 76]
- Reinaldo Bezerra Braga. *LIDU: Location-based approach to IDentify similar interests between Users in social networks*. Phd thesis, Université de Grenoble, 2012. [Cited at page 32]
- Dan Brickley and R.V. Guha. RDF Schema 1.1 - W3C Recommendation, 2014. URL <http://www.w3.org/TR/rdf-schema/>. [Cited at page 15]
- Kevin Buchin, Maike Buchin, Marc van Kreveld, and Jun Luo. Finding long and similar parts of trajectories. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '09*, page 296, New York, New York, USA, 2009. ACM Press. ISBN 9781605586496. doi: 10.1145/1653771.1653813. URL <http://portal.acm.org/citation.cfm?doid=1653771.1653813>. [Cited at page 39]
- Maike Buchin, Anne Driemel, Marc Van Kreveld, and Vera Sacristan. Segmenting trajectories: A framework and algorithms using spatiotemporal criteria. *Journal of Spatial Information Science*, 3(3):33–63, dec 2011. ISSN 1948-660X. doi: 10.5311/JOSIS.2011.3.66. URL <http://josis.org/index.php/josis/article/view/66>. [Cited at page 4]
- Maike Buchin, Helmut Kruckenberg, and Andrea Kölzsch. Segmenting Trajectories by Movement States. In Sabine Timpf and Patrick Laube, editors, *Advances in Spatial Data Handling*, pages 15–25. Springer Berlin Heidelberg, 2013. doi: 10.1007/978-3-642-32316-4-2. [Cited at page 39]
- Elena Camossi, Paola Villa, and Luca Mazzola. Semantic-based Anomalous Pattern Discovery in Moving Object Trajectories. *CoRR*, abs/1305.1, may 2013. URL <http://arxiv.org/abs/1305.1946>. [Cited at pages 4 and 28]

Eliseo Clementini, Paolino Di Felice, and Peter van Oosterom. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. *Proceedings of the 3rd International Symposium on Advances in Spatial Databases (SSD'93)*, pages 277–295, 1993. doi: 10.1007/3-540-56869-7_16. [Cited at page 21]

Eliseo Clementini, Jayant Sharma, and Max J. Egenhofer. Modelling topological spatial relations: Strategies for query processing. *Computers and Graphics*, 18(6):815–822, 1994. ISSN 00978493. doi: 10.1016/0097-8493(94)90007-8. [Cited at page 22]

Victor de Graaff and Maurice Van Keulen. Automated semantic trajectory annotation with indoor point-of-interest visits in urban areas. In *Proceedings of the 31st ACM Symposium on Applied Computing, ACM SAC 2016*, Pisa, Italy, 2016. [Cited at pages 40, 85, and 101]

Somayeh Dodge, Robert Weibel, and Anna-Katharina Lautenschütz. Towards a taxonomy of movement patterns. *Information Visualization*, 7(3-4):240–252, 2008. ISSN 1473-8716. doi: 10.1057/palgrave.ivs.9500182. URL <http://ivi.sagepub.com/lookup/doi/10.1057/palgrave.ivs.9500182>. [Cited at page 36]

Max J. Egenhofer and Robert D. Franzosa. Point-set topological spatial relations. *International Journal of Geographical Information Systems*, 5(2):161–174, 1991. ISSN 0269-3798. doi: 10.1080/02693799108927841. [Cited at page 22]

Philippe Esling and Carlos Agon. Time-series data mining. *ACM Computing Surveys*, 45(1):1–34, 2012. ISSN 03600300. doi: 10.1145/2379776.2379788. URL <http://dl.acm.org/citation.cfm?doid=2379776.2379788>. [Cited at pages 42, 45, and 133]

- Riccardo Falco, Aldo Gangemi, Silvio Peroni, David Shotton, and Fabio Vitali. Modelling OWL ontologies with Graffoo. In *ESWC 2014 Satellite Events*, pages 320–325, Anissaras, Crete, Greece, 2014. Springer International Publishing. doi: 10.1007/978-3-319-11955-7_42. [Cited at page 56]
- Renato Fileto, Marcelo Krüger, Nikos Pelekis, Yannis Theodoridis, and Chiara Renso. Baquara: A Holistic Ontological Framework for Movement Analysis using Linked Data. In *32th International Conference, ER 2013*, pages 342–355, Hong-Kong, China, 2013. doi: 10.1007/978-3-642-41924-9_28. [Cited at page 31]
- Renato Fileto, Vania Bogorny, Cleto May, and Douglas Klein. Semantic Enrichment and Analysis of Movement Data: Probably it is just Starting! *SIGSPATIAL Special*, 7(1):11–18, 2015a. ISSN 19467729. doi: 10.1145/2782759.2782763. URL <http://dl.acm.org/citation.cfm?doid=2782759.2782763>. [Cited at pages 4 and 37]
- Renato Fileto, Cleto May, Chiara Renso, Nikos Pelekis, Douglas Klein, and Yannis Theodoridis. The Baquara2 knowledge-based framework for semantic enrichment and analysis of movement data. *Data & Knowledge Engineering*, 98:104–122, jul 2015b. ISSN 0169023X. doi: 10.1016/j.datak.2015.07.010. URL <http://linkinghub.elsevier.com/retrieve/pii/S0169023X15000555>. [Cited at pages ix, 31, 32, 73, and 76]
- Walter Fisher. On grouping for maximum homogeneity. *American Statistical Association Journal*, 53:789–798, 1958. ISSN 01621459. doi: 10.2307/2281952. [Cited at page 103]
- Barbara Furletti, Paolo Cintia, Chiara Renso, and Laura Spinsanti. Inferring human activities from GPS tracks. In *Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing - UrbComp '13*,

- page 1, New York, New York, USA, 2013. ACM Press. ISBN 9781450323314. doi: 10.1145/2505821.2505830. URL <http://dl.acm.org/citation.cfm?doid=2505821.2505830>. [Cited at page 46]
- Aldo Gangemi. Ontology design patterns for semantic web content. In Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen, editors, *The Semantic Web – ISWC 2005*, pages 262–276. Springer Berlin Heidelberg, Galway, Ireland, 2005. doi: 10.1007/11574620_21. URL http://link.springer.com/chapter/10.1007/11574620_{_}21. [Cited at pages 29 and 50]
- Aldo Gangemi and Valentina Presutti. Ontology Design Patterns. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, pages 221–243. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-70999-2. doi: 10.1007/978-3-540-92673-3_10. URL <http://www.springerlink.com/index/10.1007/978-3-540-92673-3>. [Cited at page 29]
- Stephen Gorard. Revisiting a 90-year-old Debate: The Advantages of the Mean Deviation. *British Journal of Educational Studies*, 53(4):417–430, dec 2005. doi: 10.1111/j.1467-8527.2005.00304.x. URL <http://www.tandfonline.com/doi/abs/10.1111/j.1467-8527.2005.00304.x>. [Cited at page 92]
- Baris Guc, Michael May, Yucel Saygin, and Christine Körner. Semantic annotation of GPS trajectories. *11th AGILE International Conference on Geographic Information Science 2008*, 38(6):1–9, 2008. doi: 10.1016/j.combiomed.2008.02.004. [Cited at pages 33 and 37]
- Raf Guns. Tracing the origins of the semantic web. *Journal of the American Society for Information Science and Technology*, 64(10):2173–2181, oct

2013. ISSN 15322882. doi: 10.1002/asi.22907. URL <http://doi.wiley.com/10.1002/asi.22907>. [Cited at page 13]
- Torsten Hägerstrand. What about people in Regional Science?, 1970. ISSN 10568190. [Cited at page 35]
- Tom Heath and Christian Bizer. Linked Data: Evolving the Web into a Global Data Space. *Synthesis Lectures on the Semantic Web: Theory and Technology*, 1:1–136, feb 2011. ISSN 2160-4711. doi: 10.2200/S00334ED1V01Y201102WBE001. URL <http://www.morganclaypool.com/doi/abs/10.2200/S00334ED1V01Y201102WBE001>. [Cited at pages 14 and 16]
- Maarten Hilferink. Fisher’s Natural Breaks Classification, 2013. URL http://wiki.objectvision.nl/index.php/Fisher's_Natural_Breaks_Classification. [Cited at page 108]
- Jerry R. Hobbs and Feng Pan. An ontology of time for the semantic web. *ACM Transactions on Asian Language Information Processing*, 3(1):66–85, mar 2004. ISSN 15300226. doi: 10.1145/1017068.1017073. URL <http://portal.acm.org/citation.cfm?doid=1017068.1017073>. [Cited at page 23]
- Kathleen Hornsby and Max J. Egenhofer. Modeling moving objects over multiple granularities. *Annals of Mathematics and Artificial Intelligence*, 36(1-2):177–194, 2002. URL <http://link.springer.com/article/10.1023/A:1015812206586>. [Cited at page 77]
- Yingjie Hu, Krzysztof Janowicz, David Carral, Simon Scheider, Werner Kuhn, Gary Berg-Cross, Pascal Hitzler, Mike Dean, and Dave Kolas. A Geontology Design Pattern for Semantic Trajectories. In *11th International Conference, COSIT 2013*, pages 438–456, Scarborough, UK, 2013. Springer

- International Publishing. doi: 10.1007/978-3-319-01790-7_24. URL <http://link.springer.com/10.1007/978-3-319-01790-7>. [Cited at pages ix, 4, 30, 32, 59, and 76]
- Peter J. Huber and Elvezio M. Ronchetti. *Robust Statistics*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, USA, jan 2009. ISBN 9780470434697. doi: 10.1002/9780470434697. URL <http://doi.wiley.com/10.1002/9780470434697>. [Cited at page 91]
- Boris Iglewicz and David C. Hoaglin. Volume 16: How to Detect and Handle Outliers. In Edward F. Mykytka, editor, *The ASQC Basic References in Quality Control: Statistical Techniques*, page 87. ASQC/Quality Press, 1993. ISBN 087389247X. [Cited at pages 91, 92, and 97]
- Sergio Ilarri, Dragan Stojanovic, and Cyril Ray. Semantic management of moving objects: A vision towards smart mobility. *Expert Systems with Applications*, 42(3):1418–1435, 2015. ISSN 09574174. doi: 10.1016/j.eswa.2014.08.057. URL <http://linkinghub.elsevier.com/retrieve/pii/S0957417414005399>. [Cited at pages 53 and 77]
- ISO. Data Elements and Interchange Formats: Information Interchange: Erepresentation of Dates and Times, 1988. [Cited at page 22]
- Krzysztof Janowicz, Simon Scheider, Todd Pehle, and Glen Hart. Geospatial semantics and linked spatiotemporal data - Past, present, and future. *Semantic Web*, 3(4):1–13, 2012. URL <http://iospress.metapress.com/index/M065W1130043W3P4.pdf>. [Cited at page 19]
- George F. Jenks. Optimal data classification for choropleth maps, 1977. [Cited at page 103]
- George F. Jenks and Fred C. Caspall. Error on Choroplethic Maps: Definition, Measurement, Reduction. *Annals of the Association of American Geograp-*

- hers*, 61(2):217–244, 1971. ISSN 14678306. doi: 10.1111/j.1467-8306.1971.tb00779.x. [Cited at page 103]
- Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowledge and Information Systems*, 3(3):263–286, 2001a. ISSN 0219-1377. doi: 10.1007/PL00011669. URL http://research.microsoft.com/pubs/79074/time_{ }series_{ }indexing.pdf. [Cited at page 42]
- Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. An online algorithm for segmenting time series. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 289–296. IEEE Comput. Soc, 2001b. ISBN 0-7695-1119-8. doi: 10.1109/ICDM.2001.989531. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=989531>. [Cited at page 43]
- Werner Kuhn, Tomi Kauppinen, and Krzysztof Janowicz. Linked Data - A Paradigm Shift for Geographic Information Science. In *8th International Conference on Geographic Information Science (GIScience)*, Vienna, Austria, 2014. Springer International Publishing. doi: 10.1007/978-3-319-11593-1_12. [Cited at page 19]
- Patrick Laube. Computational Movement Analysis. In *SpringerBriefs in Computer Science*. Springer International Publishing, Berlin Heidelberg, 1 edition, 2014. ISBN 978-3-319-10268-9. doi: 10.1007/978-3-319-10268-9. [Cited at pages 2 and 39]
- Patrick Laube. The Low Hanging Fruit is Gone - Achievements and Challenges of Computational Movement Analysis. *SIGSPATIAL Special*, 7(1):3–10, 2015. ISSN 19467729. doi: 10.1145/2782759.2782762. URL <http://dl.acm.org/citation.cfm?doid=2782759.2782762>. [Cited at pages 6 and 52]

- Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764–766, 2013. ISSN 00221031. doi: 10.1016/j.jesp.2013.03.013. URL <http://dx.doi.org/10.1016/j.jesp.2013.03.013>. [Cited at page 92]
- Joshua Lieberman, Raj Singh, and Chris Goad. W3C Geospatial Vocabulary, 2007. URL <https://www.w3.org/2005/Incubator/geo/XGR-geo-20071023>. [Cited at page 19]
- Gerasimos Marketos, Maria Luisa Damiani, Nikos Pelekis, Yannis Theodoridis, and Zhixian Yan. Trajectory Collection and Reconstruction. In Chiara Renso, Stefano Spaccapietra, and Esteban Zimányi, editors, *Mobility Data: Modeling, Management, and Understanding*, pages 23–41. Cambridge University Press, 2013. ISBN 9781139128926. [Cited at pages ix and 37]
- Harvey J. Miller. A Measurement Theory for Time Geography. *Geographical Analysis*, 37(1):17–45, 2005. ISSN 00167363. doi: 10.1111/j.1538-4632.2005.00575.x. [Cited at page 35]
- David Mountain and Jonathan Raper. Modelling human spatio-temporal behaviour: A challenge for location-based services. *Proceedings of the Sixth International Conference on GeoComputation, University of Queensland, Brisbane, Australia (2001)*, pages 1–9, 2001. [Cited at pages 7, 32, and 37]
- Jerome L. Myers and Arnold D. Well. *Research Design and Statistical Analysis*. Lawrence Erlbaum Associates, Mahwah, New Jersey, 2 edition, 2003. ISBN 1-4106-0703-8. [Cited at page 89]
- Theodor H. Nelson. A File Structure for the Complex, the Changing, and the Indeterminate. In *ACM 20th National Conference*, pages 84–100, 1965.

- ISBN 0262232278 9780262232272. doi: 10.1145/800197.806036. [Cited at page 12]
- NIST/SEMATECH. NIST/SEMATECH e-Handbook of Statistical Methods, 2012. URL <http://www.itl.nist.gov/div898/handbook/>. [Cited at page 92]
- Tales Paiva Nogueira and Hervé Martin. Qualitative Representation of Dynamic Attributes of Trajectories. *Proceedings of the AGILE'2014 International Conference on Geographic Information Science*, pages 3–6, 2014. [Cited at pages 54 and 103]
- Tales Paiva Nogueira and Hervé Martin. Querying semantic trajectory episodes. In *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems - MobiGIS '15*, pages 23–30, Seattle, WA, USA, 2015. ACM Press. ISBN 9781450339773. doi: 10.1145/2834126.2834136. URL <http://dl.acm.org/citation.cfm?doid=2834126.2834136>. [Cited at pages ix, 57, and 58]
- Tales Paiva Nogueira, Reinaldo Bezerra Braga, and Hervé Martin. An Ontology-based Approach to Represent Trajectory Characteristics. *Computing for Geospatial Research and Application (COM.Geo), 2014 Fifth International Conference on*, pages 102–107, 2014. doi: 10.1109/COM.Geo.2014.22. [Cited at pages 32, 54, 62, 85, 101, and 103]
- Benjamin Nowack. The Semantic Web - Not a piece of cake..., 2009. URL <http://bnode.org/blog/2009/07/08/the-semantic-web-not-a-piece-of-cake>. [Cited at pages ix and 15]
- OGC. An introduction to georss: A standards based approach for geo-enabling rss feeds. *White Paper OGC*, 2006. [Cited at page 19]

Daniel Orellana and Chiara Renso. Developing an Ontology of Interactions for characterizing Pedestrian Movement Behaviour. In Monica Wachowicz, editor, *Movement-Aware Applications for Sustainable Mobility: Technologies and Approaches*, chapter 5, pages 62–86. IGI Global, New York, 2010. doi: 10.4018/978-1-61520-769-5.ch005. [Cited at page 28]

Andrey Tietbohl Palma, Vania Bogorny, Bart Kuijpers, and Luis Otavio Alvares. A clustering-based approach for discovering interesting places in trajectories. In *Proceedings of the 2008 ACM Symposium on Applied Computing - SAC '08*, New York, New York, USA, 2008. ACM Press. ISBN 9781595937537. doi: 10.1145/1363686.1363886. URL <http://dl.acm.org/citation.cfm?id=1363886><http://portal.acm.org/citation.cfm?doid=1363686.1363886>. [Cited at pages 40, 85, and 101]

Spiros Papadimitriou, Jimeng Sun, Christos Faloutsos, and Philip S. Yu. Dimensionality reduction and filtering on time series sensor streams. In Charu C. Aggarwal, editor, *Managing and Mining Sensor Data*, volume 9781461463, pages 103–141. Springer US, 2013. ISBN 9781461463092. doi: 10.1007/978-1-4614-6309-2_5. [Cited at page 42]

Christine Parent, Stefano Spaccapietra, and Esteban Zimányi. *Conceptual Modeling for Traditional and Spatio-Temporal Applications: The MADS Approach*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1 edition, 2006. ISBN 9783540301530. [Cited at page 27]

Christine Parent, Nikos Pelekis, Yannis Theodoridis, Zhixian Yan, Stefano Spaccapietra, Chiara Renso, Gennady Andrienko, Natalia Andrienko, Vania Bogorny, Maria Luisa Damiani, Aris Gkoulalas-Divanis, and Jose Antonio Fernandes de Macedo. Semantic trajectories modeling and analysis. *ACM Computing Surveys*, 45(4):1–32, aug 2013. ISSN 03600300.

- doi: 10.1145/2501654.2501656. URL <http://dl.acm.org/citation.cfm?doid=2501654.2501656>. [Cited at pages 4, 7, and 37]
- Kostas Patroumpas, Giorgos Giannopoulos, and Spiros Athanasiou. Towards GeoSpatial Semantic Data Management: Strengths, Weaknesses, and Challenges Ahead. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - SIGSPATIAL '14*, pages 301–310, 2014. ISBN 9781450331319. doi: 10.1145/2666310.2666410. URL <http://dl.acm.org/citation.cfm?doid=2666310.2666410>. [Cited at pages 7 and 132]
- Nikos Pelekis and Yannis Theodoridis. *Mobility Data Management and Exploration*. Springer New York, New York, NY, 2014. ISBN 978-1-4939-0391-7. doi: 10.1007/978-1-4939-0392-4. URL <http://link.springer.com/10.1007/978-1-4939-0392-4>. [Cited at page 7]
- Michalis Potamias, Kostas Patroumpas, and Timos Sellis. Sampling Trajectory Streams with Spatiotemporal Criteria. In *18th International Conference on Scientific and Statistical Database Management (SSDBM'06)*, pages 275–284. IEEE, 2006. ISBN 0-7695-2590-3. doi: 10.1109/SSDBM.2006.45. URL <http://ieeexplore.ieee.org/document/1644324/>. [Cited at page 38]
- Azzurra Ragone, Tommaso Di Noia, Vito Walter Anelli, Andrea Cali, and Matteo Palmonari. Exposing Open Street Map in the Linked Data cloud. In *Proceedings of the 29th International Conference on Industrial, Engineering & other Applications of Applied Intelligent Systems*, 2016. URL <http://www-ictserv.poliba.it/sisinflab/publications/2016/RDACP16>. [Cited at page 120]
- Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244–256, nov 1972.

- ISSN 0146664X. doi: 10.1016/S0146-664X(72)80017-0. URL <http://www.sciencedirect.com/science/article/pii/S0146664X72800170>. [Cited at pages 38 and 59]
- David A. Randell, Zhan Cui, and Anthony G. Cohn. A Spatial Logic based on Regions and Connection. *3rd International Conference On Knowledge Representation And Reasoning*, pages 165–176, 1992. ISSN 1098-6596. doi: 10.1.1.35.7809. [Cited at page 22]
- Chiara Renso, Miriam Baglioni, Jose Antonio Fernandes de Macedo, Roberto Trasarti, and Monica Wachowicz. How you move reveals who you are: understanding human behavior by analyzing trajectory data. *Knowledge and Information Systems*, 37(2):331–362, nov 2013. ISSN 0219-1377. doi: 10.1007/s10115-012-0511-z. URL <http://link.springer.com/10.1007/s10115-012-0511-z>. [Cited at page 26]
- Kai-Florian Richter, Falko Schmid, and Patrick Laube. Semantic trajectory compression: Representing urban movement in a nutshell. *Journal of Spatial Information Science*, 4(4):3–30, jun 2012. ISSN 1948-660X. doi: 10.5311/JOSIS.2012.4.62. URL <http://josis.org/index.php/josis/article/view/62>. [Cited at page 38]
- S. W. Roberts. Control Chart Tests Based on Geometric Moving Averages. *Technometrics*, 1(3):239–250, 1959. ISSN 0040-1706. doi: 10.1080/00401706.1959.10489860. [Cited at page 113]
- Jose Antonio M. R. Rocha, Valéria C. Times, Gabriel Oliveira, Luis Otavio Alvares, and Vania Bogorny. DB-SMoT: A direction-based spatio-temporal clustering method. In *5th IEEE International Conference Intelligent Systems*, pages 114–119. IEEE, jul 2010. ISBN 978-1-4244-5163-0. doi: 10.1109/IS.2010.5548396. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5548396>. [Cited at pages 85 and 101]

- Peter J. Rousseeuw and Mia Hubert. Robust statistics for outlier detection. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):73–79, 2011. ISSN 19424787. doi: 10.1002/widm.2. [Cited at page 92]
- Max Schmachtenberg, Christian Bizer, Anja Jentzsch, and Richard Cyganiak. Linking Open Data cloud diagram, 2014. URL <http://lod-cloud.net/>. [Cited at pages ix and 20]
- Falko Schmid, Kai-Florian Richter, and Patrick Laube. Semantic trajectory compression. In Nikos Mamoulis, Thomas Seidl, Torben Bach Pedersen, Kristian Torp, and Ira Assent, editors, *Advances in Spatial and Temporal Databases*, pages 411–416. Springer Berlin Heidelberg, 2009. doi: 10.1007/978-3-642-02982-0_30. URL http://link.springer.com/chapter/10.1007/978-3-642-02982-0_{_}30. [Cited at page 38]
- Hagit Shatkay and Stanley B. Zdonik. Approximate queries and representations for large data sequences. In *Proceedings of the Twelfth International Conference on Data Engineering*, pages 536–545. IEEE Comput. Soc. Press, 1996. ISBN 0-8186-7240-4. doi: 10.1109/ICDE.1996.492204. URL <http://ieeexplore.ieee.org/document/492204/>. [Cited at page 43]
- Ricardo Almeida Silva, João Moura Pires, Maribel Yasmina Santos, and Rui Leal. Aggregating spatio-temporal phenomena at multiple levels of detail. *Lecture Notes in Geoinformation and Cartography*, 217:291–308, 2015. ISSN 18632351. doi: 10.1007/978-3-319-16787-9_17. [Cited at page 77]
- Elena Simperl, Barry Norton, Maribel Acosta, Maria Maleshkova, John Domingue, Alexander Mikroyannidis, Paul Mulholland, and Richard Power. *Using Linked Data Effectively*. The Open University, Milton Keynes, 2013. URL <http://oro.open.ac.uk/40052/>. [Cited at page 16]

Terry Slocum, Robert McMaster, Frit Kessler, and Hugh Howard. *Thematic cartography and geovisualization*. Pearson Prentice Hall, Upper Saddle River, NJ, 3 edition, 2009. [Cited at page 103]

Stefano Spaccapietra, Christine Parent, Maria Luisa Damiani, Jose Antonio Fernandes de Macedo, Fabio Porto, and Christelle Vangenot. A conceptual view on trajectories. *Data & Knowledge Engineering*, 65(1):126–146, apr 2008. ISSN 0169023X. doi: 10.1016/j.datak.2007.10.008. URL <http://linkinghub.elsevier.com/retrieve/pii/S0169023X07002078>. [Cited at pages ix, 4, 24, 25, 26, 33, 76, and 79]

Stefano Spaccapietra, Christine Parent, and Laura Spinsanti. Trajectories and Their Representations. In Chiara Renso, Stefano Spaccapietra, and Esteban Zimányi, editors, *Mobility Data: Modeling, Management, and Understanding*, pages 3–22. Cambridge University Press, 2013. ISBN 9781139128926. [Cited at pages 36 and 37]

Claus Stadler, Jens Lehmann, Konrad Höffner, and Sören Auer. LinkedGeoData: A Core for a Web of Spatial Open Data. *Semantic Web*, 3(4):333–354, 2012. doi: 10.3233/SW-2011-0052. URL <http://iospress.metapress.com/content/141W054666871326>. [Cited at page 119]

Willem Robert van Hage, Véronique Malaisé, Gerben de Vries, Guus Schreiber, and Maarten van Someren. Combining ship trajectories and semantics with the simple event model (SEM). In *Proceedings of the 1st ACM international workshop on Events in multimedia - EiMM '09*, page 73, New York, New York, USA, 2009a. ACM Press. ISBN 9781605587547. doi: 10.1145/1631024.1631039. URL <http://portal.acm.org/citation.cfm?doid=1631024.1631039>. [Cited at page 4]

Willem Robert van Hage, Véronique Malaisé, Roxane Segers, Laura Hollink,

- and Guus Schreiber. The Simple Event Model Ontology, 2009b. URL <http://semanticweb.cs.vu.nl/2009/11/sem/>. [Cited at pages ix and 29]
- Willem Robert van Hage, Véronique Malaisé, Roxane Segers, Laura Holink, and Guus Schreiber. Design and use of the Simple Event Model (SEM). *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(2):128–136, jul 2011. ISSN 15708268. doi: 10.1016/j.websem.2011.03.003. URL <http://linkinghub.elsevier.com/retrieve/pii/S1570826811000199>. [Cited at page 29]
- Willem Robert van Hage, Véronique Malaisé, Gerben K. D. de Vries, Guus Schreiber, and Maarten Van Someren. Abstracting and reasoning over ship trajectories and web data with the Simple Event Model (SEM). *Multimedia Tools and Applications*, 57(1):175–197, mar 2012. ISSN 1380-7501. doi: 10.1007/s11042-010-0680-2. URL <http://link.springer.com/10.1007/s11042-010-0680-2>. [Cited at pages 29 and 76]
- Arnaud Vandecasteele, Rodolphe Devillers, and Aldo Napoli. From Movement Data to Objects Behavior Using Semantic Trajectory and Semantic Events. *Marine Geodesy*, 37(2):126–144, may 2014. ISSN 0149-0419. doi: 10.1080/01490419.2014.902885. URL <http://www.tandfonline.com/doi/abs/10.1080/01490419.2014.902885>. [Cited at page 30]
- Pierre-Yves Vandenbussche, Ghislain Ateazing, María Poveda-Villalón, and Bernard Vatant. Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web. *Semantic Web*, Preprint:1–16, 2016. doi: 10.3233/SW-160213. URL <http://content.iospress.com/articles/semantic-web/sw213>. [Cited at pages 18 and 81]
- W3C. W3C Semantic Web Interest Group: Basic Geo (WGS84 lat/long) Vocabulary, 2003. URL <https://www.w3.org/2003/01/geo/>. [Cited at page 19]

- W3C. OWL Web Ontology Language Reference, 2004. URL <https://www.w3.org/TR/owl-ref/>. [Cited at page 67]
- W3C. Ontologies - W3C, 2015. URL <https://www.w3.org/standards/semanticweb/ontology.html>. [Cited at page 18]
- Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, mar 2013. ISSN 1384-5810. doi: 10.1007/s10618-012-0250-5. URL <http://link.springer.com/10.1007/s10618-012-0250-5>. [Cited at page 42]
- Rouaa Wannous, Jamal Malki, Alain Bouju, and Cécile Vincent. Modeling Mobile Object Activities Based on Trajectory Ontology Rules Considering Spatial Relationship Rules. In Abdelmalek Amine, Ait Mohamed Otmane, and Ladjel Bellatreche, editors, *Modeling Approaches and Algorithms for Advanced Computer Applications*, pages 249–258. Springer International Publishing, 2013. doi: 10.1007/978-3-319-00560-7_29. URL http://link.springer.com/chapter/10.1007/978-3-319-00560-7_{_}29. [Cited at page 4]
- Zhixian Yan. *Semantic Trajectories: Computing and Understanding Mobility Data*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2011. [Cited at page 37]
- Zhixian Yan, Jose Antonio Fernandes de Macedo, Christine Parent, and Stefano Spaccapietra. Trajectory Ontologies and Queries. *Transactions in GIS*, 12(Supplement s1):75–91, dec 2008. ISSN 13611682. doi: 10.1111/j.1467-9671.2008.01137.x. URL <http://doi.wiley.com/10.1111/j.1467-9671.2008.01137.x>. [Cited at pages ix, 4, 26, 27, 31, 76, 78, and 79]

Zhixian Yan, Christine Parent, Stefano Spaccapietra, and Dipanjan Chakraborty. A hybrid model and computing platform for spatio-semantic trajectories. In *7th Extended Semantic Web Conference, ESWC 2010*, pages 60–75, Heraklion, Crete, Greece, 2010. Springer Berlin Heidelberg. ISBN 3642134858. doi: 10.1007/978-3-642-13486-9_5. URL http://link.springer.com/chapter/10.1007/978-3-642-13486-9_5. [Cited at pages 40, 85, and 101]

Zhixian Yan, Dipanjan Chakraborty, Christine Parent, Stefano Spaccapietra, and Karl Aberer. SeMiTri: A Framework for Semantic Annotation of Heterogeneous Trajectories. In *Proceedings of the 14th International Conference on Extending Database Technology - EDBT/ICDT '11*, page 259, New York, New York, USA, 2011. ACM Press. ISBN 9781450305280. doi: 10.1145/1951365.1951398. URL <http://portal.acm.org/citation.cfm?doid=1951365.1951398>. [Cited at pages 4, 84, and 117]

Zhixian Yan, Dipanjan Chakraborty, Christine Parent, Stefano Spaccapietra, and Karl Aberer. Semantic Trajectories: Mobility Data Computation and Annotation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(2):39:1—39:38, 2012. doi: 10.1145/2483669.2483682. [Cited at pages ix, 37, 38, 45, and 46]

Juan Ye, Stamatia Dasiopoulou, Graeme Stevenson, Georgios Meditskos, Efstratios Kontopoulos, Ioannis Kompatsiaris, and Simon Dobson. Semantic web technologies in pervasive computing: A survey and research roadmap. *Pervasive and Mobile Computing*, 2015. ISSN 15741192. doi: 10.1016/j.pmcj.2014.12.009. URL <http://dx.doi.org/10.1016/j.pmcj.2014.12.009>. [Cited at page 3]

Byoung-Kee Yi and Christos Faloutsos. Fast Time Sequence Indexing for Arbitrary Lp norms. *Proc. of the 26th Int. Conf. on VLDB'00*, pages 385–

394, 2000. ISSN 1553-3514. doi: 10.1016/j.cppeds.2011.05.001. [Cited at page 42]

Yu Zheng. Trajectory Data Mining: An Overview. *ACM Trans. On Intelligent Systems and Technology*, 6(3), 2015. doi: 10.1145/2743025. [Cited at pages 2, 37, and 39]

Nina Zumel and John Mount. *Practical data science with R*. Manning Publications Co., Shelter Island, NY, 2014. ISBN 9781617291562. [Cited at page 98]

Appendix

Appendix A: Demonstration of the Stop-Move-Noise (MSN) classification method

Import packages and general settings:

In [1]:

```
import os
import sys

module_path = os.path.abspath(os.path.join('../'))
if module_path not in sys.path:
    sys.path.append(module_path)

from step import preprocessing as pp
from step import stats as st
from step import util, msn

import math
import numpy as np
from scipy import stats

import gpxpy

import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns

%load_ext autoreload
%autoreload 2
%matplotlib notebook

matplotlib.rcParams['axes.labelsize'] = '13'
matplotlib.rcParams['ytick.labelsize'] = '13'
matplotlib.rcParams['xtick.labelsize'] = '13'
matplotlib.rcParams['axes.titlesize'] = '13'
matplotlib.rcParams['legend.edgecolor'] = 'k'
matplotlib.rcParams['legend.shadow'] = True
matplotlib.rcParams['legend.frameon'] = True
matplotlib.rcParams['figure.figsize'] = (7, 5)
sns.set_style('ticks', {"axes.xmargin": 0.2, "axes.ymargin": 0.2});

# For pretty printing
import warnings
warnings.simplefilter('ignore')
```

Load one trajectory file in GPX format:

In [2]:

```
file_path = r'gpx/27031614-1583713025.gpx'
#file_path = r'gpx/49996348-1533217229.gpx'
gpx = gpxpy.parse(open(file_path, 'r'))
```

Compute movement attributes:

In [3]:

```
df = pp.compute_attributes(gpx)[1:]
```

```
# first 5 rows:  
df.head()
```

Out[3]:

	latitude	longitude	distance	duration	speed	acceleration	heading	angle	elevation	timestamp	idx
2016-07-04 16:09:21	45.189722	5.766735	10.16	5	2.03	0.04	303	177.69	216	2016-07-04 16:09:21	1
2016-07-04 16:09:26	45.189773	5.766613	11.09	5	2.22	-0.02	301	177.45	215	2016-07-04 16:09:26	2
2016-07-04 16:09:31	45.189825	5.766500	10.56	5	2.11	-0.01	303	174.10	215	2016-07-04 16:09:31	3
2016-07-04 16:09:36	45.189868	5.766382	10.39	5	2.08	-0.03	297	172.89	215	2016-07-04 16:09:36	4
2016-07-04 16:09:42	45.189903	5.766245	11.42	6	1.90	-0.05	290	168.08	215	2016-07-04 16:09:42	5

Plot the trajectory:

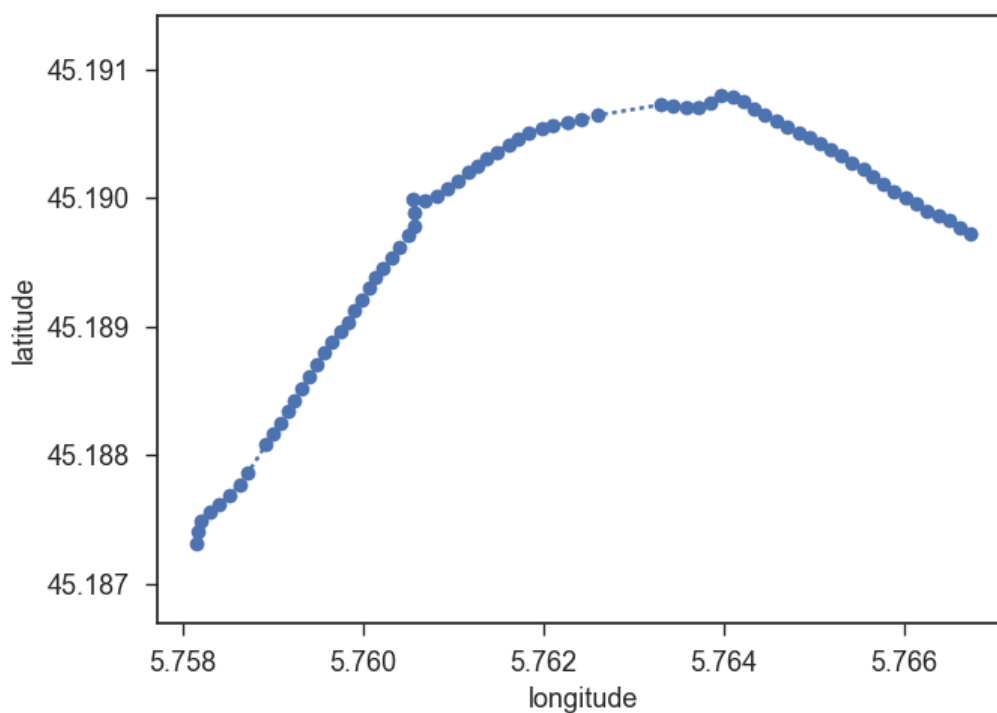
In [4]:

```
fig = plt.figure()  
ax = fig.add_subplot(111)
```

```
xx = df.longitude  
yy = df.latitude
```

```
plt.xlabel('longitude')  
plt.ylabel('latitude')  
plt.plot(df.longitude, df.latitude, 'o:', ms=7)
```

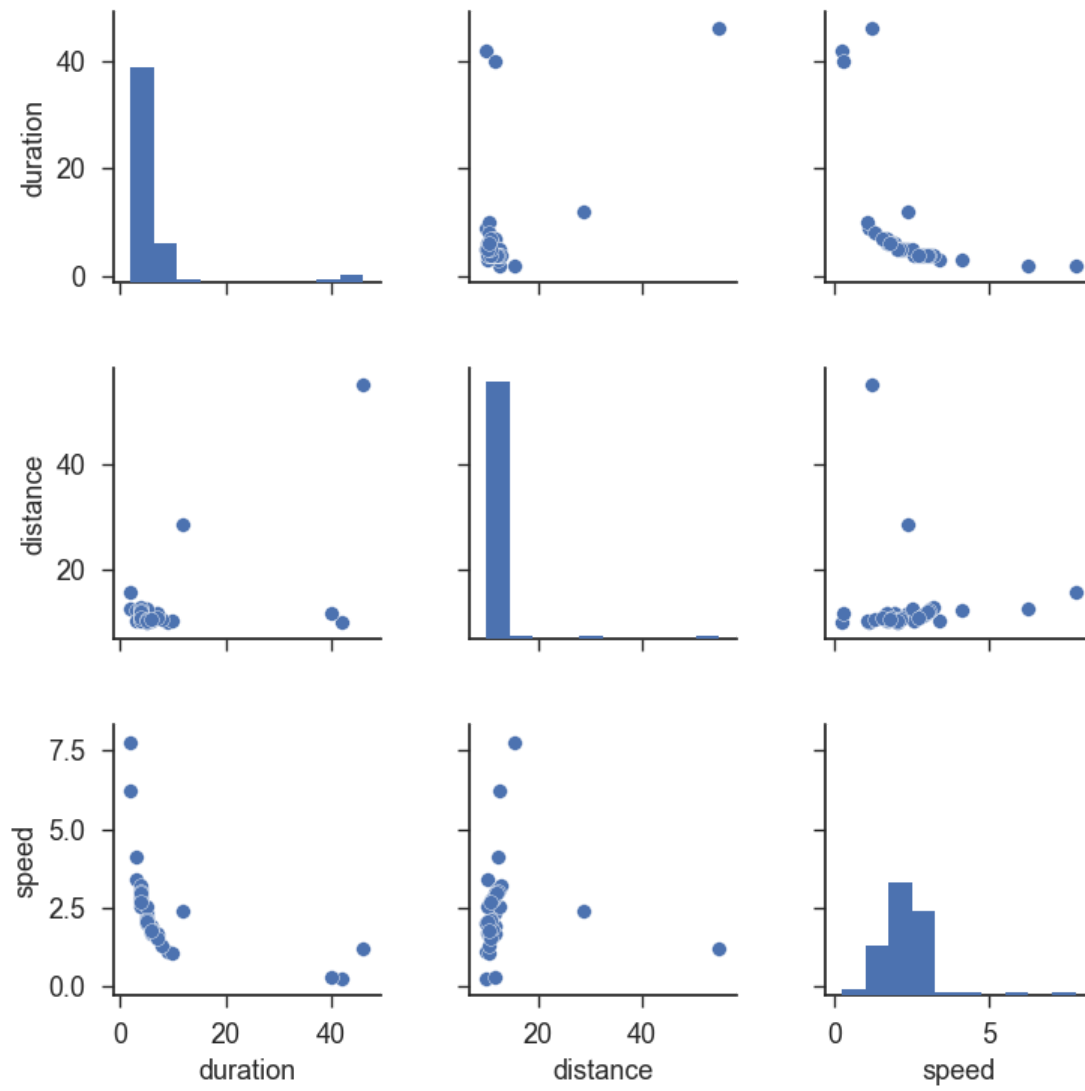
```
central_latitude = sum(plt.axes().get_ylim())/2.  
mercator_aspect_ratio = 1/math.cos(math.radians(central_latitude))  
plt.axes().set_aspect(mercator_aspect_ratio, adjustable='datalim')  
plt.tight_layout()
```



Relationship among distance, duration, and speed between each pair of sequential points:

In [5]:

```
sns.pairplot(df[['duration', 'distance', 'speed']]);  
plt.tight_layout()
```

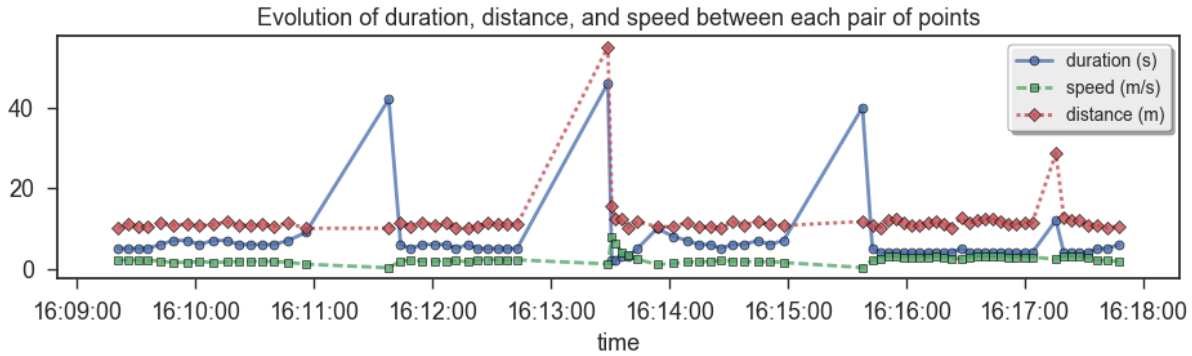


In [6]:

```
fig = plt.figure(figsize=(9.5,3))
plt.title('Evolution of duration, distance, and speed between each pair of points')
plt.plot(df['duration'], '-o', label='duration (s)', lw=2, ms=5, mew=.5, mec='k', alpha=.8)
plt.plot(df['speed'], '--s', label='speed (m/s)', lw=2, ms=5, mew=.5, mec='k', alpha=.8)
plt.plot(df['distance'], ':D', label='distance (m)', lw=2, ms=5, mew=.5, mec='k', alpha=.8)

plt.gca().xaxis.zoom(-0.1)
plt.xlabel("time")

plt.legend(loc='best', frameon=True);
plt.gca().xaxis.set_major_formatter(matplotlib.dates.DateFormatter("%H:%M:%S"));
plt.tight_layout()
```



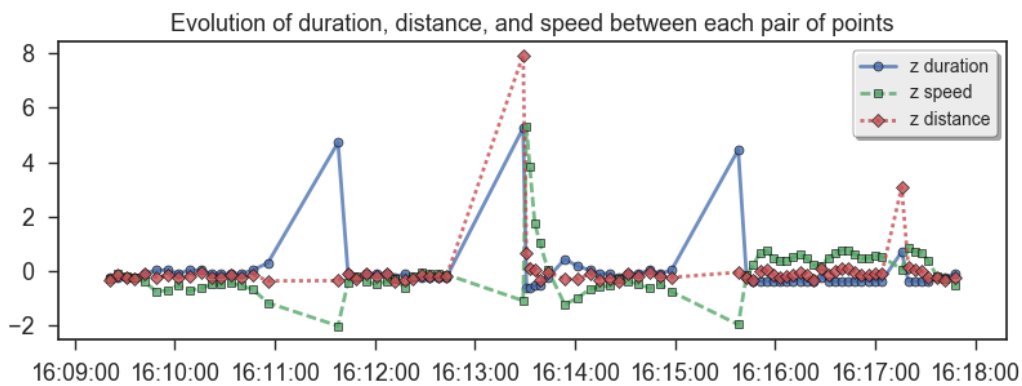
In [7]:

```
df['z_duration'] = stats.zscore(df['duration'])
df['z_speed'] = stats.zscore(df['speed'])
df['z_distance'] = stats.zscore(df['distance'])

fig = plt.figure(figsize=(9.5,3))
plt.title('Evolution of duration, distance, and speed between each pair of points')
plt.plot(df['z_duration'], '-o', label='z duration', lw=2, ms=5, mew=.5, mec='k', alpha=.8)
plt.plot(df['z_speed'], '--s', label='z speed', lw=2, ms=5, mew=.5, mec='k', alpha=.8)
plt.plot(df['z_distance'], ':D', label='z distance', lw=2, ms=5, mew=.5, mec='k', alpha=.8)

plt.gca().xaxis.zoom(-0.1)
plt.xlabel("time")

plt.legend(loc='best', frameon=True);
plt.gca().xaxis.set_major_formatter(matplotlib.dates.DateFormatter("%H:%M:%S"));
#plt.tight_layout()
```



In [8]:

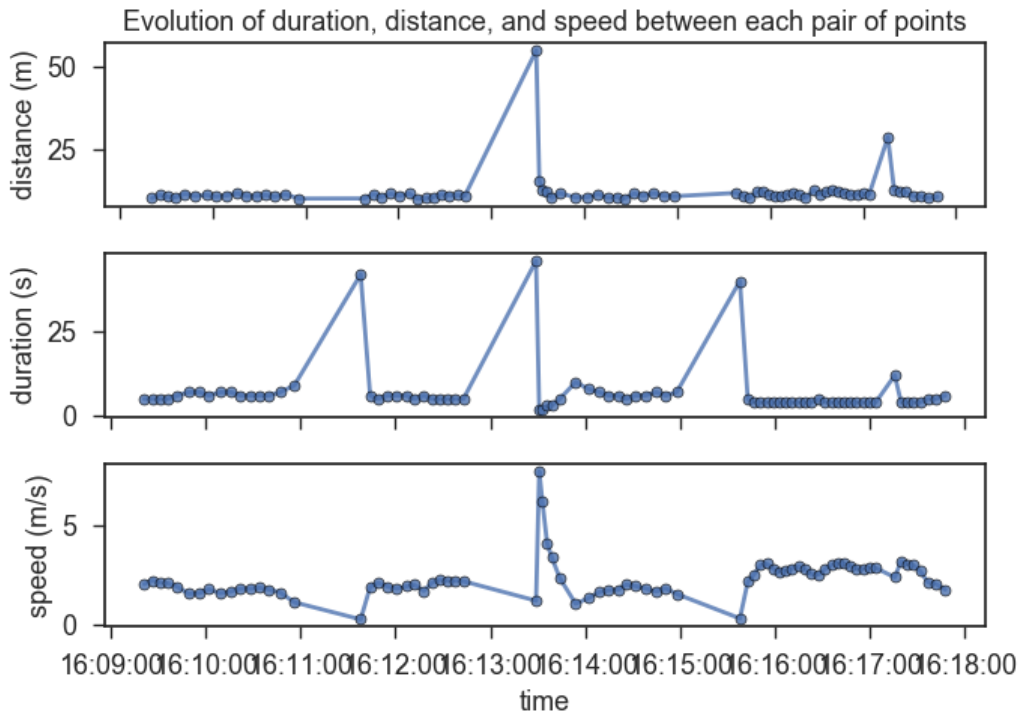
```
fig, (ax1, ax2, ax3) = plt.subplots(nrows=3, sharex=True)
ax1.set_title('Evolution of duration, distance, and speed between each pair of points')
ax1.plot(df['distance'], '-o', label='distance (m)', lw=2, ms=5, mew=.5, mec='k', alpha=.8)
ax1.set_ylabel('distance (m)')

ax2.plot(df['duration'], '-o', label='duration (s)', lw=2, ms=5, mew=.5, mec='k', alpha=.8)
ax2.set_ylabel('duration (s)')

ax3.plot(df['speed'], '-o', label='speed (m/s)', lw=2, ms=5, mew=.5, mec='k', alpha=.8)
ax3.set_ylabel('speed (m/s)')

plt.gca().xaxis.zoom(-0.1)
plt.xlabel("time")

plt.gca().xaxis.set_major_formatter(matplotlib.dates.DateFormatter("%H:%M:%S"));
plt.tight_layout()
```



Parameters

- `distance_threshold`, `duration_threshold`, `speed_threshold`: value of absolute modified z-score at which higher values are considered as outliers
- `minimum_angle`: minimum angle in degrees
- `max_sequential_sharp_angles`: maximum number of sequential angles that can be below `minimum_angle`
- `jitter`: amount of random noise that can be added to each duration

In [9]:

```
distance_threshold = 3.5
duration_threshold = 5.0
speed_threshold = 3.5

minimum_angle = 45
max_sequential_sharp_angles = 1

jitter = 0.01
```

Noise labeling

Analysis of distance between points

Statistics about distance:

In [10]:

```
df['distance'].describe()
```

Out[10]:

```
count    76.000000
mean     11.964737
std       5.449888
min       9.980000
25%      10.577500
50%      11.090000
75%      11.620000
max       54.940000
Name: distance, dtype: float64
```

Modified z-score computation:

In [11]:

```
mz_distance = st.modified_zscore(df['distance'])
long_distance_indexes = [i for i, mz in enumerate(mz_distance)
                        if mz > distance_threshold]
```

Abnormal distance(s):

In [12]:

```
df.iloc[long_distance_indexes]
```

Out[12]:

	latitude	longitude	distance	duration	speed	acceleration	heading	angle	elevation	timestamp	idx	z_dura
2016-07-04 16:13:29	45.190644	5.762608	54.94	46	1.19	3.29	261	174.44	215	2016-07-04 16:13:29	30	5.2549
2016-07-04 16:13:31	45.190608	5.762417	15.55	2	7.77	-0.77	255	177.92	213	2016-07-04 16:13:31	31	-0.6339
2016-07-04 16:17:16	45.187866	5.758729	28.62	12	2.38	0.20	213	177.05	216	2016-07-04 16:17:16	69	0.7044

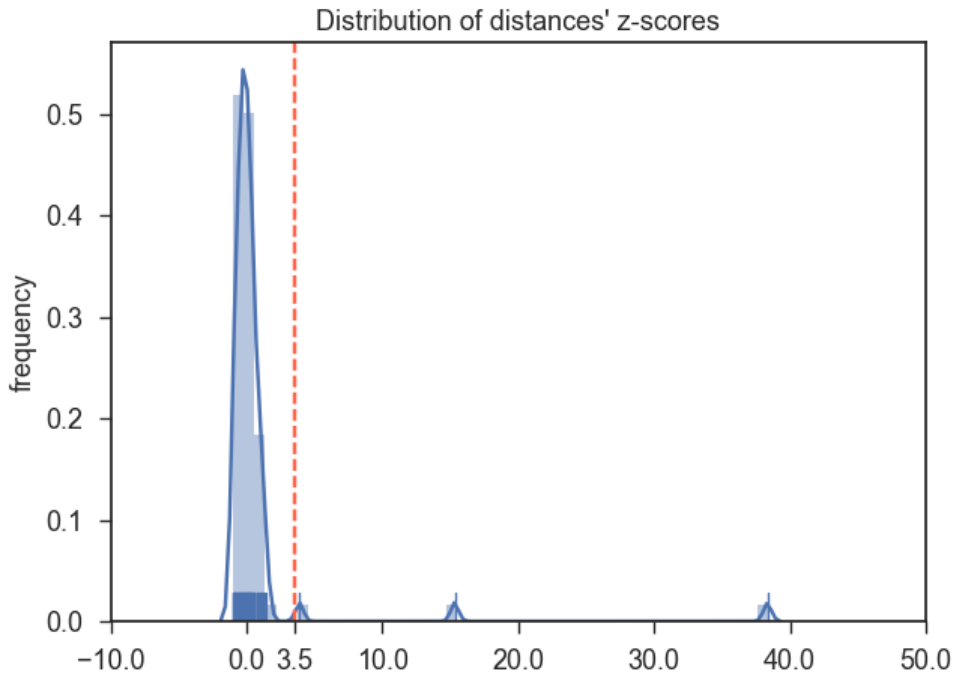
Distribution of modified-zscore of distances:

In [13]:

```
plt.figure()
plt.title("Distribution of distances' z-scores")
plt.ylabel('frequency')

sns.distplot(mz_distance.as_matrix(), rug=True)

plt.xticks(list(plt.xticks()[0]) + [distance_threshold])
plt.axvline(distance_threshold, color='tomato', linestyle='--');
```



Analysis of angles between points

In [14]:

```
sharp_angles = df[df['angle'] < minimum_angle][:-1]
sharp_angles_index = [df.index.get_loc(i)
                      for i in sharp_angles.index]

diff = np.ediff1d(sharp_angles_index, np.inf)
mask = [True if d <= max_sequential_sharp_angles + 1
        else False for d in diff]

outliers = []
for i, m in enumerate(mask):
    if m:
        outliers.append(sharp_angles_index[i])
        outliers.append(sharp_angles_index[i+1])

sharp_angle_intervals = util.get_interval_limits(outliers)

sharp_angle_indexes = []
for interval in sharp_angle_intervals:
    sharp_angle_indexes.extend(np.linspace(interval[0],
                                          interval[1],
                                          interval[1]-interval[0]+1,
                                          dtype=int))
```

Sharp angles:

In [15]:

```
df.iloc[sharp_angle_indexes]
```

Out[15]:

latitude	longitude	distance	duration	speed	acceleration	heading	angle	elevation	timestamp	idx	z_duration	z_spec
----------	-----------	----------	----------	-------	--------------	---------	-------	-----------	-----------	-----	------------	--------

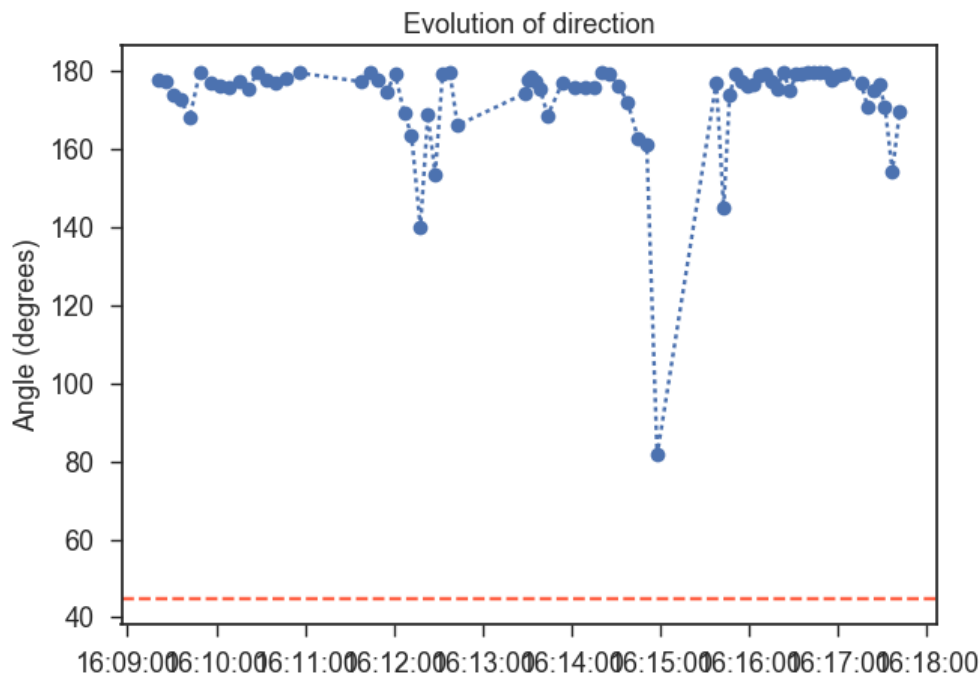


In [16]:

```
plt.figure()

plt.title('Evolution of direction')
plt.ylabel('Angle (degrees)')

plt.plot(df['angle'][::-1], 'o')
plt.plot(df.iloc[sharp_angle_indexes]['angle'], 's', color='tomato')
plt.axhline(minimum_angle, color='tomato', ls='--')
plt.gca().xaxis.set_major_formatter(
    matplotlib.dates.DateFormatter("%H:%M:%S"));
```



Remove outliers

In [17]:

```
noise_indexes = np.union1d(long_distance_indexes, sharp_angle_indexes)
noise_indexes = np.array(noise_indexes, dtype=int)
noise_intervals = util.get_interval_limits(noise_indexes)

noise_indexes = np.array([])
for interval in noise_intervals:
    linspace = np.linspace(interval[0], interval[1], interval[1]-interval[0]+1)
    noise_indexes = np.concatenate((noise_indexes, linspace))

noise_indexes = np.array(np.sort(noise_indexes), dtype=int)
noise = df.iloc[noise_indexes]

df_clean = df.drop(df.index[noise_indexes])

original_len = len(df)
without_outliers_len = len(df_clean)

print('original:', original_len)
print('without outliers:', without_outliers_len)
print('{:.2f}% of points have been dropped'.format((1 - without_outliers_len/original_len)*100))
```

```
original: 76
without outliers: 73
3.95% of points have been dropped
```

Show noisy points:

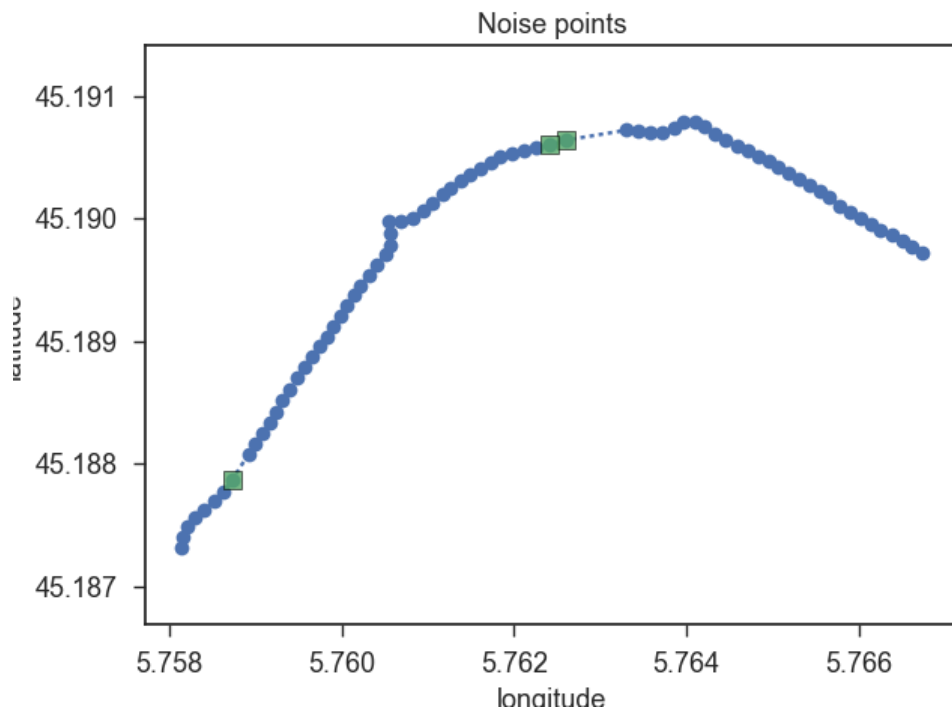
In [18]:

```
fig = plt.figure()
ax = fig.add_subplot(111)
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Noise points')

xx = df.longitude
yy = df.latitude
plt.plot(xx, yy, 'o:', ms=7, alpha=1)

xx = df.iloc[noise_indexes].longitude
yy = df.iloc[noise_indexes].latitude
plt.plot(xx, yy, 's', ms=9, alpha=.8, mew=.5, mec='k')

central_latitude = sum(plt.axes().get_ylim())/2.
mercator_aspect_ratio = 1/math.cos(math.radians(central_latitude))
plt.axes().set_aspect(mercator_aspect_ratio, adjustable='datalim')
```



Stop classification

Analysis of duration between points

Statistics about duration:

In [19]:

```
df_clean['duration'].describe()
```

Out[19]:

```
count    73.000000
mean      6.191781
std       6.043034
min       2.000000
25%      4.000000
50%      5.000000
75%      6.000000
max      42.000000
Name: duration, dtype: float64
```

Modified z-score computation:

In [20]:

```
duration_with_jitter = pp.jitter(df_clean['duration'], jitter)
mz_duration = st.modified_zscore(duration_with_jitter)
long_duration_indexes = [i for i, mz in enumerate(mz_duration) if mz > duration_threshold]
```

Abnormal duration(s):

In [21]:

```
df_clean.iloc[long_duration_indexes]
```

Out[21]:

	latitude	longitude	distance	duration	speed	acceleration	heading	angle	elevation	timestamp	idx	z_dura
2016-07-04 16:11:38	45.190511	5.764833	10.05	42	0.24	0.28	298	177.37	216	2016-07-04 16:11:38	17	4.7195
2016-07-04 16:15:38	45.189882	5.760566	11.73	40	0.29	0.37	174	177.17	214	2016-07-04 16:15:38	47	4.4519

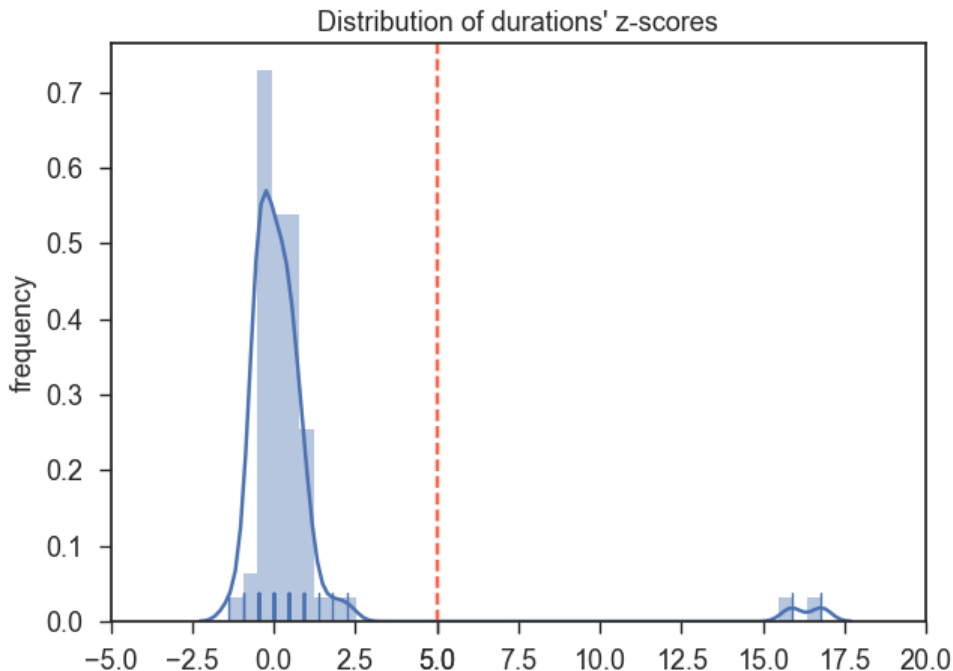
Distribution of modified-zscore of durations:

In [22]:

```
plt.figure()
plt.title("Distribution of durations' z-scores")
plt.ylabel('frequency')

sns.distplot(mz_duration.as_matrix(), rug=True)

plt.xticks(list(plt.xticks()[0]) + [duration_threshold])
plt.axvline(duration_threshold, color='tomato', linestyle='--');
```



Analysis of speed

Statistics about speed:

In [23]:

```
df_clean['speed'].describe()
```

Out[23]:

```
count    73.000000
mean      2.248767
std       0.823770
min       0.240000
25%      1.770000
50%      2.100000
75%      2.800000
max       6.230000
Name: speed, dtype: float64
```

Modified z-score computation:

In [24]:

```
mz_speed = st.modified_zscore(np.log(df_clean['speed']))
slow_speed_indexes = [i for i, mz in enumerate(mz_speed) if mz < -3.5]
```

Abnormal speed(s):

In [25]:

```
df_clean.iloc[slow_speed_indexes]
```

Out[25]:

	latitude	longitude	distance	duration	speed	acceleration	heading	angle	elevation	timestamp	idx	z_dura
2016-07-04 16:11:38	45.190511	5.764833	10.05	42	0.24	0.28	298	177.37	216	2016-07-04 16:11:38	17	4.7195
2016-07-04 16:15:38	45.189882	5.760566	11.73	40	0.29	0.37	174	177.17	214	2016-07-04 16:15:38	47	4.4519

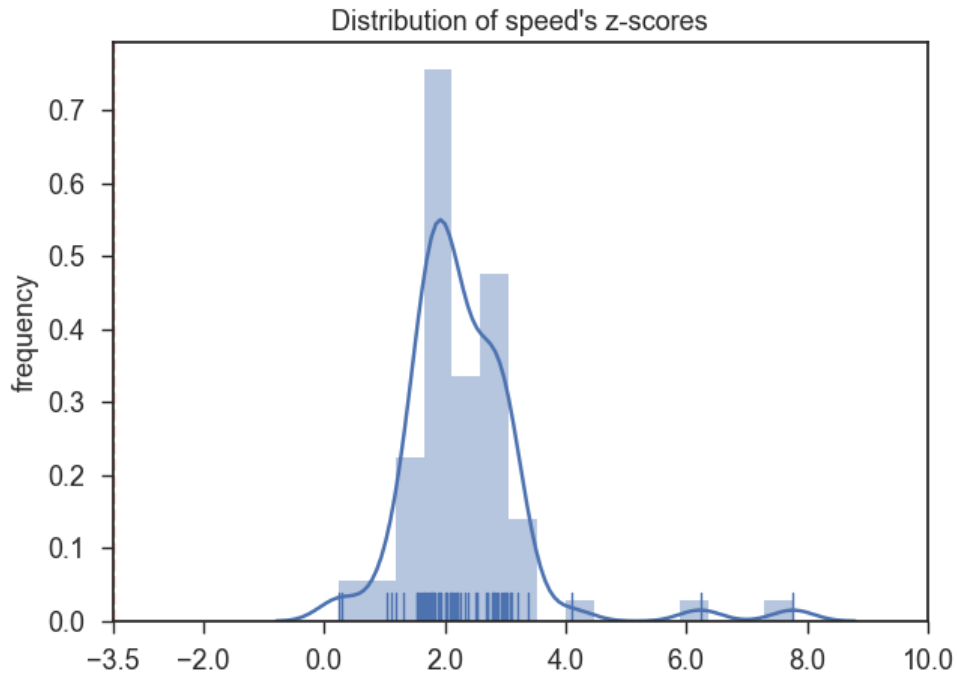
Distribution of modified-zscore of speeds:

In [26]:

```
plt.figure()
plt.title("Distribution of speed's z-scores")
plt.ylabel('frequency')

sns.distplot(df['speed'].as_matrix(), rug=True)

plt.xticks(list(plt.xticks()[0]) + [-speed_threshold])
plt.axvline(-speed_threshold, color='tomato', linestyle='--');
```

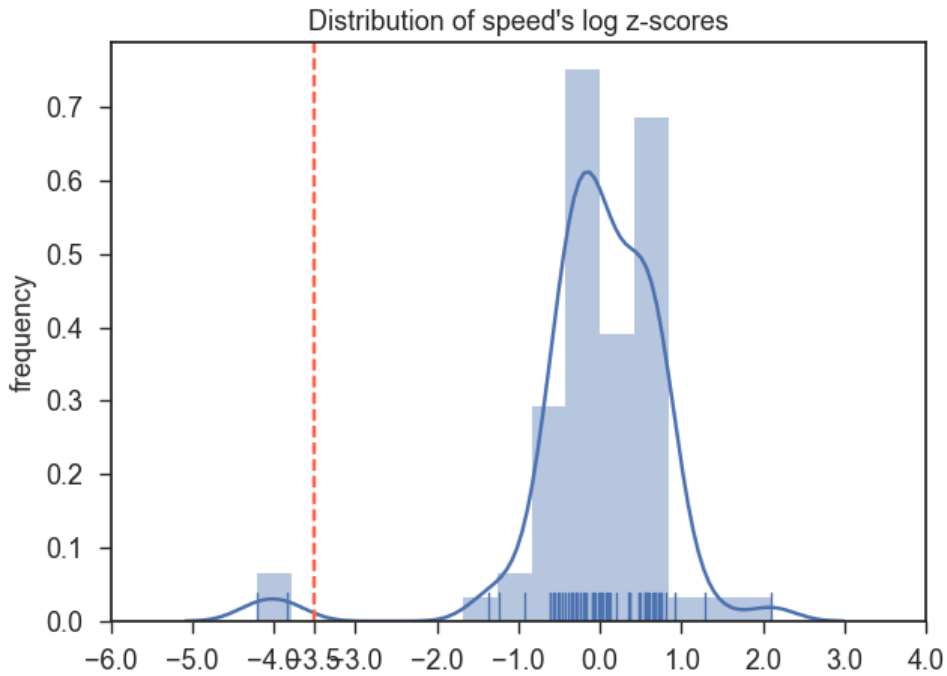


In [27]:

```
plt.figure()
plt.title("Distribution of speed's log z-scores")
plt.ylabel('frequency')

sns.distplot(mz_speed.as_matrix(), rug=True)

plt.xticks(list(plt.xticks()[0]) + [-speed_threshold])
plt.axvline(-speed_threshold, color='tomato', linestyle='--');
```



The intersection of long times and slow speed are probably stops

In [28]:

```
stops = df_clean.iloc[np.intersect1d(long_duration_indexes, slow_speed_indexes)]
stops
```

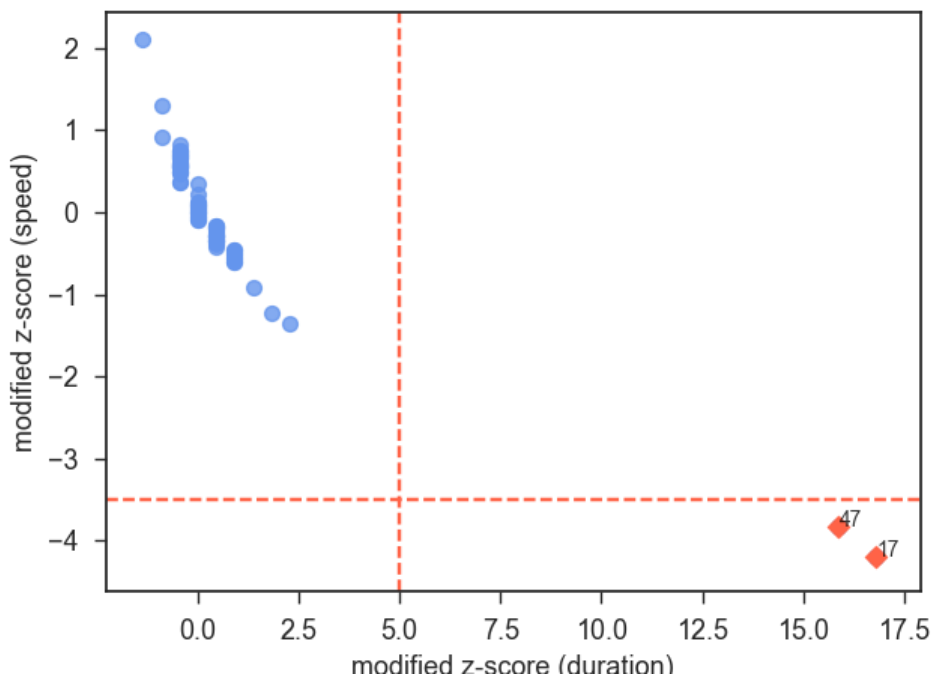
Out[28]:

	latitude	longitude	distance	duration	speed	acceleration	heading	angle	elevation	timestamp	idx	z_dura
2016-07-04 16:11:38	45.190511	5.764833	10.05	42	0.24	0.28	298	177.37	216	2016-07-04 16:11:38	17	4.7195
2016-07-04 16:15:38	45.189882	5.760566	11.73	40	0.29	0.37	174	177.17	214	2016-07-04 16:15:38	47	4.4519

Relation between duration and speed

In [29]:

```
x_label = 'modified z-score (duration)'  
y_label = 'modified z-score (speed)'  
  
x = mz_duration  
y = mz_speed  
X = np.stack((x, y), axis=-1)  
  
fig = plt.figure()  
plt.xlabel(x_label)  
plt.ylabel(y_label)  
ax = fig.add_subplot(111)  
plt.scatter(X[:,0], X[:,1], s=50, alpha=.8, color='cornflowerblue', lw=1)  
plt.axhline(-speed_threshold, color='tomato', linestyle='--')  
plt.axvline(duration_threshold, color='tomato', linestyle='--')  
plt.legend(frameon=True)  
  
for i, element in enumerate(X):  
    if x[i] > duration_threshold and y[i] < -speed_threshold:  
        ax.annotate(df_clean.iloc[i]['idx'], xy=(element[0]+0.008, element[1]+0.001), color='black', alpha=.8)  
        plt.scatter(element[0], element[1], marker='D', s=60, color='tomato')
```



Visualizing the relation of duration, distance, and speed:

Appendix B: Demonstration of the Adaptive Natural Breaks Approximation for time series

Import packages and general settings:

In [19]:

```
import os
import sys

module_path = os.path.abspath(os.path.join('../'))
if module_path not in sys.path:
    sys.path.append(module_path)

from step import msn, anba
from step import preprocessing as pp
from step import util

import math
import numpy as np
import pandas as pnd
from datetime import timedelta

import gpxpy

import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns

%load_ext autoreload
%autoreload 2
%matplotlib notebook

matplotlib.rcParams['axes.labelsize'] = '13'
matplotlib.rcParams['ytick.labelsize'] = '13'
matplotlib.rcParams['xtick.labelsize'] = '13'
matplotlib.rcParams['axes.titlesize'] = '13'
matplotlib.rcParams['legend.edgecolor'] = 'k'
matplotlib.rcParams['legend.shadow'] = True
matplotlib.rcParams['legend.frameon'] = True
matplotlib.rcParams['figure.figsize'] = (8, 3)
sns.set_style('ticks', {"axes.xmargin": 0.2, "axes.ymargin": 0.2});

# For pretty printing
#import warnings
#warnings.simplefilter('ignore')
```

The autoreload extension is already loaded. To reload it, use:
%reload_ext autoreload

Load one trajectory file in GPX format:

In [20]:

```
#file_id = '49996348-1533217229'
file_id = '27031614-1223879101'

file_path = r'gpx/{0}.gpx'.format(file_id)
gpx = gpxpy.parse(open(file_path, 'r'))
```

Compute movement attributes:

In [21]:

```
gpx = pp.discard(gpx, timedelta(seconds=30))
df = pp.compute_attributes(gpx)[1:]

# first 5 rows:
df.head()
```

Out[21]:

	latitude	longitude	distance	duration	speed	acceleration	heading	angle	elevation	timestamp	idx
2015-10-24 13:58:31	45.184790	5.737209	10.09	5	2.02	-0.09	324	128.28	216	2015-10-24 13:58:31	1
2015-10-24 13:58:40	45.184884	5.737247	10.90	9	1.21	0.11	16	120.17	217	2015-10-24 13:58:40	2
2015-10-24 13:58:46	45.184909	5.737384	11.07	6	1.84	-0.01	76	175.72	217	2015-10-24 13:58:46	3
2015-10-24 13:58:52	45.184939	5.737513	10.65	6	1.78	-0.06	71	176.94	217	2015-10-24 13:58:52	4
2015-10-24 13:59:01	45.184976	5.737644	11.03	9	1.23	0.03	68	147.96	217	2015-10-24 13:59:01	5

Plot the trajectory:

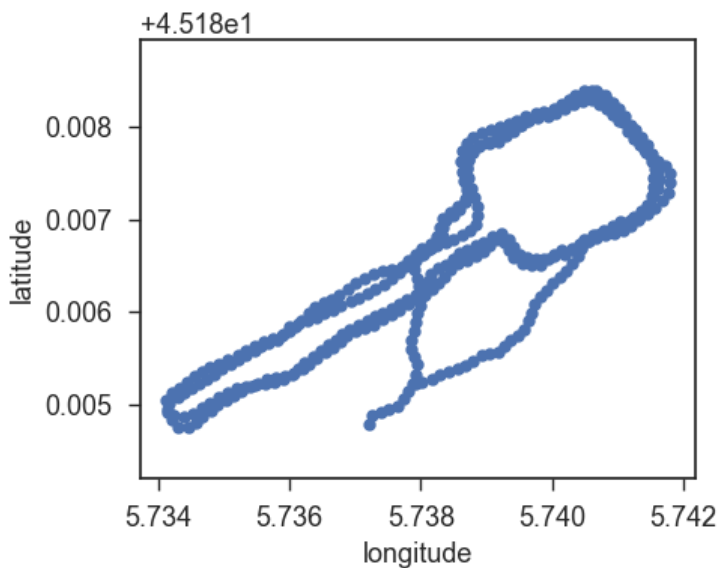
In [22]:

```
fig = plt.figure(figsize=(5,4))
ax = fig.add_subplot(111)

xx = df.longitude
yy = df.latitude

plt.xlabel('longitude')
plt.ylabel('latitude')
plt.plot(df.longitude, df.latitude, 'o:', ms=6)

central_latitude = sum(plt.axes().get_ylim())/2.
mercator_aspect_ratio = 1/math.cos(math.radians(central_latitude))
plt.axes().set_aspect(mercator_aspect_ratio, adjustable='datalim')
plt.tight_layout()
```



Call MSN algorithm to clean the data

In [23]:

```
noise_indexes = msn.get_noise(df)
df_clean = df.drop(df.index[noise_indexes])
stop_indexes = msn.get_stops(df_clean)
```

Smooth the speed data, set speed zero for stops and noise indexes to null:

In [24]:

```
data = df.speed[:].ewm(span=5).mean() #smoothed data
data[stop_indexes] = 0
data[noise_indexes] = None
```

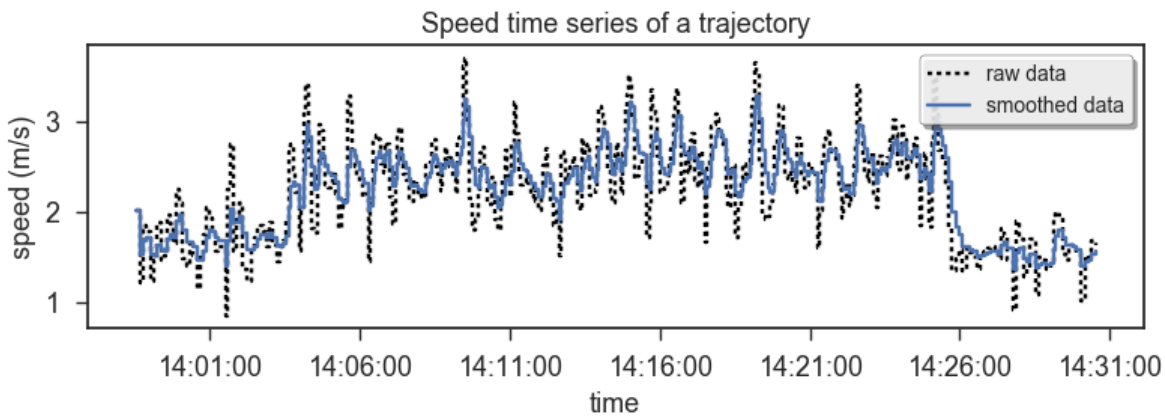
Plot the speed time series:

In [25]:

```
fig = plt.figure()
plt.title("Speed time series of a trajectory")
plt.xlabel('time')
plt.ylabel('speed (m/s)')

plt.plot(df.speed[:, 'k:', label='raw data', drawstyle='steps-post')
plt.plot(data, '-', label='smoothed data', drawstyle='steps-post')

plt.gca().xaxis.set_major_formatter(matplotlib.dates.DateFormatter("%H:%M:%S"));
plt.legend(frameon=True, loc=1)
plt.tight_layout()
```



Parameters

ANBA allows the user to inform either a number in the interval $[0..1)$ (`min_gvf`) or an absolute integer number (`nclasses`).

In the first case, the algorithm tests the GVF (Goodness of Variance Fit) for increasing number of classes until it reaches the desired value of GVF. The second option would inform the number of classes directly, which corresponds to skipping the GVF procedure previously explained.

The user can additionally define a `min_diff` value that is used to avoid two or more approximative values to be too close. In this example, we set `min_diff` to 1, which means that at there should be a difference of at least 1 m/s.

In [26]:

```
min_gvf = 0.5
min_diff = 1
```

Procedure to find the optimal number of classes:

In [27]:

```
gvf = 0.0
nclasses = 1
data_copy = np.copy(data.fillna(0))

while gvf < min_gvf:
    gvf = anba.goodness_of_variance_fit(data_copy, nclasses)
    nclasses += 1

print("{0} classes are needed to represent the input with a GVF of {1}%".format(nclasses, min_gvf*100))
```

3 classes are needed to represent the input with a GVF of 50.0%

Calling the Fisher-Jenks algorithm:

In [28]:

```
breaks = anba.jenks(data_copy, nclasses)
breaks = np.sort(list(set(breaks)))
breaks
```

Out[28]:

```
array([ 1.37240268,  1.95874069,  2.53787483,  3.2865754 ])
```

Finding the median values within each group delimited by the natural breaks:

In [29]:

```
diffs = np.diff(breaks)
groups = [[]] * nclasses

for d in data:
    for i, (break_, diff) in enumerate(zip(breaks, diffs)):
        if d >= break_ and d < break_+diff:
            groups[i] = np.append(groups[i], d)
            break

# cav = Candidate Approximative Values
cav = np.nan_to_num([np.median(x) for x in groups])
cav = np.sort(cav)
cav
```

Out[29]:

```
array([ 1.61745748,  2.36374435,  2.68776801])
```

In [30]:

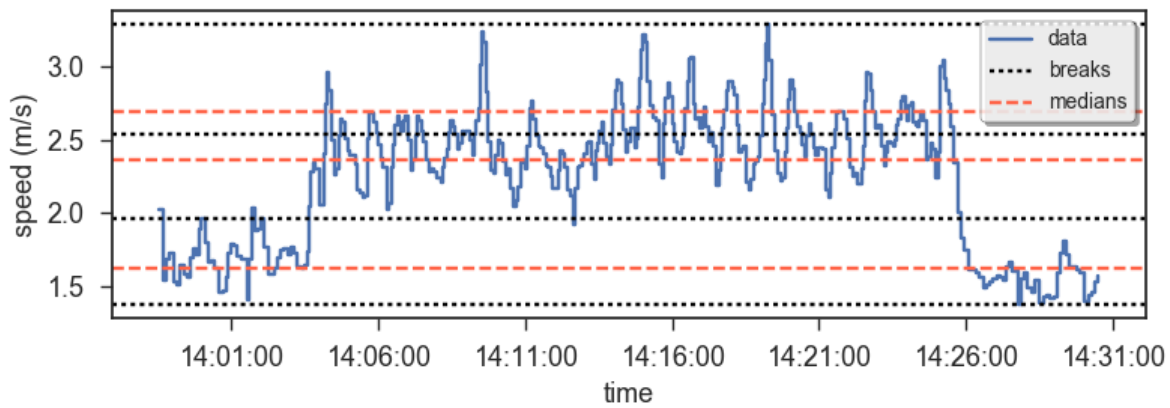
```
plt.figure()
plt.xlabel('time')
plt.ylabel('speed (m/s)')

plt.plot(data, label='data', drawstyle='steps-post')

breaks_legend = True
for b in breaks:
    plt.axhline(b, label='breaks' if breaks_legend else "", linestyle=':', color='k')
    breaks_legend = False

cav_legend = True
for v in cav:
    plt.axhline(v, label='medians' if cav_legend else "", linestyle='--', color='tomato')
    cav_legend = False

plt.gca().xaxis.set_major_formatter(matplotlib.dates.DateFormatter("%H:%M:%S"));
plt.legend(frameon=True, loc=1)
plt.tight_layout()
```



Merge medians according to min_diff parameter

In [31]:

```
#av = Approximative Values
av = cav

cav_diff = np.ediff1d(cav, to_end=np.inf)
mask = [True if d < min_diff else False for d in cav_diff]

if any(mask):
    start = None
    end = None
    intervals = []
    for i, m in enumerate(mask):
        if m:
            if start is None:
                start = i
            else:
                if start is not None:
                    end = i
                    intervals.append([start, end])
                    start = None

    newvalues = []
    oldvalues = []
    for interval in intervals:
        s = interval[0] # start
        e = interval[1] # end
        n = e - s + 1
        if n > 2:
            if (cav[e] - cav[s]) // min_diff >= 2:
                print('>=2')
                newvalues.extend(np.arange(cav[s], cav[e], min_diff))
            elif cav[e] - cav[s] > min_diff:
                newvalues.extend([cav[s], cav[e]])
            else:
                newvalues.extend([cav[s], cav[s] + min_diff])

            oldvalues.extend(cav[s:e + 1])
        elif n == 2:
            newvalues.extend([np.mean([cav[s], cav[e]])])
            oldvalues.extend(cav[s:e + 1])

    kept_cav = [x for x in cav if x not in oldvalues]
    av = np.sort(np.append(kept_cav, newvalues))

av
```

Out[31]:

```
array([ 1.61745748,  2.68776801])
```

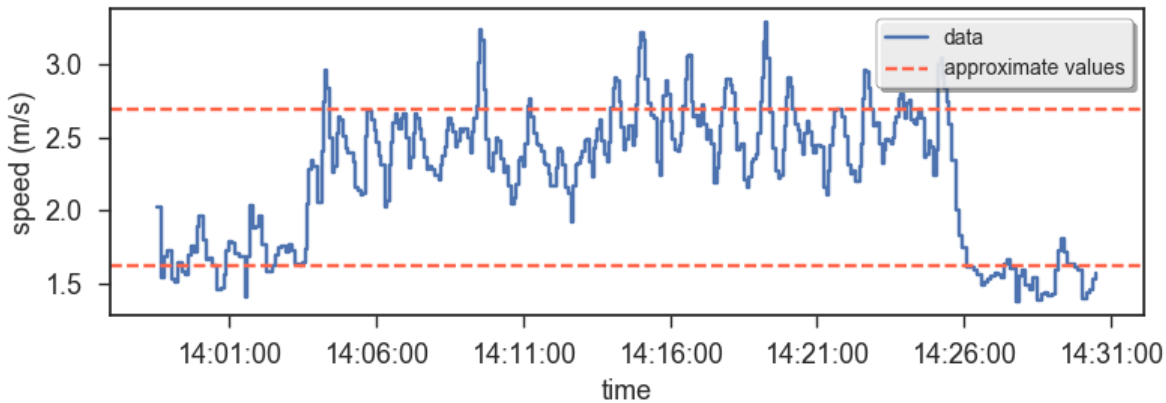
In [32]:

```
plt.figure()
plt.xlabel('time')
plt.ylabel('speed (m/s)')

plt.plot(data, label='data', drawstyle='steps-post')

av_legend = True
for v in av:
    plt.axhline(v, label='approximate values' if av_legend else "", linestyle='--', color='tomato')
    av_legend = False

plt.gca().xaxis.set_major_formatter(matplotlib.dates.DateFormatter("%H:%M:%S"));
plt.legend(frameon=True, loc=1)
plt.tight_layout()
```



Snap data values to approximative values:

In [33]:

```
newgroups = [[]] * nclasses
newdata = []
bins = []

for i, d in enumerate(data):
    absolute_distance = abs(d - av)
    bin_index = np.argmin(absolute_distance)
    bins.append(bin_index)
    newgroups[bin_index] = np.append(newgroups[bin_index], [d], axis=0)
    newdata.append(av[bin_index])

newdata = np.array(newdata)
data[stop_indexes] = 0
newdata[noise_indexes] = None
newdata = pnd.Series(newdata, index=data.index)

data[stop_indexes] = 0
data.iloc[noise_indexes] = None

newdata[stop_indexes] = 0
newdata.iloc[noise_indexes] = None
```

Appendix C: Demonstration of the Intersection-based Spatial Annotation procedure

Import packages and general settings:

In [1]:

```
import os
import sys

module_path = os.path.abspath(os.path.join('../'))
if module_path not in sys.path:
    sys.path.append(module_path)

from step import preprocessing as pp
from step import osm

import math
import re
import numpy as np

import gpxpy
import folium
from rdflib import URIRef, Namespace, Graph
import shapely
from shapely import wkt
from shapely.ops import polygonize_full, cascaded_union
import geojson

import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns

%load_ext autoreload
%autoreload 2
%matplotlib inline

matplotlib.rcParams['axes.labelsize'] = 'large'
matplotlib.rcParams['ytick.labelsize'] = 'medium'
matplotlib.rcParams['xtick.labelsize'] = 'medium'
matplotlib.rcParams['legend.edgecolor'] = 'k'
matplotlib.rcParams['legend.shadow'] = True
matplotlib.rcParams['figure.figsize'] = (5, 4)
sns.set_style('ticks', {"axes.xmargin": 0.2, "axes.ymargin": 0.2});

# For pretty printing
import warnings
warnings.simplefilter('ignore')
```

INFO:rdflib:RDFLib Version: 4.2.1

Load one trajectory file in GPX format:

In [2]:

```
file_path = r'gpx/49996348-1079694145.gpx'
gpx = gpxpy.parse(open(file_path, 'r'))
```

In [3]:

```
points_data = gpx.get_points_data()
points = [(pdata.point.latitude, pdata.point.longitude) for pdata in points_data]
```

In [4]:

```
tileset = r'http://{s}.tile.openstreetmap.se/hydda/full/{z}/{x}/{y}.png'
attribution = 'Tiles courtesy of <a href="http://openstreetmap.se/" target="_blank">OpenStreetMap Sweden</a> \
&mdash; Map data &copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>'

_map = folium.Map(tiles=tileset, attr=attribution, max_zoom=25)

start_mark = folium.Marker([points[0][0], points[0][1]], icon=folium.Icon(color='green', icon='glyphicon glyphicon-play'))
start_mark.add_to(_map)

end_mark = folium.Marker([points[-1][0], points[-1][1]], icon=folium.Icon(color='red', icon='glyphicon glyphicon-stop'))
end_mark.add_to(_map)

line = folium.PolyLine(points, opacity=0.8)
line.add_to(_map)
_map.fit_bounds(line.get_bounds())

_map
```

Out[4]:



Parameters

- `buffer_size` defines the area to be created around each polygon retrieved from OSM
- `minimum_length_ratio` defines the minimum spatial intersection between trajectory and polygon to be considered as relevant
- `minimum_duration_ratio` defines the minimum temporal intersection between trajectory and polygon to be considered as relevant
- `keys` defines which OSM keys are of interest for retrieving OSM features, an empty array means that all keys should be returned
- `expand_keys` sets whether the keys (if not empty) should be expanded according to the related keys found by OSN

In [5]:

```
buffer_size = 0.0002 # ~20 meters
minimum_length_ratio = 0.2 #delta_s
minimum_duration_ratio = 0.2 #delta_t

keys = ["leisure"]
expand_keys = True
```

In [6]:

```
excluded_keys = ['name', 'comment', 'source', 'boundary']
expanded_keys = []
```

Load Open Semantic Network (OSN) and LinkedGeoData Ontology (LGDO) graphs:

In [7]:

```
# Search for tags in OSN
if 'osn_graph' not in locals():
    osn_graph = Graph()
    osn_graph.parse('osm_semantic_network.skos.rdf')

if 'lgdo_graph' not in locals():
    lgdo_graph = Graph()
    lgdo_graph.parse('lgdo_2014-07-26.n3', format='n3')
```

Using LinkedGeoData to expand keys:

In [8]:

```
if expand_keys and len(keys) > 0:
    expanded_keys = []
    keys_regex = "|".join(keys)
    regex = r"http://spatial.ucd.ie/lod/osn/term/k:{0}.*".format(keys_regex)

    query = """
PREFIX skos:<http://www.w3.org/2004/02/skos/core#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>

SELECT ?subject ?relatedTerm ?lgdConcept
{
  ?subject skos:related ?relatedTerm .
  ?relatedTerm skos:exactMatch ?lgdConcept .

  FILTER (REGEX(STR(?subject), "" + regex + ""))
  FILTER (STRSTARTS(STR(?lgdConcept), "http://linkedgeo.org/ontology"))
}"""

    print(query)

    response = osn_graph.query(query)

    for row in response:
        uri = row[1]
        new_key = uri.split('k:')
        if len(new_key) > 1:
            new_key = new_key[-1]
        else:
            continue

        try:
            new_key = new_key.split('/')[0]
        except:
            pass

        if new_key not in keys:
            expanded_keys.append(new_key)

    expanded_keys = list(set(expanded_keys))

    print()
    print('Input keys: {0}'.format(keys))
    print('{0} new terms found in OSN: {1}'.format(len(expanded_keys), expanded_keys))
```

```
PREFIX skos:<http://www.w3.org/2004/02/skos/core#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>

SELECT ?subject ?relatedTerm ?lgdConcept
{
  ?subject skos:related ?relatedTerm .
  ?relatedTerm skos:exactMatch ?lgdConcept .

  FILTER (REGEX(STR(?subject), "http://spatial.ucd.ie/lod/osn/term/k:(leisure).*"))
  FILTER (STRSTARTS(STR(?lgdConcept), "http://linkedgeo.org/ontology"))
}
```

Input keys: ['leisure']

17 new terms found in OSN: ['barrier', 'shop', 'natural', 'building', 'fishing', 'landuse', 'boundary', 'sport', 'playground', 'tourism', 'harbour', 'waterway', 'route', 'highway', 'amenity', 'shelter', 'man_made']

Simplify trajectory's shape and create a WKT representation of it:

In [9]:

```
gpx_simple = gpx.clone()
gpx_simple.simplify()

linestring_simple = "LINESTRING("

simple_points = gpx_simple.get_points_data()

for i, point_data in enumerate(simple_points):
    point = point_data.point
    linestring_simple += str(point.longitude) + " " + str(point.latitude)
    if i < len(simple_points)-1:
        linestring_simple += ", "

linestring_simple += ")"
linestring_simple
```

Out[9]:

```
'LINESTRING(5.73425791 45.18210998, 5.7353382904 45.1824220976, 5.7408826096 45.1847253391, 5.7429230865
45.186140418, 5.7424419076 45.1868992116, 5.7424473461 45.1880505119, 5.7428454475 45.1889469214, 5.7430
898764 45.1890251298, 5.7437873918 45.1889355192, 5.7451961052 45.1884058183, 5.7454810635 45.1883963889,
5.7481449371 45.188360427, 5.7494157359 45.1884815264, 5.7504363 45.1888904565, 5.7492650439 45.188532939
3, 5.7468814271 45.1884723829, 5.7453555906 45.1887051998, 5.743829997 45.1891110944, 5.7429074605 45.188
9185302, 5.7425774415 45.1883845097, 5.742458776 45.1875742611, 5.7422837729 45.1872283783, 5.7425883036
45.1864469882, 5.7426182096 45.185734623, 5.7422678227 45.1855394129)'
```

Using Overpass to retrieve OSM features:

In [10]:

```
all_keys = keys + expanded_keys

osm_features = osm.get_spatial_features(gpx, all_keys)

[out:json];
(way[~"leisure|barrier|shop|natural|building|fishing|landuse|boundary|sport|playground|tourism|ha
rbour|waterway|route|highway|amenity|shelter|man_made"~"."];
(poly:"45.18210998 5.73425791 45.1824220976 5.7353382904 45.1847253391 5.7408826096 45.186140418
5.7429230865 45.1868992116 5.7424419076 45.1880505119 5.7424473461 45.1889469214 5.7428454475 45.1890251
298 5.7430898764 45.1889355192 5.7437873918 45.1884058183 5.7451961052 45.1883963889 5.7454810635 45.1883
60427 5.7481449371 45.1884815264 5.7494157359 45.1888904565 5.7504363 45.1885329393 5.7492650439 45.18847
23829 5.7468814271 45.1887051998 5.7453555906 45.1891110944 5.743829997 45.1889185302 5.7429074605 45.188
3845097 5.7425774415 45.1875742611 5.742458776 45.1872283783 5.7422837729 45.1864469882 5.7425883036 45.1
85734623 5.7426182096 45.1855394129 5.7422678227");
rel(bw);

relation[~"leisure|barrier|shop|natural|building|fishing|landuse|boundary|sport|playground|touris
m|harbour|waterway|route|highway|amenity|shelter|man_made"~"."];
(poly:"45.18210998 5.73425791 45.1824220976 5.7353382904 45.1847253391 5.7408826096 45.186140418
5.7429230865 45.1868992116 5.7424419076 45.1880505119 5.7424473461 45.1889469214 5.7428454475 45.1890251
298 5.7430898764 45.1889355192 5.7437873918 45.1884058183 5.7451961052 45.1883963889 5.7454810635 45.1883
60427 5.7481449371 45.1884815264 5.7494157359 45.1888904565 5.7504363 45.1885329393 5.7492650439 45.18847
23829 5.7468814271 45.1887051998 5.7453555906 45.1891110944 5.743829997 45.1889185302 5.7429074605 45.188
3845097 5.7425774415 45.1875742611 5.742458776 45.1872283783 5.7422837729 45.1864469882 5.7425883036 45.1
85734623 5.7426182096 45.1855394129 5.7422678227"););
out geom;
```

In [11]:

```
print('{0} features retrieved.'.format(len(osm_features)))  
#first 5 features:  
osm_features.head()
```

116 features retrieved.

Out[11]:

	geometry	label	relation	tags
id				
80348	MULTILINESTRING((5.7154067 45.1997808,5.715250...	Grenoble	None	{'name:sr': 'Гренобл', 'ref:INSEE': '38185', '...
80349	MULTILINESTRING((5.7430295 45.1663750,5.743081...	Saint-Martin-d'Hères	None	{'wikipedia': 'fr:Saint-Martin-d'Hères', 'addr...
278498	MULTILINESTRING((5.6147474 45.2952120,5.614575...	L'Isère	None	{'source': 'cadastre-dgi-fr source : Direction...
558483	MULTILINESTRING((5.7428003 45.1917672,5.742812...	Pont du Sablon	None	{'maxweight': '20', 'type': 'bridge', 'name': '...
558491	MULTILINESTRING((5.7491122 45.1888764,5.748643...	Pont de la Porte de Savoie	None	{'type': 'bridge', 'name': 'Pont de la Porte d...

Cleaning the dataset to remove OSM features that do not have a tags with corresponding concepts in LGDO:

In [12]:

```
osn_terms = []
invalid_chars = [' ', '<', '>', '|']

for tags in osm_features['tags']:
    for item in tags.items():
        if item[0] not in excluded_keys and re.match('[a-z]*', item[0]) and re.match('[a-z]*', item[1]):
            uri = "http://spatial.ucd.ie/lod/osn/term/k:{0}/v:{1}".format(item[0], item[1])
            uri = ''.join([i if i not in invalid_chars else '_' for i in uri])
            osn_terms.append("<{0}>".format(uri))

query = """
PREFIX skos:<http://www.w3.org/2004/02/skos/core#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>

SELECT ?osnConcept ?lgdoConcept
{
?osnConcept skos:exactMatch ?lgdoConcept .

FILTER (?osnConcept IN (""" + ", ".join(osn_terms) + """"))
FILTER (STRSTARTS(STR(?lgdoConcept), "http://linkedgeodata.org/ontology"^^xsd:string))
}
"""

response = osn_graph.query(query)

valid_tags = []
lgd_concepts = []
for row in response:
    osn_concept = row[0]
    osn_concept = 'k:{0}'.format(osn_concept.split('k:')[1])
    valid_tags.append(osn_concept)
    lgd_concepts.append(str(row[1]).split('/')[1])

print('{0} valid tags: {1}\n\ncorresponding to the LGD concepts: {2}'.format(
    len(valid_tags), valid_tags, lgd_concepts))

#Exclude features that don't have the valid tags
to_drop = []
for tags, idx in zip(osm_features['tags'], osm_features.index):
    feature_tags = ['k:{0}/v:{1}'.format(item[0], item[1]) for item in tags.items()]
    if not any([tag in valid_tags for tag in feature_tags]):
        to_drop.append(idx)

print('{0} elements before cleaning'.format(len(osm_features)))
osm_features = osm_features.drop(to_drop)
print('{0} elements after cleaning'.format(len(osm_features)))
```

```
20 valid tags: ['k:route/v:bus', 'k:highway/v:residential', 'k:natural/v:wood', 'k:highway/v:steps', 'k:landuse/v:grass', 'k:leisure/v:sports_centre', 'k:landuse/v:residential', 'k:highway/v:service', 'k:highway/v:pedestrian', 'k:highway/v:secondary', 'k:route/v:bicycle', 'k:highway/v:footway', 'k:waterway/v:stream', 'k:leisure/v:park', 'k:highway/v:cycleway', 'k:highway/v:path', 'k:barrier/v:fence', 'k:amenity/v:parking', 'k:landuse/v:forest', 'k:natural/v:water']
```

```
corresponding to the LGD concepts: ['Bus', 'HighwayResidential', 'NaturalWood', 'Steps', 'Grass', 'Sports Centre', 'LanduseResidential', 'HighwayService', 'Pedestrian', 'Secondary', 'RouteBicycle', 'Footway', 'Stream', 'Park', 'Cycleway', 'Path', 'Fence', 'Parking', 'LanduseForest', 'NaturalWater']
```

```
116 elements before cleaning
```

```
86 elements after cleaning
```

Creating closed polygons with buffers:

In [13]:

```
linestring = "LINESTRING("

for i, point_data in enumerate(points_data):
    point = point_data.point
    linestring += str(point.longitude) + " " + str(point.latitude)
    if i < len(points_data)-1:
        linestring += ", "

linestring += ")"

trajectory = wkt.loads(linestring)

features = []
features_json = []
marker_data = []

to_drop = []
for item in osm_features.itertuples():
    try:
        feature0 = wkt.loads(item.geometry)
        json_feature0 = geojson.Feature(geometry=feature0, properties={"id": str(item.Index)})

        feature1 = feature0.buffer(buffer_size)
        json_feature1 = geojson.Feature(geometry=feature1)

        feature2 = polygonize_full(feature0)[0]
        json_feature2 = geojson.Feature(geometry=feature2)

        feature3 = cascaded_union([feature1, feature2])
        json_feature3 = geojson.Feature(geometry=feature3)

        #0: original geometry (linestring)
        #1: buffered 0 (polygon)
        #2: polygonized
        #3: union of 1 and 2
        # This is for visualization purposes.
        # A more efficient solution would be to buffer the polygonized line (feature2)
        features_json.append([json_feature0, json_feature1, json_feature2, json_feature3])
        features.append(feature3)

        rep_point = feature3.representative_point()
        popup_text = "{0}\n{1}".format(str(item.Index), repr(item.tags))
        if item.relation and len(item.relation) > 0:
            popup_text += " | part of: " + str(item.relation)
        marker_data.append([(rep_point.y, rep_point.x), popup_text])
    except Exception as e:
        #print(e)
        #print(item.Index)
        #print(item.Label)
        #print(item.geometry)
        to_drop.append(item.Index)

osm_features = osm_features.drop(to_drop)
```

```
ParseException: Expected 'Z', 'M', 'ZM', 'EMPTY' or '(' but encountered : ')'
ERROR:shapely.geos:ParseException: Expected 'Z', 'M', 'ZM', 'EMPTY' or '(' but encountered : ')'
```

In [14]:

```
traj_map = folium.Map(tiles=tileset, attr=attribution, max_zoom=25)

for i, feat in enumerate(features_json):
    traj_map.choropleth(geo_str=feat[0], line_color='black', line_weight=3)
    #traj_map.choropleth(geo_str=feat[1], fill_color='gray', line_weight=1, fill_opacity=0.2)
    #traj_map.choropleth(geo_str=feat[2], fill_color='green', line_weight=1, fill_opacity=0.2)
    traj_map.choropleth(geo_str=feat[3], fill_color='red', line_weight=1, fill_opacity=.1)
    folium.Marker(location=marker_data[i][0], popup=marker_data[i][1]).add_to(traj_map)

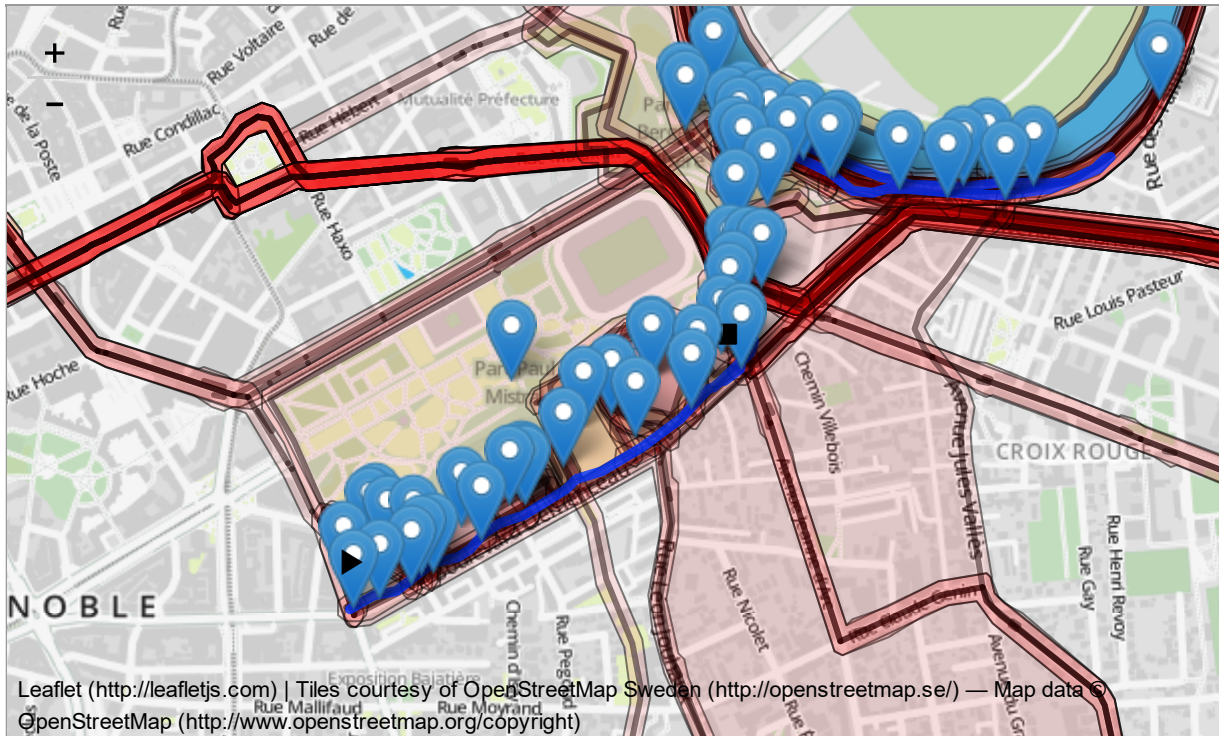
start_mark.add_to(traj_map)
end_mark.add_to(traj_map)

line.add_to(traj_map)

traj_map.fit_bounds(line.get_bounds())

traj_map
```

Out[14]:



Compute the intersections:

In [15]:

```
feature_intervals = osm.compute_intersections(gpx, osm_features)
print("{0} intersections retrieved.".format(len(feature_intervals)))
```

155 intersections retrieved.

In [16]:

```
groups = feature_intervals.groupby(level=0)
print(len(groups), 'unique features.')
```

83 unique features.

Select OSM features based on parameters:

In [17]:

```
selected = []
for group in groups:
    if group[1]['length_ratio'].sum() > minimum_length_ratio or group[1]['duration_ratio'].sum() > minimum_duration_ratio:
        print('')
        print(group[0])
        print('relations:', osm_features.loc[group[0]]['relation'])
        print(osm_features.loc[group[0]]['label'], osm_features.loc[group[0]]['tags'])
        print("length ratio: {:.2f}%".format(group[1]['length_ratio'].sum()*100))
        print("duration ratio: {:.2f}%".format(group[1]['duration_ratio'].sum()*100))
        selected.append(group[0])
```

278498

relations: None

L'Isère {'source': 'cadastre-dgi-fr source : Direction Générale des Impôts - Cadastre; mise à jour : 2009', 'type': 'multipolygon', 'note': 'taggé en mode « new tagging » cf wiki', 'name': "L'Isère", 'natural': 'water', 'comment': 'Provenance du cadastre, les bords semblent correspondre au seuil maximal inondable', 'water': 'river', 'boat': 'no'}

length ratio: 34.29%

duration ratio: 34.83%

2739358

relations: None

de Château Neuf sur Isère - à Chanaz {'route': 'bicycle', 'ref': 'V 63', 'type': 'route', 'name': 'de Château Neuf sur Isère - à Chanaz', 'network': 'ncn', 'source': 'http://www.developpement-durable.gouv.fr'}

length ratio: 39.12%

duration ratio: 40.76%

2808054

relations: None

Rive Gauche de l'Isère, amont de Grenoble {'route': 'bicycle', 'type': 'route', 'name': "Rive Gauche de l'Isère, amont de Grenoble", 'network': 'rcn'}

length ratio: 39.12%

duration ratio: 40.76%

3922430

relations: None

13 : Poisat => Meylan {'ref': '13', 'network': 'TAG', 'operator': 'Semitag Eybens', 'to': 'Meylan - Lycée du Grésivaudan', 'wheelchair': 'yes', 'route': 'bus', 'type': 'route', 'name': '13 : Poisat => Meylan', 'from': 'Poisat - Prémol', 'public_transport:version': '2', 'colour': '#28b3b4', 'opening_hours': 'Mo-Sa 05:30-20:30; Su 06:30-20:30'}

length ratio: 20.78%

duration ratio: 20.63%

4014810

relations: []

Parc Paul Mistral {'name': 'Parc Paul Mistral', 'leisure': 'park'}

length ratio: 30.28%

duration ratio: 29.11%

6690012

relations: None

None {'operator': "Département de l'Isère", 'ref': 'VV3', 'type': 'route', 'route': 'bicycle', 'network': 'lcn'}

length ratio: 39.12%

duration ratio: 40.76%

28340980

relations: []

Parc des Berges de l'Isère {'alt_name': 'Parc Paul Mistral', 'name': "Parc des Berges de l'Isère", 'leisure': 'park'}

length ratio: 35.30%

duration ratio: 33.40%

55695582

relations: [558483, 558491]

None {'flood_prone': 'yes', 'source': 'GPS', 'highway': 'path', 'surface': 'dirt'}

length ratio: 24.41%

duration ratio: 23.70%

264671213

relations: [558483, 558491, 2739358, 2808054, 6690012]

None {'ref': 'VV3', 'note': 'Voie verte', 'surface': 'asphalt', 'motor_vehicle': 'no', 'smoothness': 'excellent', 'lit': 'yes', 'oneway': 'no', 'foot': 'designated', 'highway': 'path', 'width': '3', 'bicycle': 'designated', 'embankment': 'yes', 'segregated': 'no'}

length ratio: 33.22%

duration ratio: 35.34%

382417716

relations: []

Châtelet Abbaye Jouhaux {'quarter': 'suburb', 'landuse': 'residential', 'name': 'Châtelet Abbaye Jouhaux'}

length ratio: 32.66%

duration ratio: 32.48%

440478753

relations: []

None {'natural': 'wood'}

length ratio: 42.48%

duration ratio: 43.82%

