UNIVERSITÀ DEGLI STUDI DI MILANO

GRADUATE SCHOOL IN COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

"GIOVANNI DEGLI ANTONI"

PhD in Computer Science
Cycle XXX

Segmentation techniques based on clustering for the analysis of mobility data

Supervisor: Prof. Maria Luisa Damiani

PhD school Headmaster: Prof. Paolo Boldi

PhD Thesis of:
Fatme Hachem

Academic year 2016-2017

Abstract

The Thesis focuses on segmentation methods for the partitioning of spatial trajectories in semantically meaningful sub-trajectories and their application to the analysis of mobility behavior. Spatial trajectories are complex structured data consisting of sequences of temporally ordered spatio-temporal points sampling the continuous movement of an object in a reference space. Spatial trajectories can reveal behavioral information about individuals and groups of individuals, and that motivates the concern for data analysis techniques.

Segmentation techniques are key for the analysis of spatial trajectories. In general, the segmentation task partitions a sequence of data points in a series of disjoint sub-sequences based on some homogeneity criteria. The Thesis focuses, in particular, on the use of clustering methods for the segmentation of spatial trajectories. Unlike the traditional clustering task, which is applied to *sets* of data points, the goal of this class of techniques is to partition *sequential* data in temporally separated clusters. Such techniques can be utilized for example to detect the sequences of places or regions visited by moving objects. While a number of techniques for the cluster-based segmentation are proposed in literature, none of them is really robust again noise, while the methodologies put in place to validate those techniques suffer from severe limitations, e.g., simple datasets, no comparison with ground truth. This Thesis focuses on a recent cluster-based segmentation method, called SeqScan, proposed in previous work. This technique promises to be robust against noise, nonetheless the approach is empirical and lacks a formal and theoretical framework.

The contribution of this research is twofold. First it provides analytical support to SeqScan [27], defining a rigorous framework for the analysis of the properties of the model. The method is validated through an extensive experimentation conducted in an interdisciplinary setting and contrasting the segmentation with ground truth. The second contribution is the proposal of a technique for the discovery of a collective pattern, called *gathering*. The gathering pattern describes a situation in which a significant number of moving objects share the same region, for enough time periods with possibility of occasional absences, e.g. a concert, an exhibition. The technique is built on SeqScan.

A platform, called MigrO, has been finally developed, including not only the algorithms but also a variety of tools facilitating data analysis.

Acknowledgments

Firstly, I would like to express my sincere gratitude to my advisor Prof. Maria Luisa Damiani for her continuous support and help in every step through the rough road to finish this Thesis. Her patience, wisdom and passion for the research have always motivated me to overcome the obstacles and pushed me forward. I am thankful for the excellent example she has provided as a successful woman researcher and professor. She is- and will always be- a role model I look up to in my future career.

I would like to thank the reviewers of my thesis Prof. Francesca Cagnacci, Prof. Osman Abul and Prof. Peer Kröger for their excellent reviews and helpful comments. I truly appreciate the time they spent and the efforts they made for the Thesis correction.

I gratefully acknowledge the scholarship received towards my PhD from Università degli Studi di Milano.

My time in university of Milan was made enjoyable in a large part due to the colleagues and friends. I am grateful for all the beautiful memories we shared. Special thanks go to my lab-mate and dear friend Dr. Hamza Issa for his support from the first moment till the very end. I am thankful for all the stimulating discussions, for all the helpful suggestions and for all the fun we had.

My life experience in Italy has been enriched by the precious friendships I made. I am lucky to be surrounded by true friends who became a family to me. For them, and for those who have never stopped caring and supporting despite the distance...big thanks.

Finally, my deepest gratitude goes to my family...To my Mom, for her never ending love and tenderness, because she always puts her full trust in me, believes in my potentials even in the hardest moments. She is my first and best teacher, and the reason of my success. To my Dad, who raised me to be strong and independent, and made sure I felt his confidence and encouragement. To my brothers and sisters who are always there for me, supporting me through difficulties, and with whom I share sweet and tough moments. This doctoral dissertation is dedicated to them.

This moment has been a dream.It becomes true thanks to your help, support and encouragement. For each and every one of you my deepest appreciation and gratitude.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations and research topics

The comprehension of phenomena related to movement, for example of people, vehicles, animals, has always been a key issue in many areas of scientific investigation and social analysis [41]. Recent years have witnessed a tremendous growth in the collection of trajectory data, while trajectory data analysis has become a prominent research stream [41, 97, 96, 79, 49], with important applications in e.g. urban computing, intelligent transportation, animal ecology. Spatial trajectories, in particular, are sequences of temporally correlated observations describing the movement of an object through a series of points sampling the time-varying location of the object [99].

A major analysis task over trajectories is trajectory *segmentation*. Generally speaking, the segmentation task splits a sequence of data points in a series of disjoint sub-sequences consisting of points that are homogeneous with respect to some criteria [62, 8]. Diverse segmentation criteria have been proposed in literature, even for different purposes, including time series summarization, e.g. [62, 33], and trajectory indexing in databases, e.g. [80, 24]. A study in [31] categorizes the techniques used for the segmentation according to the aim of their usage (pattern description, detection of changing points in a trajectory, processes and states identification). This research focuses on the use of segmentation techniques for the detection of mobility patterns. Mobility patterns can be defined as stereotyped behaviors of moving entities. Patterns are qualified as *individual* or *collective*, depending on the nature of entities, i.e. single individuals or groups. This work is primarily focused on the detection of individual patterns.

Two main classes of approaches have been proposed for the segmentation of individual trajectories, called *attribute-based* and *pattern-based*, respectively [25]. The segmentation is attribute-based if the homogeneity of segments is defined with

respect to selected properties of the movement that can be derived from the geometric properties of spatial trajectories. Movement attributes are, for example, speed, heading, curvature. These techniques are commonly grounded on computational geometry. However, the characterization and generalization of the segmentation criteria, the handling of outliers and the scalability of the segmentation algorithms still represent important challenges.

Probably less general from a theoretical perspective but more flexible, with respect to the application needs, are the pattern-driven approaches. These are segmentation techniques based on data mining and machine learning methods. In particular a number of approaches employees clustering methods for the detection of *stop-and-move* patterns. Stop-and-move is an abstraction of the mobility behavior of an object that repeatedly stays for some time in a small region (i.e. a stop) before moving to some other regions. This behavior can be exhibited at different temporal scales according to the application. For example, at a large scale (spatial and temporal) the stop-and-move pattern can model the migration behavior of an animal from one region to another (Figure 1.1(a)), or at a very small scale it can model the wandering of the eye gaze stopping in *fixations* during the observation of a scene (Figure 1.1(b)).

(a) (b)

Figure 1.1: Segmentation-based on clustering: (a) Synthetic data illustrating a seasonal migration of a wild animal. (b) The sequence of clusters (green points) called fixations identifying the portions of picture attracting the eye gaze of the observer.

In general, segmentation techniques suffer from a major drawback, in that are sensitive to noise, therefore the partitioning is disrupted if noise points are encountered during the scan of the trajectory. To address such an issue, a popular strategy is to introduce some constraints, for example on the maximum number of noise points that can be tolerated before a breakpoint is added and the segment created. Such an approach is however little effective in practice and does not

provide guarantees. A first systematic approach for a more effective handling of noise in clustering-based segmentation is presented in previous work and is called *SeqScan* [27]. SeqScan is a clustering technique for the discovery of stops-and-moves patterns, fully compliant with the DBSCAN model, which finds spatially dense clusters of arbitrary shapes [35]. SeqScan classifies outliers in two categories: the points indicating a temporary absence from the cluster (excursion point) and the points representing a definitive departure from the cluster towards another cluster (transition point). This technique, however, presents important limitations. Firstly, the SeqScan algorithm is grounded on an empirical approach, thus lacks a formal and theoretical model, and that makes the analysis of its properties hard. For example, the variety of patterns that can be extracted using the technique is not precisely specified. As a result, the actual potentialities as well as the limitations of the technique are only partially known. Secondly, the method has been only coarsely and qualitatively evaluated, thus the effectiveness the method is not proved. That motivates the research presented in this Thesis.

The objective of this research is twofold: i) to provide analytical support to the SeqScan method along with a robust validation methodology in collaboration with domain experts. That is important in view of an effective deployment of the method in real applications. ii) To investigate the use of SeqScan as building block for the discovery of collective patterns, such as the *gathering* pattern. Further details are provided in the following.

1.1.1 Analyzing SeqScan

SeqScan relies on the density-based paradigm. Temporal extensions of such paradigm include ST-DBSCAN[14] and DenStream [18]. However, unlike these DBSCAN-like techniques, the SeqScan clustering is strictly dependent on the order in which the points of the database are visited as well as on the temporal distance between consecutive points [35]. More specifically, the algorithm leverages the temporal relationship among points to detect a sequence of clusters. Relevant properties of SeqScan are:

- Conceptual framework. The cluster model includes the concepts of presence/absence in/from a cluster and transition from one cluster to another cluster. The clusters represent the stops, i.e. *stay regions*, while the transitions are the moves in the stop-and-move model.

- Linear ordering of clusters. Clusters are temporally disjoint and form a sequence. The noise points local to the cluster do not affect the clustering.

- Spatial separation of consecutive clusters. The separation in space can be weak or strong. Intuitively the separation is weak when the clusters can

overlap. For example, the weak separation between two consecutive clusters may be due to the presence of a moving cluster, that is the object moves progressively back to the region occupied by the previous cluster.

- Possible overlapping of non-consecutive clusters. An object can return to the same region after an arbitrary period spent in another stay region. Clusters that are spatially identical, but occurring at different times, are distinct.

- Parameters. The technique requires three parameters. Two are the density parameters requested by DBSCAN, the third parameter is the minimum *presence* in the cluster. Presence is a key concept. Unlike duration, presence indicates the minimum time spent in the region at the net of the periods of absence.

- Support for the choice of the temporal parameter. The relationship between number of clusters and minimum presence in a cluster translates into a function that helps determine the desired level of temporal granularity of clusters.

1.1.2 Validation methodology

Extensive experimentations have been conducted for the validation of the SeqScan algorithm. In particular, the goal of the task is to evaluate to what extent the cluster-based segmentation matches the ground truth (*external validation*). The ground truth consists of labeled trajectories simulating the movement of animals ("animal dataset"). In earlier work, SeqScan was evaluated using real data. However, real data is typically noisy while domain experts may be not able to classify every point with sufficient confidence. As a consequence, the early evaluation was only conducted at a coarse level. In the Thesis, the evaluation task is performed using synthetic data. The use of synthetic dataset generated by a simulator conceptually encompassing the pattern of concern, but developed independently from this research has dramatically improved the accuracy of the evaluation and guaranteed its fairness. In addition, the evaluation has been conducted using blind experiments in collaboration with biologists from Fondazione E. Mach, Trento. Based on the animal dataset, the sensitivity of SeqScan to key internal and external parameters have been finally analyzed. The experiments show a high degree of matching between the output of SeqScan and the ground truth. In addition, the technique is robust against low sampling rates, thus can be potentially employed in a variety of applications.

To conduct the experiments, a software platform has been developed encompassing the SeqScan algorithm and a number of tools supporting statistical and visual analysis. The platform is called MigrO. It has been developed on top of

the Quantum GIS system (a popular open source Geographical Information System). The system has been presented at the demo session of the conference ACM SIGSPATIAL 2015 and has received the Best Demo Award.

1.1.3 From individual to collective patterns

A related stream of research investigates how to extract collective patterns based on individual patterns. Specifically, the goal of the research conducted on this topic is to analyze the collective behavior of objects by leveraging SeqScan capabilities. The research focuses, in particular, on the discovery of collective patterns known as *gathering* [95]. A gathering can be defined as a group of individuals that share the same region approximatively at the same time and for an identical purpose. An example of event entailing a gathering is that of a concert and exhibition, where a group of persons are located in some region for a certain time. In order to discover the gatherings of at least k objects a technique called *k-Gathering* is developed. The technique is implemented and embedded in the MigrO platform. The experiments conducted on both synthetic and real data show that this is a promising direction.

1.2 Contributions and organization

The rest of the Thesis is structured as follows:

- Chapter 2 presents the state of the art on techniques for the segmentation of spatial trajectories. Most of the material presented in the chapter has been published in [25]. This study emphasizes the importance of segmentation methods for the summarization of spatial trajectories.

- Chapter 3 presents the analysis of the SeqScan technique. The key concepts are rigorously presented, while the soundness of the proposed algorithm is discussed. The algorithm substantially reshapes the early version. Most of the material of this chapter and of Chapter 4 is reported in a submitted article [26].

- Chapter 4 presents the validation methodology and the broad experimental work conducted in collaboration with biologists.

- Chapter 5 presents the research conducted on collective pattern discovery. The k-Gathering technique for the detection of gatherings of at least k-objects is presented. Most of the material presented in the chapter has been reported in a publication [50].

- Chapter 6 describes the MigrO platform developed during the project and sketches a few applications using SeqScan and k-Gathering.
- Chapter 7 concludes the Thesis with a discussion prospecting as well future research directions.

Chapter 2

Segmentation techniques for individual mobility data analysis: state-of-the-art

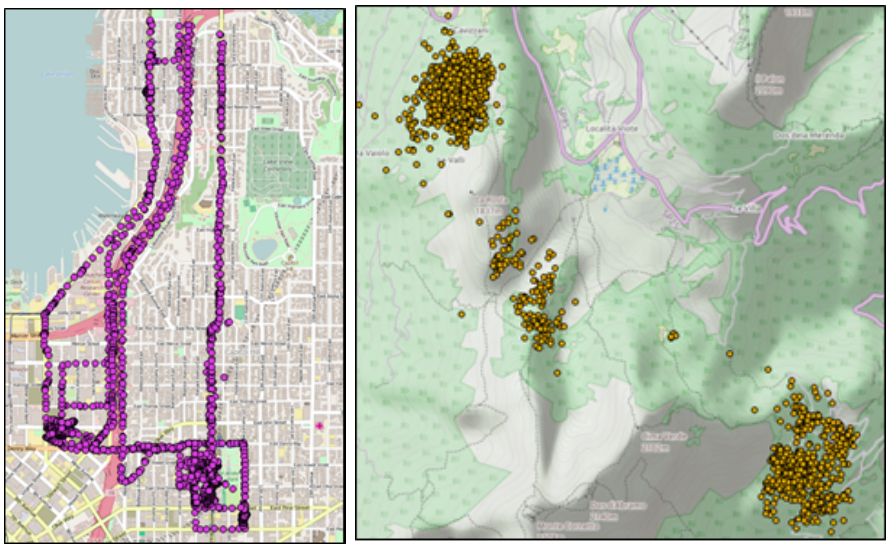
2.1 Overview

This chapter overviews state-of-the-art techniques for the segmentation of spatial trajectories, focusing in particular on the techniques that are employed for trajectory summarization. Data summarization is a major data mining task that can be concisely defined as *compressing data into an informative representation* [20]. That is, summarization discards irrelevant details from data while retaining the most important information. Unlike mere data compression, summarization involves the capability of abstracting content from data.

The rest of the chapter is structured as follows. Section 2.2 sets the context and introduces the basic notions. Section 2.3 introduces the segmentation techniques and the classification in two main categories, called attribute-driven and pattern-driven. Representative techniques for the two categories are next discussed in Section 2.4 and Section 2.5, respectively. Some conclusive considerations are reported in Section 2.6. We recall that most of the materials reported in this chapter has been published in [25].

2.2 Spatial trajectories and beyond

The technological advances in positioning, communication and application services, have dramatically fostered the collection of massive volumes of mobility data. Broadly, mobility data describes the movements of objects in a reference space. These objects can represent entities of very different nature and scale, while



(a) (b)

Figure 2.1: (a) The movement of a person tracked for a few days at high sampling rate (by courtesy of John Krumm, [68]). (b) The movement of an animal (roe deer) tracked for over one year at low and irregular sampling rate [27]

the spaces of interest can be of arbitrary size. Moreover, data can be dramatically voluminous, complex and heterogeneous. In the urban domain, for example, the data on the movement of vehicles possibly combined with additional contextual data, e.g. time series of environmental data, can help identify patterns of interest and/or make predictions, for example on relevant social and natural events, e.g. [22].

Very often mobility data take the form of *spatial trajectories*, namely sequences of coordinated locations reported at consecutive time instants and sampling the object movements in a time interval [99]. In more rigorous terms, given a reference space S, e.g. the Euclidean plane, a spatial trajectory T of length n is a sequence of n spatio-temporal *points*, i.e. $T = \{(p_i, t_i)\}_{i \in [1,n]}$ with $p_i \in S$ and $t_i < t_{i+1} \in Time$. Depending on the application, consecutive points can be equally spaced in time or not. Figure 2.1 shows two examples of spatial trajectories, reported on a planar map as sets of points, describing the movement of two different objects sampled at different frequencies.

Spatial trajectories are, however, complex structured data, difficult to handle efficiently. Even conceptually simple operations such as *range* and *spatio-temporal join* queries are computationally costly, despite the indexing and query processing techniques employed in modern trajectory databases, e.g. [24, 48]. As a result, the efficient access and effective utilization of trajectory data remain an issue.

To address such a problem, a possible strategy is to extract relevant time-dependent information from each individual movement and then use such coarser information in place of spatial trajectories. This strategy is especially appealing when the knowledge granule is not the spatio-temporal point itself, but rather the behavior exhibited by the individual in time. If such information can be extracted and then encoded into a compact form, i.e., as a summarized trajectory, then the dataset is much smaller and the access to relevant information potentially simpler. Along this line, a notion that has become popular in recent times is that of *semantic trajectory* [79]. Semantic trajectory has been proposed for the representation of time-varying behavioral information on single individuals. More recent proposals include *symbolic trajectories* [49] and *spatio-textual trajectories* [56]. A common feature of all of these models is that they provide a rich representation of the movement that goes beyond the spatio-temporal characterization. In that sense, such models can be seen as instances of the general concept of summarized trajectory.

This research focuses on the construction of summarized trajectories, independently from the data model chosen for their representation through the use of trajectory segmentation methods [96]. The systematic use of segmentation for trajectory data summarization and representation has been first proposed in [91] as part of a methodological framework aiming at supporting the semantic trajectory discovery process.

2.2.1 Trajectories handling: database vs. knowledge discovery

As emphasized earlier, spatial trajectories are complex to handle. As the movement of an object in space is sampled for long periods and/or at high sampling frequency, the length of a spatial trajectory dramatically increases. Therefore, for large populations of moving objects, the amount of data becomes overwhelming.

One way to deal with large amounts of spatial trajectories is to store such data in a powerful database and use the functionalities of the system to access the data of interest. This is the *database-centric* view. The Moving Object data model is the reference paradigm for the management of databases of spatial trajectories [47]. In essence, a Moving Object database is a database equipped with a set of data types for the representation of spatial trajectories and their efficient manipulation through the use of dedicated operations and spatio-temporal indexes. It remains the fact that efficiency is still an issue while the deployment of this class of technologies, strictly rooted in the notion of spatial trajectory, is still limited. More recently, a number of distributed platforms for the management of big spatio-temporal data have been proposed, e.g. [2].

Opposed to the database-centric view is the *knowledge discovery view*. This different perspective finds a motivation in the fact that the availability of large amounts of trajectories, though complicating data processing, offers the opportunity of extracting valuable information on the time-evolving behavior of the involved objects. In general, the longer the trajectories (and the number of objects), the richer and more accurate the behavioral information that can be extracted from such data, for example on individual mobility patterns.

As an example, consider the case of a geo-social application continuously sampling the location of a large number of individuals equipped with GPS enabled devices, e.g. smart-phones. The amount of data that is continuously collected is huge. An approach to summarize such trajectories is to report only the sequence of places visited by each individual, e.g. working places, restaurants and so on, along with temporal information on the time spent in each of these places. As a result, the information relevant for the application, for example a user profiling application, is preserved while irrelevant details on the position occupied at a certain time can be omitted or handled separately.

A practical application of this concept of trajectory summarization, in which the spatial trajectory is replaced by a sequence of places, can be found in *Google Maps Timeline*, an information service providing registered users with a map representation of the data collected by Google on their personal movement. Figure 2.2 provides a simple visual summary of the data-centric view and knowledge discovery view.

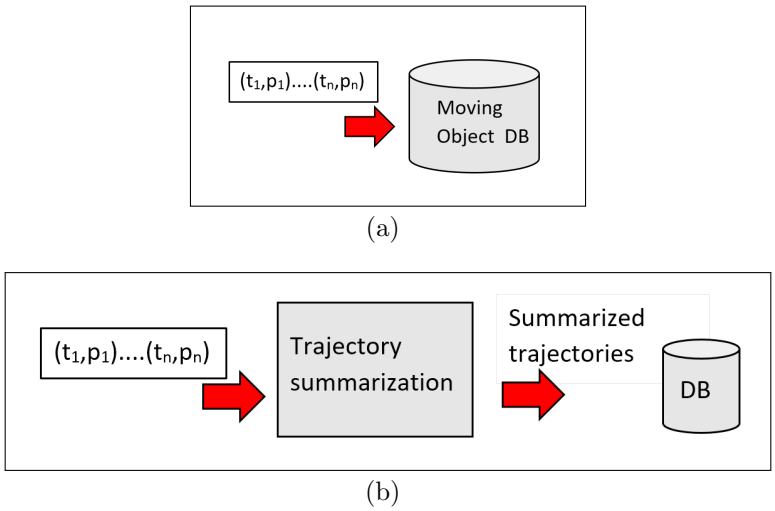


Figure 2.2: (a) Data-centric view: spatial trajectories are stored and accessed through a Moving Object database; b) Knowledge discovery view: spatial trajectories are first summarized, next possibly stored in some database or manipulated through some other application.

2.3 Segmentation techniques for the summarization of trajectories

A segmentation is a partition of a set of objects in a number of homogeneous parts. In general, when applied to spatial trajectories, the segmentation task can regard either the set of full trajectories or the points forming a single trajectory. The latter case is considered in this chapter.

More specifically, the segmentation of a spatial trajectory $T = \{(p_i, t_i)\}_i$ is a series of temporally ordered sub-trajectories, i.e.:

$$S_1 <_t .. <_t S_k \text{ with } T = \bigcup_{i=1}^{k} S_i$$

where $<_t$ denotes the relation of temporal ordering between sub-trajectories. The first point of every segment is called *breakpoint*. Segments can be annotated, for example, with a label indicating the characterizing feature of the segment, or be associated with a representative point. Figure 2.3 shows an example of segmentation of a spatial trajectory consisting of labeled segments.

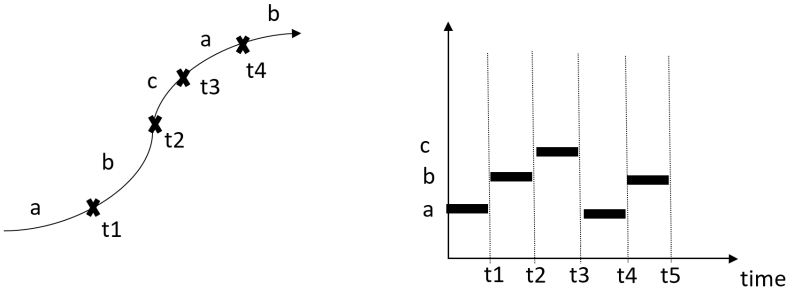


Figure 2.3: Segmentation of a spatial trajectory: the breakpoints along the input trajectory identify the begin/end of segments. Every segment corresponds to some property that holds for the whole duration of the segment

Abstractly, the segmentation task on a trajectory T can be expressed as a function $f(T, C) = \{S_1, .., S_k\}$ where C is the homogeneity criterion and $S_1, .., S_k$ are the k segments (or breakpoints) in which the trajectory is split. Often, the number k of segments is not known in advance. Key notion in segmentation is that of homogeneity criterion. There are two main interpretations of this concept of homogeneity:

- Homogeneity is defined with respect to selected properties of the movement (*movement attributes*) that can be derived from the geometric properties of

spatial trajectories. Movement attributes are, for example, speed, heading, curvature. A segment is homogeneous if the conditions specified on such attributes are satisfied by the points of the segment. We refer to the class of segmentation methods which rely on this notion of homogeneity as *attribute-driven.*

- Homogeneity is defined with respect to the activity performed by the object in the corresponding time interval. We think of activities as behavioral patterns that can be extracted from spatial trajectories using knowledge discovery techniques. Example activities are residing in a region or migrating from one residence to another residence. A segment is thus homogeneous if it can be associated with a certain activity. We refer to the corresponding class of segmentation techniques as *pattern-driven.*

The taxonomy in Figure 2.4 reports the general classification of the segmentation techniques. In the next sections, the attribute-driven and pattern-driven segmentation techniques are explored.

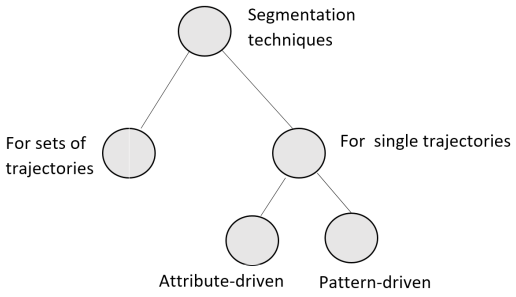


Figure 2.4: General taxonomy for segmentation techniques

2.4 Attribute-driven segmentation

This class of techniques has its roots mainly in computational geometry [44, 8], though it is also inspired by work on time series. Thus, we will provide some background on time series segmentation in order to highlight similarities and differences with prior work.

2.4.1 Time series segmentation

A time series is an arbitrarily long sequence of correlated numeric values $r_1, .., r_n$ with $r_i \in \mathcal{R}$ typically collected from measurements made at uniformly spaced time

instants[33], e.g. the series of temperature measurement over a period of time. A time series of length n is commonly modeled as a point in a n-dimensional space [61]. Therefore an operation such as the Euclidean distance between two time series of equal length $T = r_1, .., r_n$ and $T' = r'_1, .., r'_n$ can be expressed as the distance between two n-dimensional points. Unfortunately, distance-based operations, such as similarity-based search, performed over a large number of long sequences, can be extremely inefficient. Moreover the notion itself of distance in a high-dimensional space is problematic because of the curse of dimensionality [13]. Therefore segmentation is often adopted as a key approach in order to reduce the dimensionality [61].

The segmentation task splits a time series into a number of sub-sequences and replaces each sub-sequence with an approximated representation based on a *segment model* [62]. For example, a segment can be represented by a single numeric value, e.g. a median data point in the sub-sequence.

The *segmentation algorithm* determines the breakpoints along the sequence based on the input parameters, typically the number of segments to be searched for. The problem of finding the breakpoints in the sequence is commonly framed as an optimization problem, specifically, given a time series of length n compute $k << n$ segments so as to minimize the error based on some error function. The error function can be for example the Euclidean distance from the approximating curve or point [86]. Another method for finding breakpoints in time series suggests using a sliding window technique and a likelihood estimation in order to identify the moments of significant changes [46, 94].

Spatial trajectories vs. time series.

In the data mining literature, spatial trajectories are often equated to multivariate time series, namely the X-coordinate and Y-coordinate of the spatial trajectory form the components of the multivariate series [65]. However, the two notions of time series and spatial trajectories are slightly different especially with respect to the role of time and segmentation. In particular, the key feature of time series is the sequential structure of correlated data, while the time attribute is simply a property inducing a total order over a discrete set of values. Moreover, the goal of the segmentation task is to reduce the length of the time series to a predefined number of data points. By contrast, in spatial trajectories, the temporal information is the basis for the distinction between *discrete* and *continuous* movement. Whenever the movement is continuous, the missing points between two consecutive samples can be estimated by interpolation and a number of additional properties can be computed with a certain accuracy, such as trajectory curvature, speed and velocity. That is, a spatial trajectory can be seen as a sequence of data points with associated number of time-varying scalar and vector functions. Such properties are

those utilized by the techniques for attribute-based segmentation, discussed next.

2.4.2 Attribute-driven segmentation of spatial trajectories

The problem can be informally formulated as the problem of partitioning a spatial trajectory in a *minimum number of segments* in such a way that the movement inside each segment is nearly uniform with respect to some condition on movement attributes (referred to as segmentation criteria). The number of resulting segments is commonly unspecified a priori, while the segmentation criteria can be grouped in three classes: *monotone, non-monotone* and *application specific*. These criteria are described in what follows. The extended taxonomy for segmentation techniques is shown in Figure 2.5.

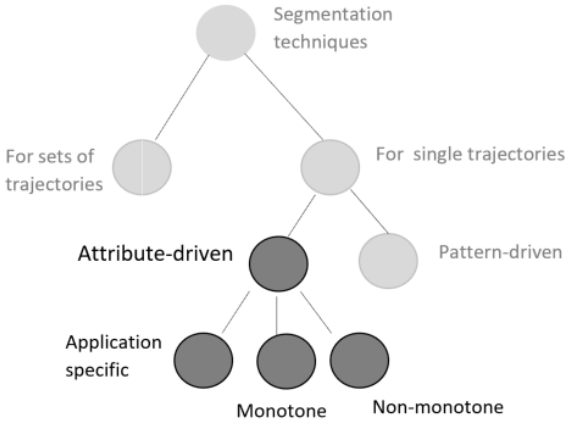


Figure 2.5: Taxonomy refinement: attribute-driven segmentation

Application specific criteria for segmentation

The term "application specific" signifies those segmentation criteria that are defined on ad-hoc basis. For example, in [93], the problem is to split a spatial trajectory in a minimum number of segments approximating a linear movement at a constant speed while decreasing the effect of noise. This type of segmentations is designed to address a particular and specific application, which makes it unable to be adapted for wider purposes.

Segmentation based on monotone criteria

This class of segmentation techniques is described in [16], it consists of specifying criteria and split the spatial trajectory into as few number of segments as possible, in a way that the criteria hold for any attribute within each segment of

the segmentation. A criterion is defined as a constraint on a movement attribute that has to be satisfied by all of the points in the segment. For example, a criterion can require the difference Δ_{speed} between the maximum and minimum speed value in the segment to be lower than $50km/h$. A criterion is *monotone* if for any sub-trajectory τ, it holds that if τ satisfies the criterion, then any sub-trajectory $\tau' \subseteq \tau$ also satisfies the criterion. In other terms, if we have any sub-trajectory for which the criterion is not satisfied, then extending the sub-trajectory cannot satisfy the criterion too. There are many examples of criteria that are monotone. For example the criterion $\Delta_{speed} < 50k/h$ is monotone while $\Delta_{speed} > 50k/h$ is not. Multiple criteria can be combined to form a set of criteria. It is shown that given a set of monotone criteria the optimal segmentation satisfying such criteria can be computed efficiently, in nearly linear time with respect to the trajectory length.

This segmentation approach presents important limitations, shown when an experience of application/evaluation was performed in [17]. It consists of a case study regarding the movement of a group of birds monitored during their Spring migration from the Netherlands to Siberia. The ultimate goal of the application is to discriminate the bird activities, called *states*, such as flying and resting using a proper set of segmentation criteria. For example, the status of *flight* is informally defined as "little variation in heading, speed at least 20 km/h for at least 5 hours". Unfortunately, the constraint on the minimum time duration (e.g., at least 5 hours) is not monotone. Moreover, real data often contain outliers that cause incorrect breakpoints. That is, there exist segmentation criteria of practical interest that cannot be expressed using the proposed framework.

Segmentation based on non-monotone criteria

The third and more recent class of segmentation criteria tries to overcome the above mentioned issues by extending the theoretical framework to encompass non-monotone criteria.[8]. It is shown that computing the minimum number of segments satisfying non-monotone criteria is computationally hard under the assumption of continuous movement. Under certain circumstances, however, and only for certain non-monotone criteria, it is shown that the optimal segmentation can be computed efficiently in polynomial-time. In particular two criteria are found for which the problem is tractable. One such criterion requires that the difference between the maximum and minimum value of an attribute inside the segment does not exceed a given value, while allowing a certain percentage of outliers. The second criterion requires that on each segment the standard deviation of the attribute is below a certain threshold [8]. Interestingly, both these criteria are related to noise handling. However, a complete characterization of the non-monotone criteria for which efficient segmentation methods can be found is still open.

2.5 Pattern-driven segmentation techniques

In this second class of segmentation techniques, the homogeneity is considered with respect to the movement behaviors or the patterns presented in a spatial trajectory. They may be less general from a theoretical perspective but they have more flexibility with respect to the application needs. This class of techniques is based on machine learning methods. Principally, segmentation methods based on clustering are employed for the detection of specific behaviors (or patterns). Such patterns can be either defined with respect to specific domains, e.g. human mobility, or be generic, such as the *stop-and-move* pattern. In particular, a stop-and-move pattern is an abstraction of the mobility behavior of an object that repeatedly stays for some time in a relatively small region (i.e. a stop) before moving to other regions. Objects exhibiting such a behavior include, for example, animals tracked while foraging or migrating, and the eyes gaze exploring a visual scene. We will provide in what follows some background information on clustering methods to spot light on temporal aspects and approaches that can be used for the pattern-driven segmentation.

2.5.1 Background on clustering of temporally annotated data

The clustering task subdivides a set of objects in groups of similar objects based on some criteria of similarity. Clustering, therefore, consists of grouping the similar objects together, hence one cluster's elements are expected to be similar with each other and different from other clusters' ones. The similarity is defined based on the application needs.[12] Thus a broad range of clustering techniques have been proposed, based on diverse paradigms such as partitional, hierarchical, density-based, grid-based clustering [51]. Classically, all of these methods apply to unordered sets.

Two popular techniques are conceptually relevant to the trajectory segmentation problem, *K-means* and *DBSCAN*. *K-means* is representative of the class of partitional clustering techniques. Accordingly, the clustering problem is to find a partition of k clusters, with k being an input parameter, that optimizes a properly defined clustering quality function. In contrast, *DBSCAN* generates clusters based on density criteria. The number of clusters is unknown, while the two input parameters ϵ and N specify the density requirements, i.e. ϵ, the radius of a neighborhood, and N the minimum number of data points in a ϵ-neighborhood, respectively [35]. Unlike *K-means*, *DBSCAN* is robust with respect to noise, therefore the presence of unstructured points does not have a disruptive impact on the partitioning of the set as in *K-means*.

Time-aware clustering.

An important class of techniques addresses the problem of clustering temporally annotated data points. Two major categories of such techniques regard the clustering of spatio-temporal events, and the clustering of stream data, respectively. In particular:

- *Spatio-temporal events* are uncorrelated spatio-temporal points, e.g. seismic events. The goal of the clustering task is to group together the events that are close in space and time. This problem is commonly approached introducing some notion of spatio-temporal distance. For example, *ST-DBSCAN*, a temporal extension of *DBSCAN*, introduces two metrics, one for space and one non-spatial while the temporal aspect is supported by filtering the points and retaining only the temporal neighbors and their corresponding spatial values [14]. Accordingly, the key notion of ϵ-neighborhood for a point p is slightly modified, i.e. the ϵ-neighborhood contains N points whose temporal distance from p does not exceed a threshold value.

- *Data streams* are unbounded sequences of data of arbitrary type. The clustering task groups the data points as they arrive, under the memory constraints imposed by the data streaming context. In this case, the temporal information can be utilized, for example, to limit the amount of data to cluster, e.g. the clusters in the last year, last month, last week, as in [1]. A different strategy is to use the temporal information to award recent and evolving clusters against oldest and stable clusters. For example, in *DensStream* [18] every point is given a weight that decreases exponentially with time via a fading function $f(t) = 2^{-\lambda t}$, where $\lambda > 0$ is a system defined parameter. As the cumulative weight of a dense group of points exceeds a threshold value then such a group becomes a cluster. Next, if no new point is added, the weight will decay gradually until the group of points becomes noise and eventually the memory space is released for new clusters.

A common feature of all of these techniques is that the resulting clusters are not temporally separated. Therefore such methods are conceptually unsuitable for the segmentation of spatial trajectories.

Clustering techniques for pattern-driven segmentation

We turn to consider the clustering techniques that return temporally disjoint clusters. This form of segmentation is referred to as *clustering-based*. It includes three major classes of techniques: *heuristic-driven, density-based, partitional*. Accordingly, the general taxonomy introduced earlier can be refined as shown in Figure 2.6. An orthogonal and important distinction to bear in mind during the analysis

of these categories is between the techniques that are sensitive to noise and those that are not. This distinction will be discussed at the end of this section.

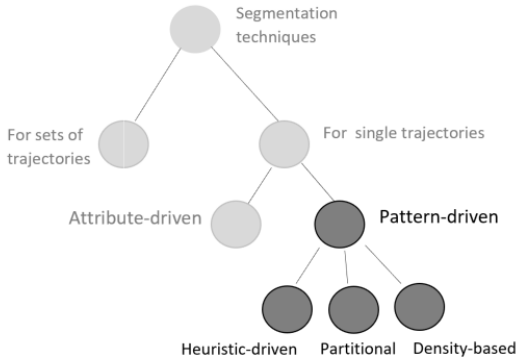


Figure 2.6: Taxonomy refinement: pattern-driven segmentation

Heuristic-driven segmentation. This class encompasses methods relying on simple heuristics. An early approach, which can be taken as representative of the class, is proposed in [60] for the analysis of human mobility. The application goal is to detect the places of interest visited by an individual, where a place of interest is a region where the individual stays for a minimum time. The idea is to compare every new point that arrives with the centroid of the current cluster. If far away from the centroid, the point is considered to belong to a different cluster. Finally, the clusters which represent places of interest are filtered out based on the duration threshold. Along this line, another popular technique has been proposed by Yu Zheng et al. [98]. A major problem with this class of approaches is the lack of a more general framework providing guarantees.

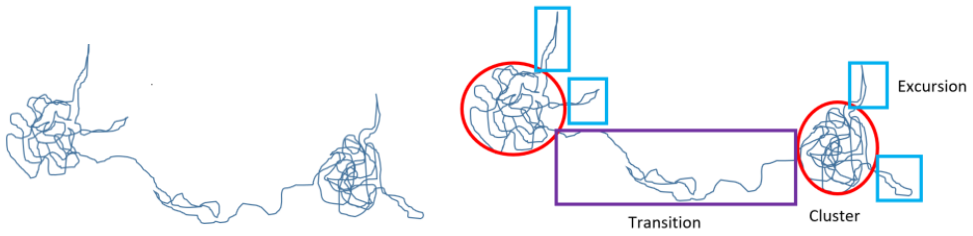
Based on partitional clustering. These approaches are inspired by *K-means*. For example, *Warped K-means* is an algorithm that, like *K-means*, allocates data points based on the analysis of the effects on a quality function, caused by moving a sample from its current cluster to a potentially better one [71]. Because of the ordering constraint, the first half of points in cluster j are only allowed to move to cluster $j-1$, and, respectively, the last half of data points are only allowed to move to cluster $j+1$. A point will be reallocated if and only if the operation is beneficial for the quality of clustering. This process is iterated until no transfers are performed. A more recent approach, taking inspiration from *Warp K-means*, employs the notion of density in place of the quality function, without requiring in input the parameter k [82]. In practice, a breakpoint is created as the density of the data points in proximity of the current cluster representative is less than a threshold value. As relying on *K-means*, these techniques are sensitive to noise.

Figure 2.7: CB-SMoT clustering: the output is a sequence of temporally ordered stops. Every stop is associated with a time interval [78]

Based on density-based clustering. This class of approaches relies on *DBSCAN*. One of the earliest and probably most representative approaches is *CB-SMoT* (Clustering-Based Stops and Moves of Trajectories). This is a technique proposed for the extraction of *stops* from spatial trajectories. Stops are defined as segments of minimum length and minimum duration, along which the speed is thus limited [78]. An example from the original article and illustrating a sequence of stops is shown in Figure 2.7. *CB-SMoT* inherits key concepts from *DBSCAN*, yet such concepts are formulated in slightly different terms. In particular the notion of ϵ-neighborhood (neighborhood of radius ϵ) is defined along the linear representation of the trajectory, Moreover the constraint on the minimum number of points for a ϵ- neighborhood to be dense is reformulated as temporal constraint on the minimum duration of the sub-trajectory. Whenever the condition on the minimum speed for a minimum time is no longer satisfied, the cluster-segment is broken. This happens irrespective of the fact that the variation can be only temporary and thus could represent noise. That is, unlike *DBSCAN*, this technique is not robust against noise.

Noise handling. As it is seen, noise sensitivity is an issue common to all of those techniques. To deal with this problem, a common strategy is to introduce some additional constraints, for example on the maximum number of noise points that can be tolerated before a breakpoint is added, as in e.g. [60]. Such a parameter is, however, hard to set, especially whenever the sampling intervals are irregular and the clustering is to be applied to a large number of trajectories. As a result, these techniques are highly time consuming and little effective in practice.

A first attempt to deal in more systematic way with the problem of noise in clustering-based segmentation is represented by *SeqScan* [27]. The detailed analysis and validation of SeqScan is the major topic of this Thesis. SeqScan distinguishes two classes of outliers: the points indicating a temporary absence from the cluster (*excursion point*) and the points representing a definitive departure from the cluster towards another cluster (*transition point*). The measure of *presence* is an estimate of the time spent inside the cluster excluding the periods of absence. The segmentation algorithm determines the sequence of clusters along with the classified outliers. Figure 2.8 shows the components, i.e. the clusters, the



(a) (b)

Figure 2.8: (a) Spatial trajectory; (b) SeqScan clustering: clusters, excursion points, transition points

excursion points, and the transition points, for an example trajectory.

2.6 Concluding remarks

In this chapter we have seen two major categories of approaches addressing the problem of segmenting spatial trajectories. While the potential applications for these techniques are similar, the methodologies underneath are substantially different. Both directions present pros and cons. Attribute-driven methodologies are built on solid theoretical frameworks. Yet, the characterization and generalization of the segmentation criteria, the handling of outliers and the scalability of the segmentation algorithms still represent important challenges. Probably less general from a theoretical perspective but more flexible, with respect to the application needs, are the cluster-based segmentation techniques, in particular the SeqScan technique that will be analyzed in the next chapter. For the sake of completeness, it is worth mentioning that segmentation methods are also investigated in other scientific domains, especially in animal ecology for the detection of behavioral states, e.g. foraging, resting, migrating. Such methods substantially employee three different methodologies relying on: movement attribute analysis (in the sense discussed above), time-series analysis and state-space models, e.g., hidden Markov models, respectively [31]. There is, however, a substantial difference between these methods and SeqScan. Firstly, the patterns we can discover are *generic* (vs. *behavioral*) [30], namely can be used as building blocks for the detection of different types of behaviors. Secondly, our technique applies to sequences of temporally annotated data points defined in a metric space, thus the solution is not confined to the analysis of the physical movement in an Euclidean space. In this sense, the solution is of more general interest and can be employed in diverse spaces. In summary, we recall that we have started contrasting the database and the knowledge discovery view, as two orthogonal strategies for the effective management of large

volumes of trajectory data. In the light of these last considerations, it appears that these two views are, in reality, converging towards the definition of a unifying framework enabling the efficient access to content-rich trajectory datasets and as an alternative to parallel and distributed spatio-temporal databases.

Chapter 3

SeqScan: model, properties and application

3.1 Overview

In this chapter we analyze the *SeqScan* technique. We recall that SeqScan is a cluster-based segmentation technique for the discovery of stop-and-move patterns, defined in previous work [27], thus outside the scope of this Thesis. The problem encountered with this method is the lack of a formal ground enabling the in-depth analysis of the properties of the method. As a consequence, key questions such as which patterns can be detected and why, how robust and accurate the method is, are still unsolved. In this chapter we present a formal model of SeqScan and discuss key properties of the model. Moreover, to highlight the application potential of the technique, we investigate how the framework can be used to facilitate the discovery of additional mobility patterns, which we call *derived*, such as recursive movement patterns [72]. Specifically, we propose a technique for location periodicity detection. The aspects more related to the quality of clustering, will be instead discussed in the next chapter. Most of the materials reported in this Chapter and in Chapter 4 has been submitted for publication in [25].

3.2 SeqScan in a nutshell

SeqScan partitions a spatial trajectory in a sequence of segments based on spatial density and temporal criteria. The result is a set of temporally separated clusters interleaved by sub-sequences of unclustered points. A major novelty is the proposal of an outlier or noise model based on the distinction between intra-cluster (local noise) and inter-cluster noise (transition).

In broad terms, *noise* consists of data points that do not fit into the clustering

structure and that for such a reason can be considered as diverging [35, 51]. As an example, consider the spatial trajectory in Figure 3.1.(a). The trajectory is split in sub-trajectories as shown in Figure 3.1.(b). Some of these sub-trajectories are clusters. We can see that there are two classes of unclustered points: the points representing a transition and the points that do not belong to either a cluster or a transition. In the latter case, the noise indicates, in essence, a temporary departure or *absence* from the cluster. This notion of temporary absence can be exemplified by an individual leaving the area where he/she resides, for example for a travel, and then returning back after a period of time. Note that such an absence cannot be qualified as a transition because the individual in reality does not change residence but simply leaves it for a period. Put in different terms, absence points represent a form of noise that is somehow local to clusters in contrast to transition, which represents an inter-cluster noise. To emphasize this characteristic, we will refer to this form of noise as *local noise*.

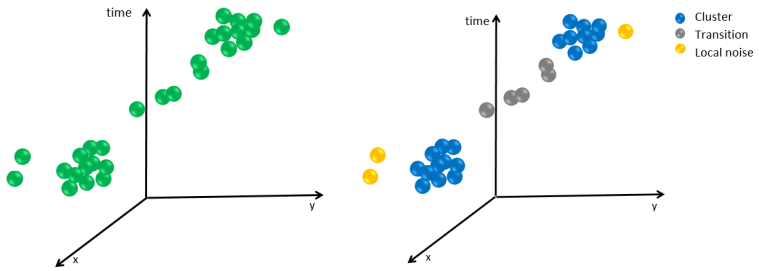


Figure 3.1: (a) A spatial trajectory in the spatio-temporal coordinate system. The dots sample the time-varying location of an object moving in the Euclidean plane. (b) Segmentation of the trajectory. The segmentation consists of two clusters with a few local noise points, and one transition

Local noise can have an application-dependent meaning. For example, in the field of animal ecology, biologists use the term "excursion" to characterize the temporary absence from the home-range where the animals reside [29]. In summary, the local noise represents a semantically meaningful ingredient of many dynamic phenomena and as such cannot be neglected.

It is worth noting that in case of no local noise - the individual simply moves from one residence to another residence - the cluster-based segmentation is fairly straightforward. It is sufficient to aggregate the points of the sequence in clusters using a DBSCAN-like technique [35]. Next, once a cluster is created, the first point in the sequence that cannot be added to such a cluster determines the break of the segment. The problem with this solution is that whenever the unclustered

points do not have an univocal interpretation, e.g. can represent both a temporary absence and the definitive departure from the current cluster, such points cannot be correctly classified until the actual destination or, put in other form, the object's behavior is known. A different approach is thus needed. That is the problem SeqScan addresses.

3.3 The SeqScan segmentation model: a formal framework

This section presents the cluster-based segmentation model. Preliminarily, we briefly review the main concepts underlying density-based clustering [67], specifically the DBSCAN cluster model [35], which provides the ground for the proposed framework, and introduce the basic terminology. Then the cluster and stay region models are defined, to further define the sequence of stay regions.

3.3.1 Preliminaries

The DBSCAN cluster model

Consider a database P of points in a metric space. Let $d_s(.)$ be the distance function, e.g. the Euclidean distance, and $\epsilon \in \mathbb{R}$ (i.e. the distance threshold), and $K \in \mathbb{N}$ (i.e. the minimum number of points in a cluster) the input parameters. The DBSCAN cluster model is built on the following definitions [35]:

- The ϵ-*neighborhood* of $p \in P$, denoted $N_\epsilon(p)$, is the subset of points that are within distance ϵ from p, i.e. $N_\epsilon(p) = \{p_i \in P, d_s(p, p_i) \leq \epsilon\}$.
- A point p is *core point* if its ϵ-neighborhood contains at least K points, i.e. $|N_\epsilon(p)| \geq K$. A point that is not a core point but belongs to the neighborhood of a core point is a *border point*.
- A point p is *directly density-reachable* from q if q is a core point and $p \in N_\epsilon(q)$.
- Two points p and q are *density reachable* if there is a chain of points $p_1, .., p_n$, $p_1 = p, p_n = q$ such that p_{i+1} is directly reachable from p_i.
- Points p and q are *density connected* if there exists a core point o such that both p and q are density-reachable by o.
- A *cluster* C with respect to ϵ and K is a non-empty subset of points satisfying the following conditions:

[1] $\forall$ p, q: if $p \in$ C and q is density-reachable from p, then $q \in$ C (Maximality)

[2] $\forall$ p, $q \in$ C: p is density-connected to q (Connectivity)

- A point p is a *noise* if it is neither a core point nor a border point. This implies that noise does not belong to any cluster.

An example illustrating the DBSCAN concepts is shown in Figure 3.2. Apart a few peculiar situations[1], the result of DBSCAN is independent of the order in which the points of the database are visited [35]. Therefore, the algorithm cannot detect sequences of clusters based on some ordering relation over points.

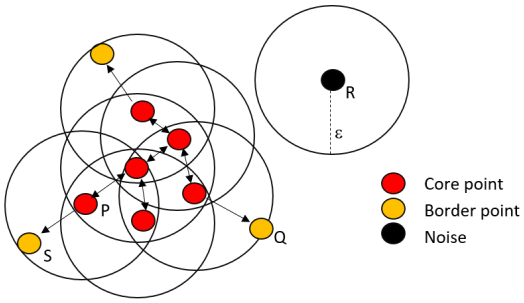


Figure 3.2: DBSCAN cluster, parameters: $\epsilon, K = 4$. P is a core point; Q is a border point because contained in the ϵ-neighborhood of P, though not core point; Q is directly reachable from P; Q and S are density connected; R is a noise point.

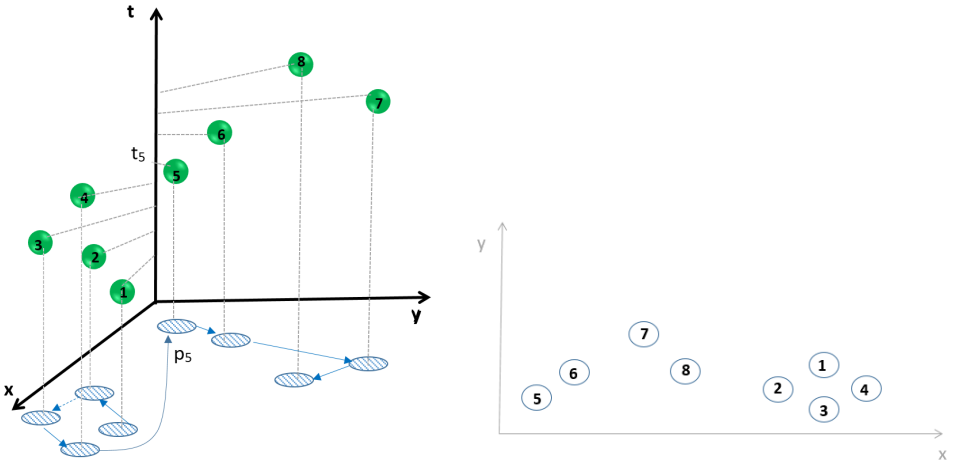
Basic notation

Consider now a *trajectory* T of n points $(p_1, t_1), .., (p_n, t_n)$. For the sake of simplicity, the trajectory is represented by the interval of indices $[1, n]$, with i indicating the i-esim point (p_i, t_i). Besides the spatial distance $d_s(i, j)$, consider the function $d_t(i, j)$ computing the temporal distance between points i and j, respectively. A *sub-trajectory* of T is represented by a connected interval $[i, j] \subseteq T$, while a *segment* is represented by a possibly disconnected interval - a union set of disjoint connected intervals. Intuitively, a segment is a sub-trajectory that can have "holes". As shown in Example 3.1, a trajectory is visually represented as sequence of numbered circles indicating the indexed points on the plane. The basic notation used throughout the paper is summarized in Table 3.1.

[1]The property is not satisfied by border points. Yet, that is marginal for our work

Table 3.1: Notation

I=[i,j]	Sequence of indexed points
$\bigcup_i I_i$	Segment
p_j, t_j	Spatial point, temporal annotation
$d_s(i,j)$	Spatial distance
$d_t(i,j)$	Temporal distance
K, ϵ	DBSCAN parameters
$N_\epsilon(p)$	Neighbourhood of point p of radius ϵ
δ	Presence threshold
S	Cluster/stay region
$\mathcal{P}(S)$	The value of presence in S
$\mathcal{D}(S)$	Duration of S
$\mathcal{N}(S)$	Noise local to S
$S_i \rightarrow .. \rightarrow S_j$	Sequence of stay regions
r_j	Transition
$\mid$	Spatial separation predicate
$\widehat{S}$	Minimal Stay Region in S



(a) Spatio-temporal representation of the trajectory T

(b) Simplified graphical notation of T

Figure 3.3: Graphical notation for trajectories

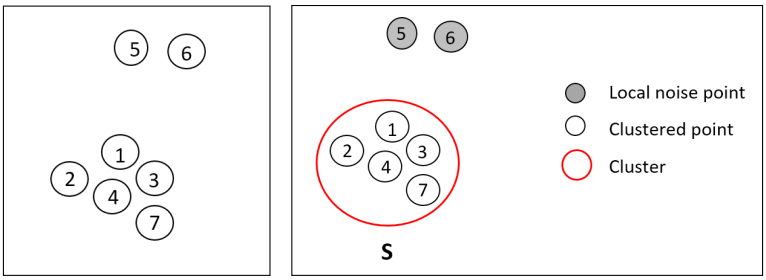
Example 3.1. Figure 3.3.(a) illustrates the trajectory T=[1,8] in the space-time coordinate system along with the projection on plane and time of the points, i.e. p_i, t_i. The subset $[2, 7]$ of T is a sub-trajectory; the subset $[2, 3] \cup [5, 7]$ a segment. Figure 3.3.(b) shows the simplified visual representation that will be used next.

3.3.2 Cluster and stay region model

We start introducing the model for the single stay region, next the sequence of stay regions. Basically, a stay region is a cluster satisfying a temporal constraint on *minimum presence*.

Definition 3.1 (Cluster)**.** *Given a trajectory T, a cluster $S \subseteq T$ is a segment consisting of points that, projected on the reference space, constitutes a DBSCAN cluster (w.r.t. density parameters ϵ, K). Moreover, if the segment is bounded by points i and j then the DBSCAN cluster includes the projection of i and j. The set difference $[i, j] \setminus S$ specifies the corresponding local noise, denoted $\mathcal{N}(S)$.*

Example 3.2. Consider the trajectory T=[1,7] in Figure 3.4.(a). If we run DBSCAN on the set of spatial points $p_1, .., p_7$ with parameters K=4 and ϵ sufficiently small, we obtain that the set $\{p_1, p_2, p_3, p_4, p_7\}$ forms a DBSCAN cluster. Thus the segment $S = [1, 4] \cup [7, 7]$ is a cluster in our model, while the points 5 and 6 are local noise.



(a) (b)

Figure 3.4: Cluster and local noise. (a) The trajectory consists of seven points $[1, 7]$. For brevity, we omit the plane axes. (b) The trajectory contains the cluster S (w.r.t. $K = 4, \epsilon$ small) and local noise. $S = [1, 4] \cup [7, 7]$ (open circles) while the points 5 and 6 (grey shaded circles) are local noise.

A cluster S has a duration $\mathcal{D}(S)$, and a presence $\mathcal{P}(S)$. The duration $\mathcal{D}(S)$ is simply the temporal distance between the first and last point of the segment, namely $d_t(i, j)$. The presence $\mathcal{P}(S)$ estimates the residence time in the cluster, with exclusion of the absence periods, i.e. local noise. Specifically, the presence of S is defined as the cumulative duration of the connected intervals in S.

Definition 3.2 (Presence)**.** *Given a cluster* $S = S_1 \cup .. \cup S_m$*, with* S_i *a connected interval,* $\mathcal{P}(S)$ *is defined as follows:*

$$\mathcal{P}(S) = \Sigma_{i=1}^{m} \mathcal{D}(S_i) \tag{3.1}$$

This definition of presence relies on the following assumption, that if two consecutive points $i, i+1$ are both members of the cluster then the whole time between t_i and t_{i+1} is assumed to be spent "inside" the cluster, or more precisely inside the spatial region where the object resides. Conversely, if at least one of the points does not belong to the cluster, then the whole time between t_i and t_{i+1} is spent outside the cluster. We postulate that with points relatively close in time - as we assume - the presence value can provide a good estimation of the time spent inside the residence.

Example 3.3. Consider again Figure 3.4. Assume for simplicity that the time interval between consecutive points is 1 time unit. We can see that $\mathcal{P}(S) = 3$.

proposition 3.1. The presence in a cluster S ranges in the interval $[0, \mathcal{D}(S)]$.

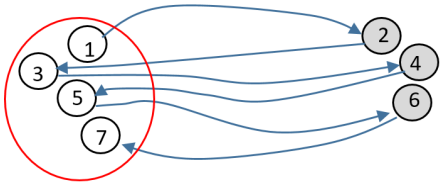


Figure 3.5: Null presence in a cluster. The trajectory [1,7] contains the cluster $S = \{1, 3, 5, 7\}$ and local noise $\{2, 4, 6\}$ (w.r.t. K=4, ϵ). The oriented lines show the movement flow. Following the definition, it holds that $\mathcal{P}(S) = 0$

The presence upper-bound $\mathcal{D}(S)$ follows straightforwardly from the definition of presence. Less obvious is the fact that the presence can be 0. This property can be shown through an Example. Consider the trajectory [1,7] in Figure 3.5 containing one cluster and a few local noise points. The cluster is the segment $[1, 1] \cup [3, 3] \cup [5, 5] \cup [7, 7]$. By applying the definition, the presence in the cluster is 0.

Intuitively, the presence measures how assiduously a residence is frequented. The presence is thus maximal when the object never leaves the cluster region before departing from it and 0 when the object moves back and forth to/from the region without residing steadily in it. To enable the filtering of clusters based on the presence value, the notion of stay region is introduced.

Definition 3.3 (Stay region)**.** *A stay region S is a cluster satisfying the* minimum presence constraint *defined as:*

$$\mathcal{P}(S) \geq \delta \tag{3.2}$$

where $\delta \geq 0$ is the presence threshold*.*

Example 3.4. The cluster S in Figure 3.4 is a stay region if $\delta \leq 3$. Conversely, if $\delta > 3$, the time spent in S is not enough for the cluster to represent an object's residence.

proposition 3.2 (Monotonicity)**.** If a minimum presence constraint is satisfied by stay region S_1, then it is also satisfied by any stay region S_2 such that $S_1 \subset S_2$.

3.3.3 Sequence of stay regions

After presenting the notion of single stay region, we turn to consider sequences of stay regions. We begin providing a refined definition of cluster-based segmentation, next we discuss relevant properties.

Definition 3.4 (Cluster-based segmentation)**.** *Let $T = [1, n]$ be a trajectory and K, ϵ, δ the segmentation parameters. A segmentation is a set of disjoint segments $\{S_1, .., S_m\} \cup \{r_0, .., r_m\}$, covering the whole trajectory where:*

- *$S_1, .., S_m$ are stay regions satisfying the following conditions:*
  - *Stay regions are temporally separated*
  - *Stay regions are of maximal length, i.e. any point that can be included in the cluster without compromising the temporal separation of the clusters is included.*
- *$r_0, .., r_m$ are possibly empty* transitions*. Transitions do not include any point that can be added to stay regions.*

A segmentation can be represented as follows:

$$\xrightarrow{r_0} S_1 \xrightarrow{r_1} S_2 .. \xrightarrow{r_{m-1}} S_m \xrightarrow{r_m}$$

The sequence of stay regions is referred to as path*.*

Example 3.5. Figure 3.6 shows a segmentation comprising two stay regions w.r.t. $k = 4, \epsilon, \delta = 0$:

$$S_1 \xrightarrow{r_1} S_2 \xrightarrow{r_2}$$

where: $S_1 = [1, 4], S_2 = [5, 7] \cup [11, 11]$, $r_1 = \emptyset, r_2 = [12, 13]$. It can be noticed that the stay regions are maximal. The object moves straightforwardly from S_1 to S_2, next it experiences a period of absence from S_2 and finally leaves it.

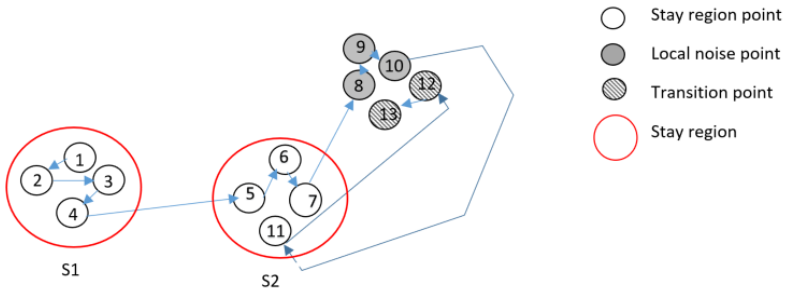


Figure 3.6: Trajectory segmentation: the segmentation consists of two stay regions S_1, S_2 w.r.t. $K = 4, \epsilon, \delta = 0$. It can be seen that the stay regions S_1 and S_2 are both maximal.

Analysis of the spatial separation property

Following the definition of segmentation, the stay regions in a path are temporally separated. A straightforward question is whether the stay regions are also separated in space. Indeed spatial separation is a natural property of "conventional" clusters. In our model, however, the situation is slightly different because the trajectory describes an evolving phenomenon, therefore the notion of spatial separation requires a more precise characterization that takes time into account.

We begin with a general definition of spatial separation between two stay regions, next we discuss some key properties that are at the basis of the algorithm presented next. Basically, a stay region S_2 is spatially separated by S_1 if no point exists in either S_2 or in the corresponding local noise that is reachable - in the DBSCAN sense - from a point of S_1. In other words, while residing in S_2 the moving object has to be sufficiently far from S_1 even during the periods of absence. Formally:

Definition 3.5 (Spatial separation)**.** *Let S_1, S_2 be two stay regions in a path, non-necessarily consecutive, denoted for simplicity (i.e. we omit the transitions)*

$$S_1 \rightarrow .. \rightarrow S_2$$

We say that S_2 is spatially separated by S_1, denoted $S_2|S_1$, if no point $p \in S_2 \cup \mathcal{N}(S_2)$ belongs to the ϵ-neighborhood of any core point $q \in S_1$ (i.e. p is not reachable from S_1). Two stay regions that are not spatial separated are said to overlap. $\diamond$

Example 3.6. Figure 3.7 illustrates the segmentation of the trajectory $T = [1, 10]$ in two stay regions $S1, S2$ connected through the transition $\{6\}$. S2 is separated from S1 and viceversa.

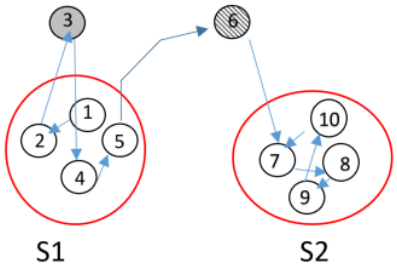


Figure 3.7: Two spatially separated stay regions $S1, S2$ with $S_1 = [1, 2] \cup [4, 5]$ and $S_2 = [7, 10]$. Point 3 is a noise point local to S_1, point 6 a transition point

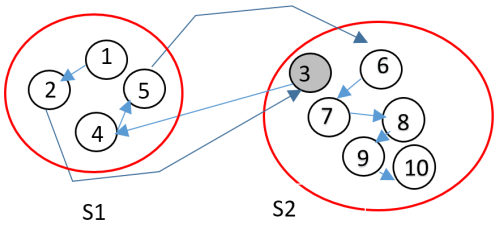


Figure 3.8: Asymmetry of the spatial separation relationship: S_2 is spatially separated from S_1 while, by contrast, S_1 is not separated from S_2 because point 3 - local noise for S_1- is reachable from S_2

The spatial separation property is directional. Formally;

proposition 3.3. The relationship of spatial separation between stay regions is asymmetric.

$$S_j|S_i \nRightarrow S_i|S_j \tag{3.3}$$

Example 3.7. Figure 3.8 shows the segmentation of the trajectory $T = [1, 10]$: $S_1 \overset{\emptyset}{\rightarrow} S_2$. Following Definition 3.5, it holds that S_2 is spatially separated from S_1 because no point exists in S_2 and in the corresponding local noise $\mathcal{N}(S_2)$ that is *reachable* from S_1. In contrast, S_1 is not separated by S_2 because the point 3, belonging to the local noise $\mathcal{N}(S_1)$, is reachable from S_2.

The next concept is that of Minimal Stay Region (MSR). This concept is at the basis of the cluster-based segmentation algorithm. In essence, the MSR is the "seed" of a stay region. Formally:

Definition 3.6 (Minimal Stay Region)**.** *The MSR of a stay region S (w.r.t. $\epsilon, K.\delta$), denoted $\widehat{S}$, is the stay region of minimal length contained in S that is created for first in time.*

Example 3.8. The trajectory [1,7] in Figure 3.9 is a stay region (w.r.t. k=4 $\epsilon, \delta = 0$) and the MSR is the connected interval [2,5]. Note that the sub-trajectory [1,4] is not a stay region because it does not contain a cluster of at least 4 elements. The cluster is only created at time t_5. At that time, the cluster of minimal length is [2,5].

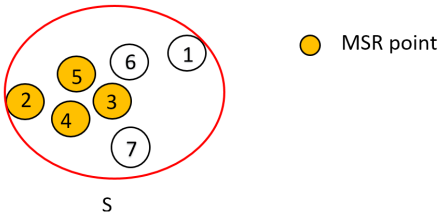


Figure 3.9: Minimal Stay Region (MSR). The stay region S=[1,7] contains the MSR [2,5] (yellow points)

From the definition of segmentation, we can derive the following Theorem stating a necessary condition on the spatial separation of consecutive stay regions:

Theorem 3.1. *For any pair of consecutive stay regions* $S_i \xrightarrow{r} S_{i+1}$*, it holds that the MSR* $\widehat{S}_{i+1}$ *is spatially separated from* S_i*, namely* $\widehat{S}_{i+1}|S_i$.

Proof. The proof is by contradiction. Suppose that $\widehat{S}_{i+1}$ is not separated from S_i. Based on Definition 3.5, at least one point exists in the set $\widehat{S}_{i+1} \cup \mathcal{N}(\widehat{S}_{i+1})$ that is directly reachable from S_i. Let j be the point with lowest index reachable from S_i and consider the segment $S_i \cup \{j\}$. We see that: a) no other cluster can exist in between S_i and j because j is the lowest index; b) the segment satisfies the minimum presence constraint based on Proposition 3.2. Thus $S_i \cup \{j\}$ is a stay region in the path. However, that contradicts the assumption that S_i is a cluster of maximal length. Therefore $\widehat{S}_{i+1}$ must be separated from the previous stay region, which is what we wanted to demonstrate.

The next two corollaries provide a motivation for specific mobility behaviors that can be observed in a trajectory. In particular Corollary 3.1.1 states that two consecutive stay regions can spatially overlap for some time. The intuition is that when the object leaves a residence for another residence, after a while it can start moving back gradually to the previous region. Corollary 3.1.2 states that two non-consecutive stay regions, even identical in space, but frequented in different periods, are treated as two different stay regions. In other terms, non-consecutive stay regions can fully overlap. Formally:

Corollary 3.1.1. *Let $S_i \xrightarrow{r} S_{i+1}$ be two consecutive stay regions. The points in S_{i+1} following in time the minimal stay region $\widehat{S}_{i+1}$ may be not spatially separated from S_i. We refer to this property as weak spatial separation.*

Example 3.9. Figure 3.11 shows two weakly separated stay regions $S_1 \rightarrow S_2$ with S1=[1,4] and S2=[5,14] (w.r.t. $K = 4, \epsilon, \delta$). While the MSR of S_2 is separated from S_1, it can be seen that the individual moves progressively from S_2 towards the previous residence to finally overlap S_2.

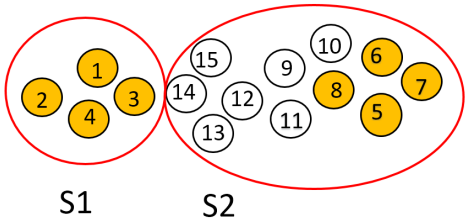


Figure 3.10: Weakly separated stay regions. The yellow circles highlight the points of the MSRs. While the MSRs are separated, the points of S_2 are reachable from S_1

Corollary 3.1.2. *Two non-consecutive stay regions can overlap*

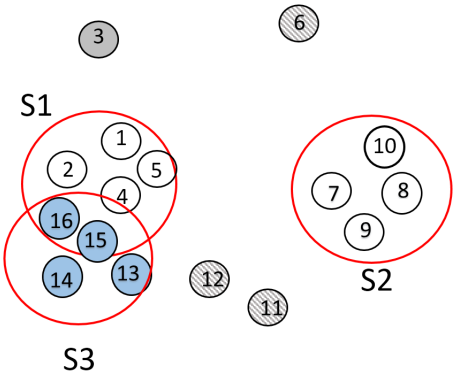


Figure 3.11: The segmentation includes three stay regions $S_1 \xrightarrow{\{6\}} S_2 \xrightarrow{\{11,12\}} S_3$. Point 3 is a noise point local to S_1. For the sake of readability the points in S_3 are coloured. Of these stay regions, S_1 and S_3 overlap.

Example 3.10. Figure 3.11 illustrates the case of three stay regions $S_1 \xrightarrow{\{6\}} S_2 \xrightarrow{\{11,12\}} S_3$, where S_1 and S_3 overlap.

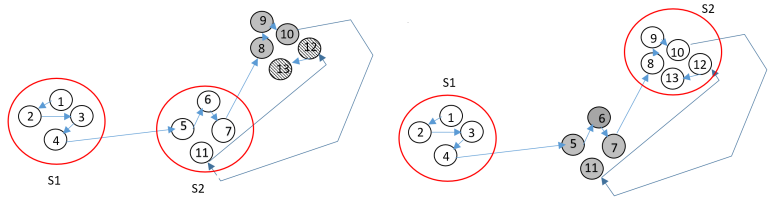
Finally, the following theorem reformulates the notion of path in more specific terms. It follows straightforwardly from the above results.

Theorem 3.2. *A path in a trajectory (w.r.t. ϵ, K, δ) is a sequence of temporally separated stay regions of maximal length and possibly pairwise weakly spatially separated*

3.4 Segmenting trajectories: the SeqScan algorithm

Based on the specified model, we now describe the algorithm for the cluster-based segmentation of a trajectory. Preliminarily, we show an important property, that the segmentation of a trajectory may be not unique. This raises the question of which segmentation to select.

3.4.1 The choice of the segmentation



(a) (b)

Figure 3.12: Two different segmentations for the same trajectory (w.r.t. $K = 4, \delta = 0, \epsilon$) both including two stay regions. The segmentations differ in the composition of S_2. In (a), S_2 is the stay region detected for first; in (b), S_2 is the region containing the highest number of points.

proposition 3.4. The segmentation of a trajectory, based on a common set of parameters ϵ, k, δ, may be not unique

Example 3.11. Figure 3.12 shows two different segmentations for the same trajectory both including two stay regions S_1, S_2. Such regions are however selected based on different criteria. In 3.12.(a) S_2 is the stay region that is created for first after S_1. In 3.12.(b), S_2 is the stay region with the highest number of points after S_1.

Algorithm 1 SeqScan

```
procedure SeqScan( In: T = [1, n], ε, K, δ; Out: stayRegionsSet)
    stayRegionsSet ← ∅
    C ← ∅                                   ▷ the context of the active stay region
    P ← ∅                                   ▷ pool of points for MSR search
    i ← 1                                   ▷ current scan index
    activeStayRegion ← ∅
    while i ≤ n do
        Insert(C,i)                         ▷ incremental clustering of C
        if canExpand(activeStayRegion, i, C) then   ▷ if i can be added
            P ← ∅                           ▷ pool reset
        else
            Insert(P,i)                     ▷ incremental clustering of P
            nextStayRegion ← findMSR(P)     ▷ find the next minimal stay region
            if nextStayRegion ≠ ∅ then
                stayRegionsSet ← addStayRegion (activeStayRegion)
                activeStayRegion ← nextStayRegion
                C ← P                       ▷ set the context for the new MSR
                P ← ∅                       ▷ reset of the pool
        i ← i+1
```

Since the segmentation may be not unique some criterion is to be put in place to select the stay regions of the sequence. We choose the following criterion: at each step the next stay region is the one whose MSR appears for first. This criterion has an intuitive explanation: *an object resides in a region until another attractive residence is found*. The resulting path is called hereinafter *first path*.

3.4.2 First path discovery: the segmentation algorithm

We now revisit the early version of the algorithm [27] called SeqScan at the light of the previous results. We first provide an overview, next we detail key aspects.

Overview of SeqScan

The SeqScan algorithm extracts from the trajectory T a sequence of stay regions, based on the three input parameters K, ϵ, δ. The noise points can then be obtained by difference from T and straightforwardly classified in transition and local noise points. The algorithm scans the trajectory, iterating through the following phases: i) Find a Minimal Stay Region. Such MRS becomes the *active* stay region. ii) Expand the active stay region. iii) Close the active stay region. Once closed, a

stay region cannot be expanded anymore. More specifically:

(i) Search: the algorithm runs the DBSCAN algorithm on the spatial projection of the input sequence (i.e. spatial points). The clustering algorithm processes the points in the temporal order, progressively aggregating points in clusters. The cluster that for first in time satisfies the minimum presence constraint determines the new MSR S_i. S_i becomes the *active* stay region.

(ii) Expand: the active stay region is expanded. The question at this stage is how to determine the end of the expansion and thus the break of the segment. We recall that, in the stay region model, a point that is not reachable from a cluster can indicate either a temporary absence, or a transition or be an element of a more recent stay region. Therefore such a point cannot be correctly classified, until the movement evolution is known.The proposed solution is detailed next.

(iii) Close: the active stay region is deactivated, or closed, when a more recent MRS is found. Such MRS becomes the new active stay region S_{i+1}. A closed stay region is simply a stay region in its final shape.

Detailed algorithm.

The pseudo-code is reported in Algorithm 1. At each step, the algorithm tries first to expand the active stay region S_j, and if that is not possible, tries to create a new MSR S_{j+1}. To perform such operations, the algorithm maintains two different sets of points that are clustered incrementally using the Incremental DBSCAN algorithm [34]. These sets are called C and P, respectively. C represents the *Context* of the active stay region S_j, namely the set of points that at each step can be used for the expansion of the cluster. Such points follow the previous stay region in the sequence, thus C is separated from S_{j-1}. The set P is the *Pool* of points following in time the active stay region and representing the space where to search for the next MSR. Accordingly P is temporally separated from S_j. When a new point is added to either C or P, the set is clusterized incrementally using the Incremental DBSCAN technique[34]. The processing of the input point i is thus as follows:

- i is first added to the Context C. If the point can be added to the current cluster, then the stay region is prolonged to include the point. Next the Pool is reset to the emptyset.
- if i cannot be added to the active stay region, then i is added to the Pool P. If a MSR can be created out of P then such a MSR becomes the new active

stay region S_{i+1}. Accordingly, S_i is closed, the Pool P becomes the Context for S_{i+1} and P is reset to the emptyset.

The run-time complexity of SeqScan is that of Incremental DBSCAN, i.e. $O(n^2)$ [34, 39].

Example 3.12. We illustrate the algorithm through an Example focusing on the expansion part. Consider the trajectory T=[1,13] in Figure 3.13.(a). The clustering parameters are set to: K=4, $\delta = 0$, ϵ sufficiently small. We analyze the expansion of the active stay region S_1, starting from the MSR depicted in the Figure. For every subsequent point, we report the change of state defined by the triple: active stay region, C, P.

[1] State: $S_1 = [1,1] \cup [3,5]$, C= [1,5], P= $\emptyset$

[2] Read point: 6. The point cannot be added to the active stay region, thus the state is: S_1, C= [1,6], P=[6,6]

[3] Read point: 7. The point cannot be added to the active stay region. State: S_1, C= [1,7], P=[6,7]

[4] Read point 8. The point can be added to the active stay region. State: $S_1 = [1,1] \cup [3,5] \cup [8,8]$, C=[1,8], P=$\emptyset$.

[5] Read point 9. The point cannot be added to the active stay region. State: S_1, C=[1,9], P=[9,9]

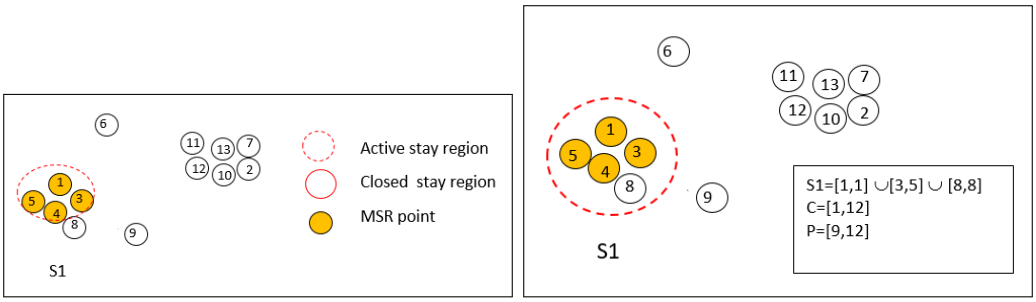
[6] Read point 10, as above. State: S_1, C=[1,10], P=[9,10]

[7] Read point 11, as above. State: S_1, C=[1,11], P=[9,11]

[8] Read point 12, as above. State: S_1, C=[1,12], P=[9,12].

[9] Read point 13. The point cannot be added to the active stay region. However, the insertion of the point in P, i.e. P=[9,13], generates a new stay region. Accordingly S_1 is closed, $S_2 = [10,13]$, C=[9,13], P=$\emptyset$. The scan is terminated

The final stay regions are thus $S_1 = [1,1] \cup [3,5] \cup [8,8]$ and $S_2 = [10,13]$. The noise points can then be classified. Points 2, 6,7 fall in the temporal extent [1,8] of S_1 thus are local noise; point 9 is a transition point



(a) State at step 5. Next input point: 6

(b) State at step 12. Next input point: 13

(c) State at step 13. Next the trajectory terminates and S_2 is closed

Figure 3.13: SeqScan steps; (a) S_1 becomes the active stay region; (b) expansion of S_1; (c) the creation of S_2

The following Theorem states that SeqScan solves the problem of extracting the first path.

Theorem 3.3. *Algorithm 1 computes the first path, if any path exists in the input trajectory.*

Proof. We need to prove that the resulting stay regions are maximal, temporally separated and pairwise weakly spatially separated. Moreover, every stay region is the first to be created after the previous one. The reasoning is as follows. The algorithm builds a stay region by expanding the first MSR that is encountered. Further the stay region is expanded based on the Context C that includes all of the points following the end of the previous stay region, thus all those that potentially can be added to the stay region. The stay region is thus maximal and the first to be created. The stay regions are weakly spatially separated because when a MSR is found in the Pool P, it cannot contain points "close" to the previous stay region (otherwise such point would be added to that stay region). Yet, once the MSR is created, the subsequent points that are added to the active stay region during the expansion phase can be located even in close proximity with the preceding

stay region. The stay regions are also temporally separated because stay regions are created and next expanded from the two sets P and C that by definition are separated from the previous stay region.

3.4.3 Parameter analysis

SeqScan requires in input the presence threshold parameter δ. This threshold somehow constraints the temporal granularity of the stay regions in the path. Hence, the analysis of the relationship between the number of stay regions and δ helps to determine the desired level of temporal granularity. We report in more detail this analysis, although resulting from previous work [55], because the functionality is incorporated in the MigrO system detailed later.

Consider a trajectory $T = [1, n]$ and let $f_T : [0, \mathcal{D}(T)] \rightarrow \mathbb{N}$ be the function yielding the number of clusters in T, for values of $\delta \in [0, \mathcal{D}(T)]$. The other parameters K, ϵ are fixed. Issa [55] shows that the number of stay regions remains constant for values of δ ranging in properly defined intervals. That is, the function f_T has a step-wise behavior. The constructive definition of f_T is presented in Algorithm 2.This algorithm runs SeqScan multiple times with different values of the parameter δ until the number of resulting stay regions is 0. The presence threshold is initially set to $\delta = 0$. We illustrate the iterative process as follows. After the first run, SeqScan returns a sequence S of stay regions, based on which, the minimum value of presence in the respective MSRs is computed. Such a value, say v_1, forms the upper bound of the first interval $I_1 = [0, v_1]$. For values of δ falling in such interval, the number of stay regions is $|S|$. Next, SeqScan is run with $\delta = v_1 + \theta$ (with $\theta > 0$ is a small constant used to handle the discontinuity) to possibly determine the second interval I_2. The process iterates until the terminating condition is met.

Algorithm 2 Computing the function f_T

procedure f_T(In: $T = [1, n], \epsilon, K$, step; Out: SetOfPairs)
  min $\leftarrow$ *0, SetOfPairs* $\leftarrow \emptyset$
  $\delta \leftarrow 0$
  while *min* $\neq$ -1 **do**
    $SeqScan(T, \epsilon, K, \delta, S = [S_1, .., S_m])$
    if $S \neq \emptyset$ **then**
      min $\leftarrow$ *minumum presence of minimal stay regions from* $[\widehat{S}_1, .., \widehat{S}_m]$
      Add ([δ*, min], m) to SetOfPairs*
      $\delta \leftarrow min + \theta$
    else *min=-1*

3.5 Discovering derived patterns

We argue that the SeqScan framework can facilitate the discovery of additional mobility patterns. We refer to the patterns that are built on the notion of stay region as *derived*. In this section, we present an approach to the discovery of recursive movement patterns [72, 11]. This type of movement can be broadly defined as *repeated visitation to the same particular locations in a systematic manner* [11]. "Office", "work", are examples of locations. Detecting which and how those locations are frequented can reveal important features of the object behavior. We focus, in particular, on the detection of locations that are frequented regularly on a periodic basis. To avoid possible conflicts with the terminology used in the rest of the chapter, we call *zones* the 'locations' visited by an object. We split the problem in two sub-problems:

- To discover the zones $z_i, .., z_j$
- To discover the periodic zones, i.e., $Zone(t) = Zone(t + \mathcal{T})$, where $\mathcal{T}$ is the period and $Zone(t)$ the zone where the object is located at time t.

Coherently with the work presented so far, the overarching assumption is that the behavior may contain noise.

3.5.1 Discovery of zones

Periodic zones are commonly modeled as spatial clusters [72, 19]. To extract these clusters, Cao et al. use DBSCAN [19], while the MoveMine project [74] the Worton method [90]. All of these clustering techniques ignore time, i.e. are *spatial-only*. We argue that the use of spatial-only clustering, in place of spatio-temporal clustering, may result into a rough approximation that impacts the quality of the analysis. To begin, we observe that objects spend some time inside a zone. Therefore, if the location is sampled at a frequency that is relatively high with respect to the time spent inside a zone, a visit results in a dense set of sample points. While such a set can be straightforwardly modeled as a cluster, it is highly unlikely that the clusters at different times are perfectly identical. That suggests modeling a frequented location as *set* of clusters. This change of perspective has important implications. For example, it can be shown that spatial-only clustering can generate clusters including points that in reality represent noise. An example can better explain the problem. Consider the points of a trajectory projected on plane and assume that these points form the clusters C_1, C_2 as shown in Figure 3.14.(a). These clusters appear compact, i.e., no noise. In reality, there are 4 agglomerates, i.e. spatio-temporal clusters, along with a few noise points. These agglomerates are pairwise close to each other, thus, once projected on plane, collapse in a unique

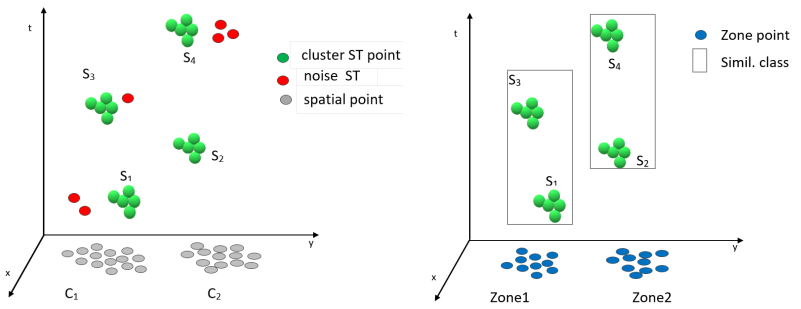


Figure 3.14: Spatial-only vs. spatio-temporal clustering. (a) Spatial-only clusters as projection of spatio-temporal clusters with noise; b) Spatio-temporal clusters grouped in similarity classes.

spatial cluster, which absorbs the noise points. As a result the noise information is lost. Clearly, that can be avoided taking time into account. Another important reason for using spatio-temporal clusters, in particular stay regions, in place of spatial-only clusters, is that the individual movement can be given a discrete sequential representation, which can be more easily manipulated.

In the light of these considerations, we propose the following approach: to extract the sequence of stay regions and then group together the stay regions that are close to each other, based on a properly defined notion of proximity, to finally associate each such groups a *zone*. We recall that, based on Corollary 3.1.2, two stay regions can *overlap*. We need, however, a more restrictive and noise independent notion of cluster proximity, therefore we introduce the concept of *spatial similarity* (of clusters). Similar stay regions form a *zone*. In the following we detail the process starting from the notions of spatial similarity and zone.

Spatial similarity of clusters. We say that two stay regions S_1, S_2 are spatially similar if there is at least one core point of S_1 that is *directly reachable* from S_2 (in the DBSCAN sense), or viceversa, there is at least one core point of S_2 that is *directly reachable* from S_1. We quantify the spatial similarity as the maximum percentage of core points that are directly reachable from the core points of the other set. More formally: let $O_1(O_2)$ be the set of core points in stay region $S_1(S_2)$ falling in the ϵ-neighborhood of some core point in S_2 (S_1). We define the function of spatial similarity $Sim(S_1, S_2)$ as follows:

$$Sim(S_1, S_2) = \max\{\frac{|O_1|}{|S_1|}, \frac{|O_2|}{|S_2|}\} \tag{3.4}$$

The two stay regions are spatially similar if $Sim(S_1, S_2) \geq \psi$, where $\psi \in [0, 1]$ is

the similarity threshold. Note that this notion of similarity is not affected by the relative size of clusters, in other terms the similarity can be 1, even though the clusters are of very different size.

Zones. Similar stay regions can be grouped in classes. The *similarity class* C_i of the stay region S_i is defined recursively as follows: C_i contains S_i and all of the regions similar to at least one region of the class. A similarity class is maximal, that is every stay region that can be added to the class, belongs to the class. Moreover, the relation of spatial similarity induces a partition over the set of stay regions. For every similarity class we define the corresponding zone as follows. The similarity class $C_i = \{S_i, ..S_j\}$ associated with a zone Z_i is the projection on space $\pi_{x,y}$ of the union set of the stay regions in C_i

$$Z_i = \pi_{x,y}(\bigcup_{S_j \in C_i} S_j) \tag{3.5}$$

A nice property that follows from the above definitions is that a zone is itself a cluster. However, in contrast with spatial-only clusters, zones do not contain noise. An example is shown in Figure 3.14.(b). The 4 stay regions $S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4$ of the Figure can be pairwise grouped to form 2 zones, *Zone1* and *Zone2*. Note that, at this stage, we can rule out the local noise because it is not relevant for the problem at hand. Thus the trajectory can be rewritten as sequence of temporally annotated zones for example using the formalism of symbolic trajectories [49]: $(I_1, Zone1)(I_2, Zone2)(I_3, Zone1)(I_4, Zone2)$. As a result, we obtain a simple and compact representation of the trajectory.

3.5.2 Discovery of periodically visited zones

The zones may be be visited periodically. The period, however, is not known, thus can range between 1 and $n/2$ where n is the length of trajectory, moreover it can be imprecise. To our knowledge the only approaches dealing with the periodicity of locations are built on spatial-only clustering [72, 19, 74]. For the analysis of location periodicity, we propose to leverage the symbolic representation of the trajectory obtained at the previous step, map it onto a time series and use a technique for the periodicity analysis of symbolic time series with noise. Specifically, we utilize the WARP technique [32][2].

WARP. Consider a time series $T = [x_0, x_1, .., x_{n-1}]$ of n elements. The key idea

[2]The implementation of the algorithm has been kindly provided by M. Elfeky, co-author of WARP [32]. Another implementation is available on: `//github.com/Serafim-End/periodicity-research`

underlying WARP is that if we shift the time series of p positions and compare the original time series to the shifted version, we find that the time series are very similar, if p is a candidate period [32]. Therefore the greater the number of matching symbols, the greater the accuracy of the period. For every possible period $p \in [1, n/2]$, WARP computes the similarity between the time series T and the time series shifted p positions, $T^{(p)}$ using Dynamic Time Warping (DTW) as similarity metric. The underlying distance function measures whether two symbols are identical or not. The value of the distance $DTW(T, T^{(p)})$ ranges between 0 and $n - p$, where 0 indicates that the time series of length p is perfectly periodic. The confidence of a period p is defined as: $1 - \frac{DTW(T,T^{(p)})}{n-p}$.

Process. We obtain the time series from the symbolic trajectory resulting from the previous phase as follows. First, we specify the temporal resolution of the time series, e.g. week, year. Next, for every temporally annotated zone, we create a sequence of repeating symbols, one per time unit. We repeat the same process with the *transitions*, which are assigned a system-defined symbol. The resulting time series is given an input to WARP, which returns the candidate periods for every period p along with the confidence value. The periods with high confidence are those of interest. An application of the method will be shown in the next chapter.

3.6 Summary

In summary, in this chapter we have provided a rigorous specification of the model for clustering-based segmentation with local noise and we have analyzed the properties of such a model, such as the notion of spatial separation of clusters that in the dynamic context requires a specific characterization. Also, we have proposed a novel technique for the discovery of periodically visited locations, which leverages *SeqScan* and the novel concept of cluster spatial similarity. We contrast our solution with state-of-the-art methods, using real data. The approach is shown to be effective, simpler to use, and more informative than these methods even in case of periodicity with noise.

Chapter 4

Evaluation of SeqScan

4.1 Overview

This chapter focuses on the evaluation of the effectiveness of the SeqScan framework. In particular the goal is to contrast SeqScan clustering with ground truth. This form of evaluation is commonly called *external validation*. Three series of experiments are presented, each covering a different aspect:

[1] We start showing the main patterns that can be detected by SeqScan. These main patterns are represented by manually defined synthetic trajectories. SeqScan is run on these trajectories, while the segmentation is evaluated with respect to the actual behavior, displayed visually, on a *qualitative* basis.

[2] We present a systematic approach to the *quantitative* evaluation of the cluster-based segmentation against ground truth. The ground truth consists of labeled trajectories simulating the movement of animals ("animal dataset"). The evaluation is conducted using blind experiments in collaboration with domain experts.

[3] Based on the animal dataset, we analyze the sensitivity of the analytical framework to key internal and external parameters.

[4] We present the evaluation of the technique for the discovery of periodic locations. We compare our approach with the solution developed in the MoveMine project [74], based on a real dataset.

[5] Finally, we discuss the performance of Seqscan.

The experiments discussed in this chapter are conducted using the MigrO environment, a plug-in written in Python for the Quantum GIS system[1] providing

[1] http://www.qgis.org

a number of functionalities for SeqScan-based analysis, such as visualization tools [28]. The MigrO system is described in Chapter6. For the statistical tests, we use the R system[2]. The hardware platform consists of a computer equipped with Intel i7-4700MQ, 2.40GHz processor with 8 GBytes of main memory.

The chapter organization is as the following: we will first describe the synthetic trajectories representing the main patterns, and show how SeqScan works on these trajectories, then the experiments done for the external evaluation are described in details in Section 4.3, further investigations concerning the sensitivity analysis are explained in Section 4.4. In Section 4.5 the experimental evaluation of the derived pattern discovery technique will be presented, and the performance evaluation of the algorithm will be analyzed in Section 4.6. And finally the discussion and conclusions are mentioned in Section 4.7.

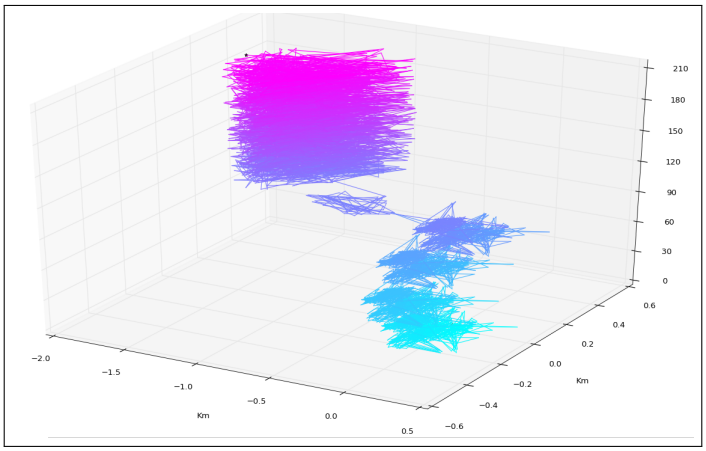
4.2 SeqScan at work

We illustrate the three major typologies of patterns that can be detected by SeqScan: the linear ordering of clusters with local noise and transitions, weakly separated consecutive clusters, and overlapping non-consecutive clusters. For clarity, we illustrate the trajectories both in the spatio-temporal coordinate system and projected on plane. The trajectories have been generated manually.

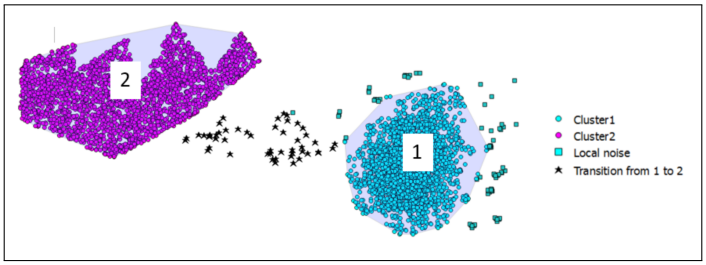
Pattern 1: linear ordering of stay regions with local noise.

The trajectory displayed in Figure 4.1.(a) shows the behavior of a moving object in space and time. We can see that there are two temporally separated clusters, of irregular shape, and that such clusters are also clearly spatially separated. The first cluster (following the temporal order) has associated some local noise while the second group is compact both in space and time. Moreover, there is a clear transition between the two stay regions. SeqScan is run with parameters $\delta = 0$, $\epsilon = 70m, K = 20$. The result is shown in Figure 4.1.(b). It can be seen that the segmentation correctly identifies the two clusters, the transition and some local noise associated with one of the clusters.

[2]https://www.r-project.org/



(a) Spatio-temporal representation

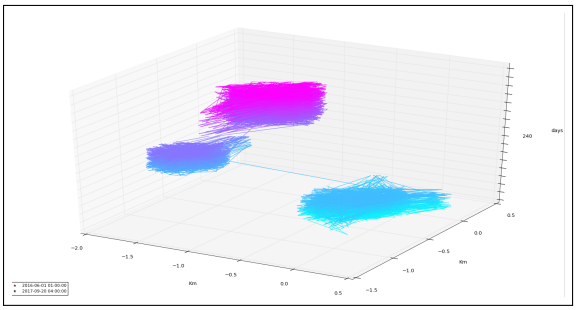


(b) Planar representation of the trajectory segmentation

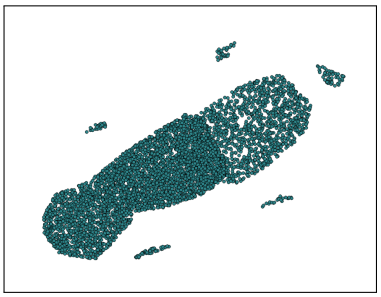
Figure 4.1: Pattern 1: (a) Spatio-temporal representation: the vertical axis measures the temporal distance from the start of the trajectory (day unit), the color gradient the evolution in time, the space units the distance from the starting point. (b) Segmentation: points are classified and displayed using a different symbology; the two stay regions are enclosed in polygons for readability.

Pattern 2: weak separation of consecutive stay regions.

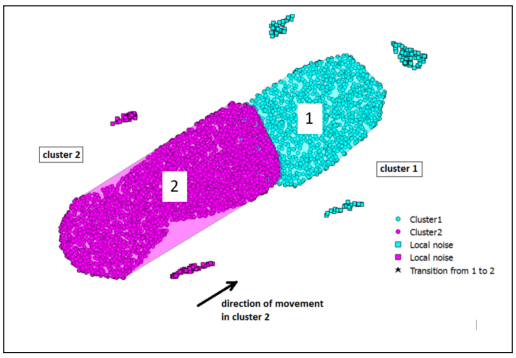
The planar representation of the trajectory displayed in Figure 4.2.(b) contains two clusters that are spatially separated only for a limited period of time. Their movement behavior is represented in space and time Figure4.2.(a). In particular it can be noticed that the object moves back from the second region to the initial region. We run with the same parameters as above, we obtain the two stay regions reported in Figure 4.2.(c). The two stay regions are evidently not disjointed and this is coherent with the fact that consecutive stay regions can be weakly separated, in accordance with Corollary 3.1.1 explained in the previous chapter.



(a) Spatio-temporal representation



(b) Planar representation

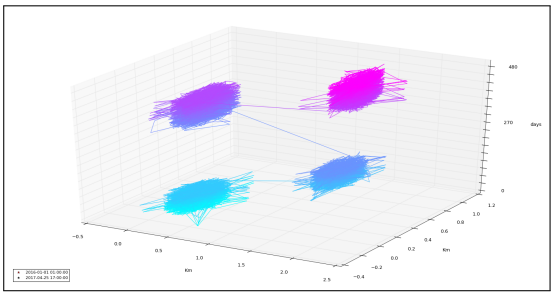


(c) Segmentation: two stay regions and local noise. Details are provided in tabular form

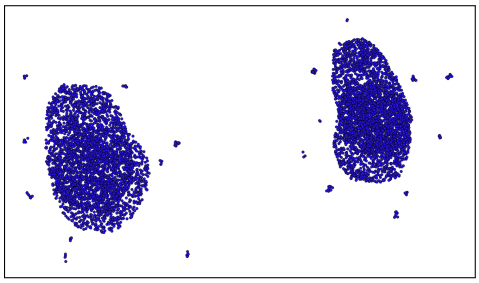
Figure 4.2: Pattern 2: two weakly separated stay regions. The two clusters, enclosed by polygons, are distinct though partially overlapping. The points outside the polygons are local noise. There is no transition.

Pattern 3. Non-spatially separated stay regions.

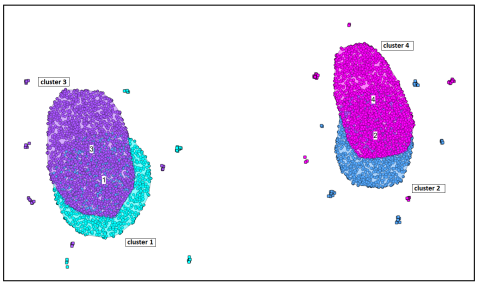
The trajectory in Figure 4.3.(a) exemplifies the case of an object moving back and forth between two regions. The space-time cube shows that there are four clusters and that the clusters that are consecutive are spatially separated. Coherently with Corollary 3.1.2, SeqScan detects the correct sequence of clusters as shown in Figure 4.3.(c).



(a) Trajectory in space-time



(b) Planar representation



(c) Segmentation: the sequence of 4 clusters, each bounded by a polygon and labelled with a progressive id. The points outside the polygons are local noise, no transition

Figure 4.3: Pattern 3. The trajectory of an object moving between two regions of space. There are 4 stay regions; those that are non-consecutive overlap in space

4.3 Evaluation based on external criteria

The next goal is to confront the SeqScan clustering with the ground truth, based on *external indexes* (vs. relative and internal criteria [66]). Generally speaking, the ground truth consists of labeled points where the labels specify the categories the points belong to. Moreover, the ground truth can be either generated by a simulator or consists of real data labeled by domain experts. In the specific context of trajectory data, it is often the case that real trajectory data are irremediably of low quality (e.g. missing points). On the other hand, synthetic datasets can be engineered to match assumptions of the occurrences and properties of meaningful clusters [36]. For a sustainable and fair evaluation, we have thus experienced a different methodology. We use synthetic data as ground truth, yet such data is generated by a simulator grounded on an independent model, specifically designed for the simulation of the animal movement. In addition, we keep the generation of the movement patterns and the subsequent confrontation with the SeqScan clus-

tering independent using blind experiments (see also [43] for a similar approach). The experimental setting, describing the ground truth generation process and the evaluation metrics, and the experiments are detailed in the following.

4.3.1 Experimental setting

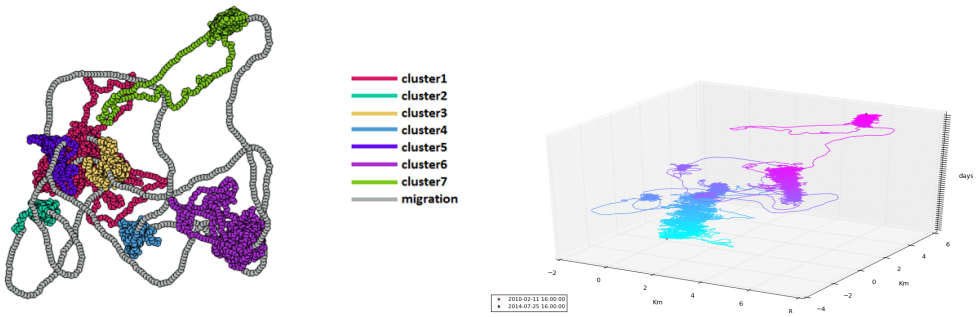
Datasets

We use a dataset of raw synthetic trajectories generated by a simulator of animal movement developed by a team of biologists from E. Mach Foundation (domain experts, hereinafter). The trajectories have been selected from a larger dataset based on the number of clusters, i.e. those containing at least 6 clusters. The raw simulations have been translated in a format suitable for clustering evaluation.

In more detail, the simulated trajectories are sequences of labeled points $T = \{(x_i, y_i, u_i, l_i)\}_{i\in[1,n]}$ where l_i is a label or category (i.e. "residence", "migration", "excursion"), x_i, y_i the spatial coordinates, u_i the time unit. The ground truth is extracted from the simulated trajectories by mapping the ecological concepts onto the concepts of our model while stay regions are given a progressive numeric identifier, and timestamps are assigned to points based on the chosen time interval. The process has been applied to create a dataset of 12 spatial trajectories of 19500 points each, with labeled points indicating stay region, transition, local noise, and time interval of 2 hours. This dataset, called hereinafter "animal dataset", is the ground truth. An Example showing some chunks of a dataset of one of these trajectories is given in Figure4.4 to explain how this dataset is composed. The spatio-temporal and planar representation of two examples trajectories, labeled IND1 and IND14 respectively, is shown in Figure 4.5 and Figure 4.6, respectively. In these figures, local noise points are displayed as cluster points. The planar representation of the 10 other trajectories is reported in Figure 4.7 while summary statistics for the full set are reported in Table 4.1.

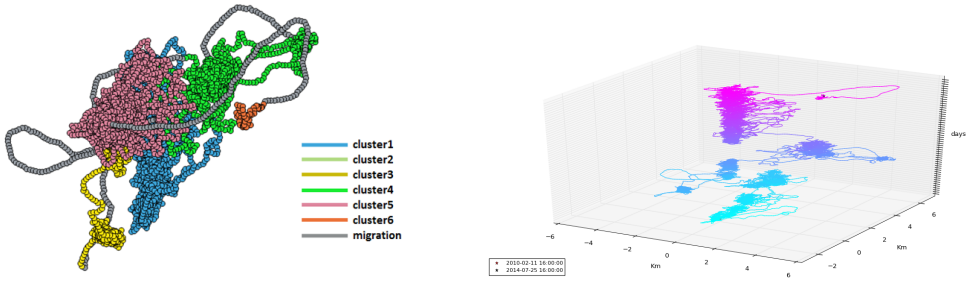
id	x	y	time	behavior
1	2362.56	4439.51	11/02/2010 16:00:00	Resident in S1
2	2439.06	4503.92	11/02/2010 18:00:00	Resident in S1
3	2533.15	4537.78	11/02/2010 20:00:00	Resident in S1
836	3220.27	4186.94	22/04/2010 06:00:00	Excursion from S1
837	3188.01	4092.28	22/04/2010 08:00:00	Excursion from S1
838	3089.64	4110.25	22/04/2010 10:00:00	Resident in S1
7475	3525.72	5905.52	27/10/2011 12:00:00	Resident in S1
7476	3625.1	5894.44	27/10/2011 14:00:00	Migrating
7477	3714.21	5849.04	27/10/2011 16:00:00	Migrating
19500	7788.7	9519.1	25/07/2014 14:00:00	Resident in S7
19501	7870.86	9576.11	25/07/2014 16:00:00	Resident in S7

Figure 4.4: Example showing some chunks of a trajectory ground-truth dataset. The *id* field serves as an identifier for each point in the trajectory, x and y are the spatial coordinates fields, *time* field shows the timestamps of points,time interval is 2 hours between two consecutive points, the ground truth is presented in the *behavior* field. The values of this field can be: Resident to specify that this point was inside a stay region S, Excursion and Migrating.



(a) Planar representation (b) Spatio-temporal representation

Figure 4.5: Ground truth: Trajectory IND1.



(a) Planar representation (b) Spatio-temporal representation

Figure 4.6: Ground truth: Trajectory IND14.

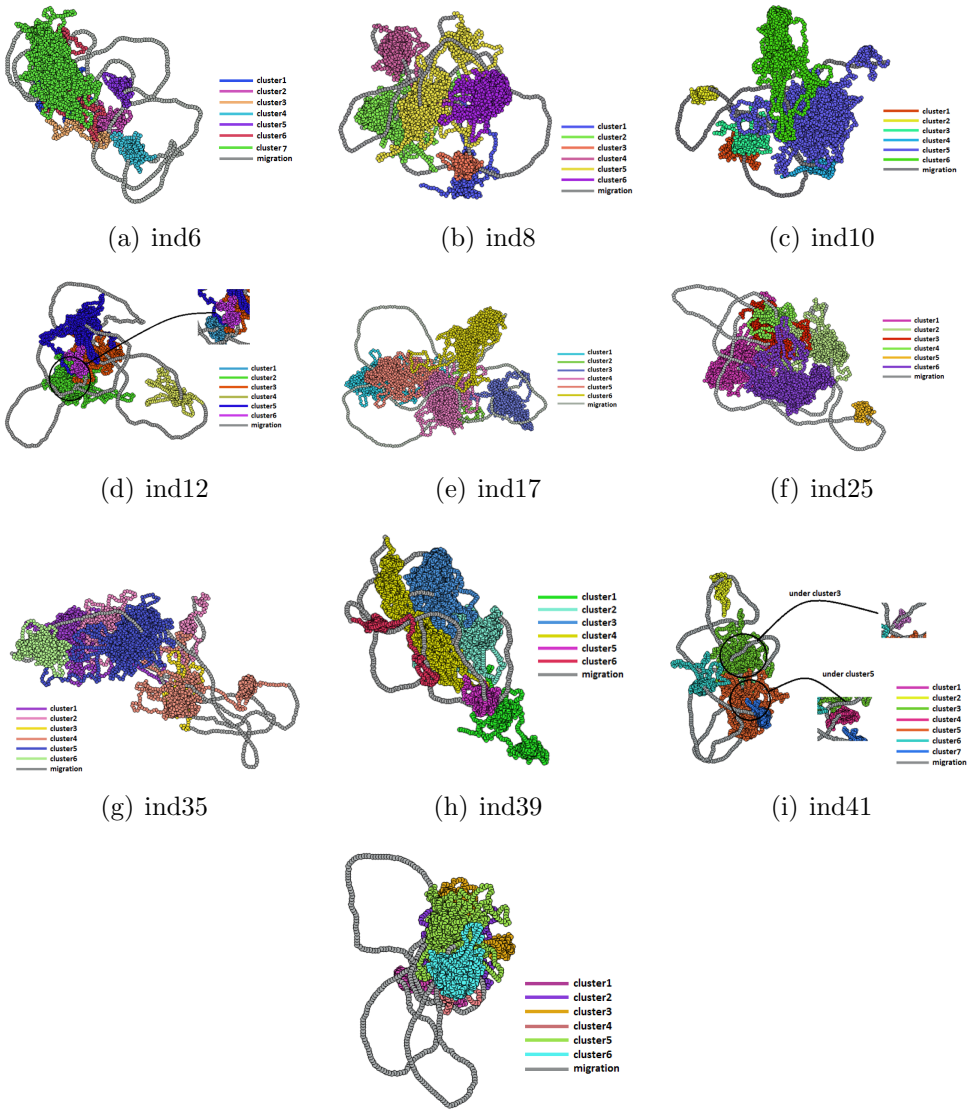
Table 4.1: Point classification in the ground truth

Traj-Id	**number of clusters**	**% clustered points**	**% local noise**	**% transition points**
Ind6	**7**	87.06	10.49	2.45
Ind41	**7**	91.09	6.50	2.41
Ind1	**7**	92.06	4.80	3.14
Ind35	**6**	85.27	12.44	2.28
Ind10	**6**	85.85	12.14	2.00
Ind39	**6**	86.45	11.57	1.97
Ind14	**6**	86.57	11.54	1.88
Ind25	**6**	86.23	11.36	2.40
Ind17	**6**	86.61	11.26	2.13
Ind12	**6**	87.41	10.12	2.47
Ind8	**6**	88.17	9.70	2.12
Ind49	**6**	88.58	8.88	2.54

Evaluation metrics

There are numerous quality metrics available to compare clustering w.r.t. ground truth. Among these, we choose, for deliberate redundancy, two metrics from different families, set-matching and counting pairs, respectively [9].

Further, we consider a third metric counting the different number of clusters as simple measure of structural similarity of segmentations. Let $R = \{r_i\}_{i\in[1,n]}$ be the set of n "true" clusters, $S = \{s_i\}_{i\in[1,m]}$ the set of m stay regions detected by SeqScan, N_s the total number of clustered points in the SeqScan output, and N_r the total number of clustered points in the ground truth. The metrics are



(a) ind6 (b) ind8 (c) ind10

(d) ind12 (e) ind17 (f) ind25

(g) ind35 (h) ind39 (i) ind41

(j) ind49

Figure 4.7: Ground truth: planar representation of 10 trajectories.

Table 4.2: Evaluation metrics

Metric	Defs
H-Purity (R, S)	Purity(R,S)= $\frac{1}{N_s}\sum_k \max_j \|s_k \bigcap r_j\|$ InvPurity(R,S)= $\frac{1}{N_r}\sum_k \max_j \|s_j \bigcap r_k\|$ H-Purity (R,S)= $\frac{2\times Purity(R,S)\times InvPurity(R,S)}{Purity(R,S)+InvPurity(R,S)}$
Pairwise F-measure (R,S)	TP=#pairs assigned to the same cluster is R and S FP=#pairs assigned to different clusters in R but to the same cluster in S FN=#pairs assigned to different clusters in S but to the same cluster in R Precision=$\frac{TP}{TP+FP}$ Recall=$\frac{TP}{TP+FN}$ F-measure (R,S)=$\frac{2\times Precision(R,S)\times Recall(R,S)}{Precision(R,S)+Recall(R,S)}$
Diff (R,S)	$\|card(S) - card(R)\|$

described in the sequel while their definition is reported in Table 4.2:

- **Harmonic mean of Purity and Inverse Purity: H-Purity**
  - **Purity [81]**: The purity is expressed by the following formula:

$$Purity(R,S) = \frac{1}{N_s}\sum_k max_j|s_k \bigcap r_j|$$

    In general, the Purity metric penalizes clusters containing items from different categories, while it does not reward the grouping of items from the same category.
  - **Inverse purity:[3]** The Inverse purity can be expressed by the following formula:

$$InvPurity(R,S) = \frac{1}{N_r}\sum_k max_j|r_k \bigcap s_j|$$

    Inverse Purity rewards grouping items together, but it does not penalize mixing items from different categories.
  - **Harmonic mean: [52]** The Harmonic mean of purity, hereinafter denoted H-Purity, mediates between Purity and Inverse Purity. It is defined by the formula:

$$H - purity = \frac{2 \times Purity \times InvPurity}{Purity + InvPurity}$$

- **Pairwise F-measure [3]:** Pairwise F-measure is defined as the harmonic mean of pairwise precision and recall. Knowing that:
  - TP= True positive. Number of pairs assigned to same cluster is R and in S.
  - FP= False positive. Number of pairs assigned to different clusters in R but to the same cluster in S.
  - TN= True negative. Number of pairs assigned to different clusters in R and in S.
  - FN= False negative. Number of pairs assigned to the same cluster in R but to different clusters in S.

  The precision and recall definitions are defined as follows:

$$Precision = \frac{TP}{TP + FP} Recall = \frac{TP}{TP + FN} \tag{4.1}$$

  The precision indicates the percentage of pairs of points detected correctly as clustered from all the pairs that were detected as clustered by the algorithm. While the recall is the portion of pairs detected correctly as clustered among all the pairs that are clustered in the truth. F-measure is the harmonic mean:

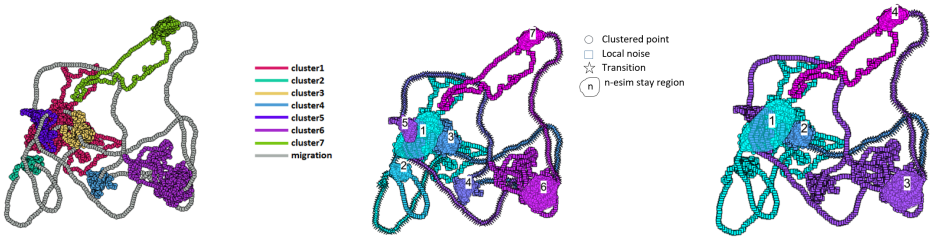
$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

- **Diff:** in brief, it computes the difference between the number of stay regions and the number of clusters

4.3.2 Experiments

Experiment 1: structural comparison and impact of the δ parameter. The goal is threefold: to compare structurally the two segmentations by confronting the number of stay regions detected by SeqScan with the actual number of clusters in the ground truth; to determine a suitable set of input parameters for the algorithm; and to assess the impact of δ over segmentation. For every trajectory of the dataset, SeqScan is run with parameters that only differ for the value of the presence δ, set to 20 days and 100 days respectively. In both cases the density parameters are: $\epsilon = 200, K = 50$. Such parameters are chosen through an iterative process. The resulting number of clusters constrasted with the true number is reported in Table 4.3. With $\delta = 20$ days the number of stay regions in the trajectory segmentation is substantially close to the number of clusters in the groud truth while with $\delta = 100$ days the number of stay regions is substantially

different (the statistical significance is evaluated through the Kruskal-Wallis test [42]). Visual analysis can provide further information.



(a) Traj IND1: the ground truth in Figure 4.5

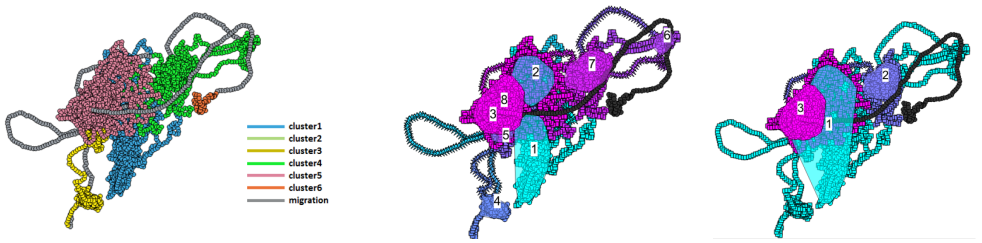
(b) Segmentation with $\delta = 20$

(c) Segmentation with $\delta = 100$

Figure 4.8: Segmentation of trajectory IND1 with varying values of δ

As an example, Figure 4.8 and Figure 4.9 illustrate the segmentation of the trajectories IND1 and IND14, respectively for the two values $\delta = 20$ days and $\delta = 100$ days. The clustered, transition and noise points are displayed using different shapes. The color gradient indicates the temporal order of stay regions. The clustered points are enclosed in a progressively number polygon obtained as convex hull of the set of points.

The segmentation of the trajectory IND1 in Figure 4.8.(b) for $\delta = 20$ days, contains the same number of clusters of the ground truth in Figure 4.8.(a). Moreover there is visual evidence of good matching of the segmentation with the ground truth. By contrast, the segmentation in Figure 4.8.(c) only contains four large clusters. In this sense, the parameter δ allows for the tuning of the temporal granularity of clusters. In the second trajectory, IND14, the number of clusters detected by SeqScan with $\delta = 20$ days is 8 against the true 6 clusters (4.9.(b)). It can be noticed that SeqScan recognizes as distinct two clusters (cluster 1 and 2) that in reality are part of a unique cluster in the ground truth. With $\delta = 100$ days, the number of clusters decreases to 3 (Figure 4.9.(c)).



(a) Traj IND14 (b) Segmentation with $\delta = 20$ days (c) Segmentation with $\delta = 100$ days

Figure 4.9: Segmentation of trajectory IND14 with varying values of δ. In this case, spatial overlapping clusters are not fully visible

Table 4.3: Experiment 1: Number of clusters with $\delta = 20$ days and SeqScan with $\delta = 100$ days

Trajectory-Id	**Number of clusters**		
	Ground truth	**SeqScan: δ=20days**	**SeqScan: δ=100days**
IND1	7	7	4
IND6	7	5	3
IND8	6	7	5
IND10	6	6	2
IND12	6	4	3
IND14	6	8	3
IND17	6	7	3
IND25	6	6	4
IND35	6	8	5
IND39	6	8	3
IND41	7	6	4
IND49	6	7	1

Experiment 2: analysis of the quality indexes

The visual comparison performed at the previous step, though useful, does not provide any quantitative measure. To that end, we run SeqScan with the "good" parameters determined at the previous step: $\epsilon = 200, K = 50, \delta = 20$. Next we compute, for every trajectory, the quality indexes resulting from the comparison of the SeqScan outcome with the ground truth. Table 4.4 reports the indexes

H-Purity and Pairwise F-Measure for every trajectory. It can be seen that the two indexes are substantially aligned and that overall the quality of the clustering is high. It can be noticed however that the H-purity value of the trajectory IND14 seen earlier - the one reporting the highest structural difference - is among the lowest in the table. This means that clusters may lack either compactness or homogeneity, in line with the visual analysis. By contrast the quality of the segmentation of IND1 is quite high, again in line with the structural comparison. Overall, the indices show a good matching with the ground truth both at the level of structure and of single clusters.

Table 4.4: Experiment 2: H-Purity and Pairwise F-measure

Traj-Id	**H-purity**	**Pairwise F-measure**
IND1	0.98	0.95
IND6	0.89	0.73
IND8	0.90	0.84
IND10	0.93	0.86
IND12	0.93	0.83
IND14	0.87	0.84
IND17	0.95	0.89
IND25	0.9	0.81
IND35	0.91	0.83
IND39	0.91	0.85
IND41	0.97	0.93
IND49	0.96	0.92

Experiment 3: noise analysis

In this experiment we analyze the contribution of the local noise to the quality of clustering. We recall that the unclustered points can represent either transitions or local noise. For this experiment, we contrast the quality of clustering in presence of local noise with the quality of the clustering in absence of local noise (i.e. the local noise points are seen as elements of the clusters). For the evaluation, we use the Pairwise F-measure because seemingly less favorable than H-Purity. Table 4.5 reports the value of the index in the two cases. The Kruskal-Wallis test confirms the significance of the discrepancy that can be seen in the table, or, put in other terms, that the local noise has an impact on the quality of clustering.

Table 4.5: Experiment 3. Pairwise F-measures in the two cases: clustering without local noise and clustering with local noise, respectively. The green color highlights the greater value in each row

Traj-Id	**Pairwise F-measure**	
	no local noise	**with local noise**
IND1	0.99	0.95
IND6	0.81	0.73
IND8	0.98	0.84
IND10	0.97	0.86
IND12	0.92	0.83
IND14	0.93	0.84
IND17	0.98	0.89
IND25	0.90	0.81
IND35	0.96	0.83
IND39	0.96	0.85
IND41	0.99	0.93
IND49	0.97	0.92

4.4 Sensitivity analysis

We take advantage of the animal dataset, used in the previous section, to test two additional properties related to the sensitivity of the algorithm to internal and external parameters. In particular, the first experiment is to perform the analysis of the input δ parameter through the computation of the function f_T (see Section 3.4.3). This experiment provides insights into the temporal granularities of the clusters contained in the trajectories of the dataset.

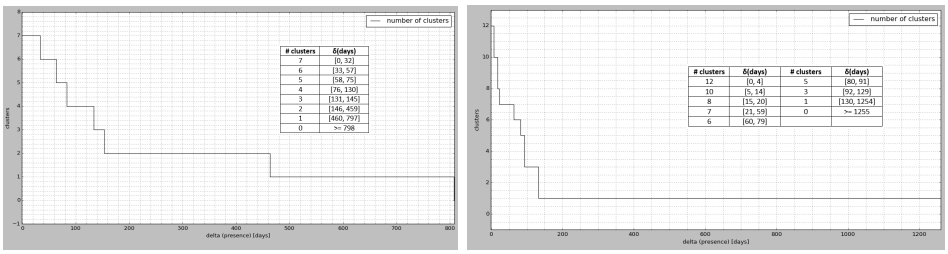
The second experiment is to analyze the sensitivity of SeqScan to the sampling rate. We recall that the temporal granularity of clusters depends on *presence*, and that the presence measurement is heavily affected by the size of the temporal interval between points. The purpose of the experiment is to evaluate the impact on clustering quality of the sampling rates, in other terms the robustness of the clustering framework against incomplete location information. The experiments are detailed next. For the sake of space, we limit ourselves to analyze the two trajectories IND1 and IND14 in Figure 4.5.

Experiment 4: analysis of the δ parameter.

We analyze the behavior of the function $f_T()$ describing the relationship between the presence parameter δ and the number of stay regions detected by SeqScan. We recall that the function $f_T()$ is computed by the Algorithm 2 that repeatedly runs SeqScan with different values of δ. This function is implemented as part of the MigrO environment. In this experiment, the function is built by running SeqScan with the usual density parameters $\epsilon = 200$ and $K = 50$ and with δ varying between 0 and the duration of the trajectory. The number of iterations is limited to 160, i.e. SeqScan is invoked 160 times. Figure 4.10 displays the plots of $f_T()$ for the two trajectories IND1 and IND14. For clarity, the function is also reported in tabular form. It can be seen that, in both cases, the number of clusters decreases for increasing values of δ. Thus the maximum number of clusters is obtained with $\delta = 0$. In particular, the segmentation of IND1 (Figure 4.10.(a)) consists of 7 clusters for δ ranging between 0 and 32 days. This is in line with the result in Table 4.3 reporting the number of clusters for $\delta = 20$ days. The same holds for IND14 (Figure 4.10.(b)). An additional consideration is that the two trajectories exhibit a different sensitivity to the parameter. In particular, for IND14 a relatively small variation of δ, for low values of the parameter, heavily affects the segmentation.

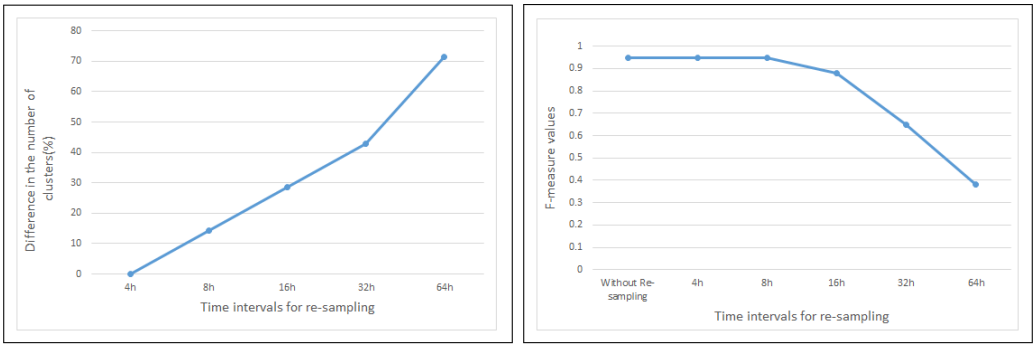
Experiment 5: sensitivity to the sampling rate

For this experiment, the trajectories are re-sampled considering time intervals of 4, 8, 16, 32, 64 hours (we recall that in the animal dataset the time interval has a width of 2 hours). Next the SeqScan is run over the re-sampled trajectories and the result contrasted with the ground truth, using two of the quality indexes discussed earlier, the difference in the number of clusters with respect to the regular trajectory (normalized) and the Pairwise F-measure. SeqScan is run with clustering parameters ϵ=200, δ=20 and K=50 points. The corresponding graphs are reported in Figure 4.11 and Figure 4.12. It can be seen that for lower sampling rates, the normalized difference in the number of clusters increases, while the quality of the cluster (i.e. F-measure) decreases. While this is the expected behavior, more interesting is the fact that the quality of clustering is not dramatically compromised if the sampling rate is reduced by 2 and even 4 times (i.e. time interval of 4 and 8 hours).



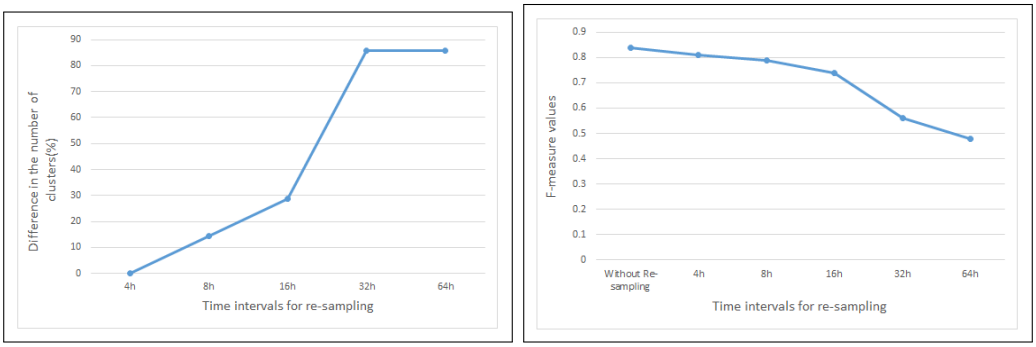
(a) Traj IND1 (b) Traj IND14

Figure 4.10: Experiment 4. The function f_T for the two trajectories IND1 and IND14. The function is reported in both graphic and tabular form.



(a) IND1: difference in #clusters (b) IND1: Pairwise F-measure

Figure 4.11: Experiment 5. Re-sampling of the trajectory IND1. (a) Normalized difference in the number of clusters (with respect to the regular trajectory); (b) Pairwise F-Measure computed w.r.t. ground truth



(a) IND14: difference in #clusters (b) IND14: Pairwise F-measure

Figure 4.12: Experiment 5. Re-sampling of the trajectory IND14.

4.5 Evaluation of the SeqScan-based technique for the discovery of periodic locations

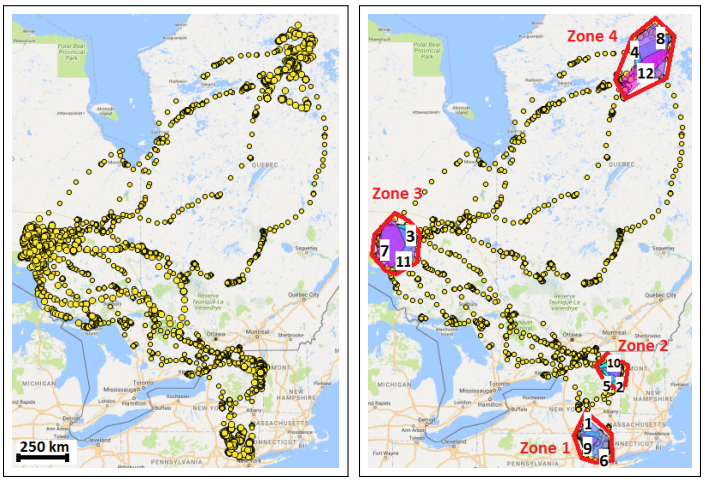
We turn to use SeqScan with real data and confront the solution proposed for the discovery of periodic locations with the MoveMine approach [74]. We use a dataset containing the GPS trajectory of one eagle observed for nearly three years while flying between US and Canada. The sample points have been collected between mid January 2006 and end December 2008 at a sampling rate that is highly irregular. The data can be downloaded from the Movebank database [3]. Notably, this dataset is the same used in the MoveMine project. Some cleaning operations are preliminarily performed over data. As a result, we obtain a trajectory of 14,442 points, extending over 1080 days, with an average step length of 3210 meters. The trajectory and its spatio-temporal representation is reported in Figure 4.13.(a) and 4.13.(c), respectively. In the following, we analyze: the zones and the periodicity of zones.

4.5.1 Zones discovery.

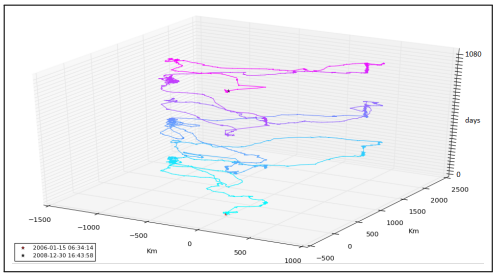
The analysis is performed in three main steps:
Step 1. Compute the sequence of stay regions. We run SeqScan with parameters: $\epsilon = 60km, N = 100points, \delta = 20days$. We obtain 12 stay regions (numbered from 1 to 12). The stationarity index [29] is high in all of the cases, meaning that the local noise in the region is limited and thus the staying is 'temporally dense'.

We recall that the transitions and the local noise are not relevant for this kind of analysis.
Step 2. Compute the similarity classes. We set the parameter $\psi = 0$ and obtain four classes, each containing three stay regions: $C_1 = \{1, 9, 6\}, C_2 = \{2, 5, 10\}, C_3 = \{3, 7, 10\}, C_4 = \{4, 8, 12\}$. The similarity table is reported in Figure 4.13.(d).
Step 3. Compute the zones and the number of visits. For each class we compute the corresponding zone as union set of the stay regions. The zones are denoted: $\hat{1}, \hat{2}, \hat{3}, \hat{4}$. The sequence of 12 stay regions can be rewritten in terms of zones: $\hat{1}, \hat{2}, \hat{3}, \hat{4}, \hat{2}, \hat{1}, \hat{3}, \hat{4}, \hat{1}, \hat{2}, \hat{3}, \hat{4}$. The stay regions and the grouping in zones are reported in Figure 4.13.(b). Every zone is visited three times. It is evident that the sub-sequence $\hat{1}, \hat{2}, \hat{3}, \hat{4}$ repeats itself, although with some irregularity. The following periodicity analysis provides further information.

[3] `\http:www.movebank.org`. Study: Raptor Tracking: NYSDEC



(a) The trajectory (b) Stay regions 1-12 and zones



(c) Spatio-temporal representation

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}
S_1	-	0	0	0	0	1	0	0	1	0	0	0
S_2	0	-	0	0	1	0	0	0	0	1	0	0
S_3	0	0	-	0	0	0	1	0	0	0	1	0
S_4	0	0	0	-	0	0	0	1	0	0	0	0.99
S_5	0	1	0	0	-	0	0	0	0	1	0	0
S_6	1	0	0	0	0	-	0	0	1	0	0	0
S_7	0	0	1	0	0	0	-	0	0	0	1	0
S_8	0	0	0	1	0	0	0	-	0	0	0	0.91
S_9	1	0	0	0	0	1	0	0	-	0	0	0
S_{10}	0	1	0	0	1	0	0	0	0	-	0	0
S_{11}	0	0	1	0	0	0	1	0	0	0	-	0
S_{12}	0	0	0	0.99	0	0	0	0.91	0	0	0	-

(d) Spatial similarity table

Figure 4.13: Analyzing the real trajectory of an eagle: stay regions and zones discovery.

4.5.2 Periodicity analysis

We analyze first the periodicity of single zones and then of the entire sequence of zones. The temporal resolution of time series is set to 1 week. For each zone, we create a time series as follows. We consider the trajectory in the period between the beginning of the first visit and the end of the last visit. We split the temporal extent of the trajectory in weeks. Hence for every week, if the object is inside the zone at any instant, we create the symbol '1', '0' otherwise. We recall that the local noise is overlooked at this stage, thus the object is assumed be continuously present inside a stay region. We run the WARP algorithm over the time series and we select the smallest period with the highest confidence value. As a result, we obtain: two zones $(\hat{3}, \hat{4})$ have periods 53 and 51 weeks, respectively, with maximum confidence; the other two zones $\hat{1}, \hat{2}$ have periods 48 and 55 weeks, respectively. The confidence of the period of zone $\hat{2}$, is the lowest (0.9) among the zones.

To make the analysis of the behavior periodicity comparable with MoveMine, we consider the zones $\hat{1}, \hat{2}$ as a unique zone. From the analysis of the time series created out of the sequence of 3 zones, we find that the period of the behavior is 52 weeks with maximum confidence.

Comparison. The results are substantially aligned with those obtained by MoveMine (MoveMine detects a periodic behavior of period 363 days). In both cases the eagle flies from the New York area towards the Great Lakes area, to finally reach the Quebec area, before returning back to the New York area. There is, however, a substantial difference between the two methods. MoveMine detects zones (called reference spots) as spatial-only clusters, and exploits signal processing techniques to extract from a noisy signal the sequence of temporally separated regions. Our technique does the opposite: it starts from the extraction of temporally separated regions, through the use of SeqScan, and finds the zones. Consequently, the noise can be easily separated from the clusters at early stage, and that simplifies the analysis. It is interesting to observe a discrepancy in the results obtained by the two methods. In particular, MoveMine detects one intermediate spot during the flight back from Quebec to New York area, that our technique seems not to recognize. In reality, such a stop has a duration of only few days, therefore SeqScan does not recognize it as a stay region. Such an observation has been made possible by the segmentation mechanism, which discriminates between clusters and transitions, allowing for a detailed inspection of the behavior.

4.6 SeqScan performance evaluation

In this section we report a few experiments analyzing the performance of SeqScan. Preliminarily we detail some implementation aspects that are relevant for the discussion. We emphasize that the implementation of SeqScan is not part of the Thesis (further details can be found in [38])

4.6.1 Insights into the implementation of SeqScan

Figure4.14 illustrates the UML diagram for the main classes used in the implementation of SeqScan. In more detail:

- The class *Point* represents each point in a trajectory.
- The class *Region* represents a group of points forming a DBSCAN cluster that possibly is or will become a stay region

- *Time descriptor* class represents the time segment for each Region, specifying the presence within and the absence from it. It consists of a set of time intervals.

The Figure4.14 shows the data structure of these objects and the relations between them.

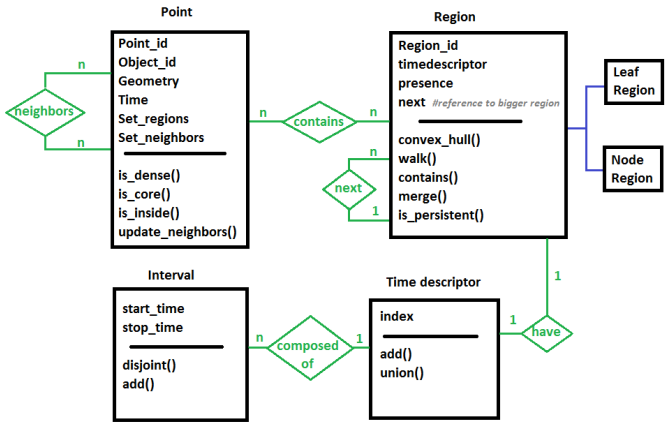


Figure 4.14: UML diagram for the classes used in the implementation of SeqScan

The program implementing SeqScan takes as input the list of points representing the trajectory of an object, temporally ordered. The output is:

[1] A set of clusters, each represented by a set of points

[2] The local noise

[3] The transitions

This program iterates over the input points, and for a point p with a timestamp t_p, the situation is as follow: The points processed before p and already assigned to a closed clusteror detected as noise, are not going to be included into any further processing. In other terms, supposing that the last found cluster is closed at time t_{last_C}, all the points with $t<t_{last_C}$ will not be taken into further consideration. The points with timestamps $\in]t_{last_C}, t_p[$, will be referred to as "the *pool*". These points can:

- belong to the active cluster
- belong to a dense region
- not belong to any region yet

The first step is to compute the distance between p and every point in the pool. If this distance is less than the threshold ϵ, the point is added to the neighborhood N_p of p. Then the neighborhood of each point $q \in N_p$ is updated. The update process is as follow: p is added to the neighborhood N_q of q. If q was a core point, p is added to q's regions. In the case where q becomes a core after the addition of p, two situations need to be considered:

- q was a border point, then all its regions are merged into a new one and all its neighbors are added to this new region.
- A new region is created with q as core and all of its neighbors as border.

At this stage, p is added to all the regions where valid. So if p is found to be inside the active cluster, we say that the latter is expanded. Otherwise, the possibility of creating a new cluster with p and its neighbors is checked; if so, the current active cluster is closed, the new cluster becomes the active one, the pool is updated accordingly.

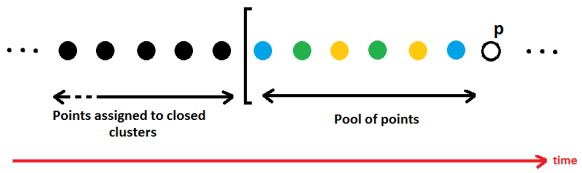


Figure 4.15: Points processing: At every instant a point in the pool may belong to the active stay region, to a cluster or to none.

4.6.2 Efficiency evaluation and experiments

The most expensive operation is to determine the $\epsilon - neighborhood$ of every point. In the worst case the pool consists of all the points of the trajectory, thus we have to compute $\sum_{i=1}^{n} i = \frac{n \cdot (n-1)}{2}$ distances. The complexity of the algorithm is $O(n^2)$.

In the following we describe two experiments conducted in order to study the effect of the increased number of input points on the run time of the algorithm.

Experiment 1: Increasing the density

Consider a trajectory of length 19500 points, we duplicate the points many times (39000, 78000, 156000...) and we run SeqScan every time in order to get the run time value (average value of repeated tests). The duplication is made in such a way that each point has a copy of itself with same spatial location but with timestamp increased by half the time interval between two consecutive points in the original

trajectory. So the duplicated trajectory will have the same duration as the original one, but with doubled clusters density. The results are shown in Figure4.16.

Experiment 2: Increasing the trajectory length

We repeat the same procedure as the previous experiment using the same trajectory, but this time the duplication is made in such a way that the original trajectory is repeated, so its duration is doubled, the number of resulting clusters is doubled too but their densities remain the same. The results are shown (in blue) in Figure4.16.

Discussion of the results

The results show that when we duplicate the number of input points the run time increases rapidly in experiment 1, while its increase is stable and almost linear in experiment 2. These results are interpreted by the fact that the most expensive phase of the algorithm is to compute the distance between a point and the corresponding pool points, and by doubling the points that are supposed to form a cluster, we are duplicating the pool points for every iteration and thus increasing this computation process and the memory consumption which will require more time for running.

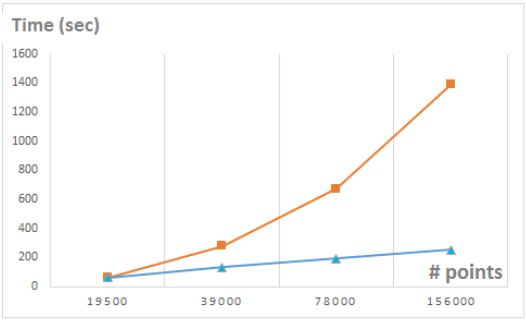


Figure 4.16: Experiment 1: run time of SeqScan for increasing clusters density (orange). Experiment 2: run time for increasing trajectory duration (blue).

4.7 Discussion and conclusion

In this chapter, we have experienced a research methodology that combines two activities generally kept distinct, namely the investigation of a novel theoretical framework and the application of such a framework in the context of a real domain. Actually, we have chosen to combine the two streams to ensure a more robust

evaluation of the effectiveness of the analytical framework, also in view of a possible deployment. Additional considerations:

- Validation strategies. Probably the most challenging question, with respect to validation, is whether the proposed solution can be effective in real applications and that motivates the concern for external validation practices [36]. To that end, in [29] we have experienced a first approach where we evaluate SeqScan using real animal trajectory data. The problem with real data is that if the behavior is inherently complex and only known at macroscopic level (e.g. migratory behavior), domain experts may be not in the condition of classifying every point with sufficient confidence and thus the evaluation can be only conducted at a coarse level. In this sense, the use of a synthetic dataset built on an independent movement model conceptually encompassing the pattern of concern has dramatically improved the accuracy of the evaluation.

- Evaluation metrics. We have used Purity and Pairwise F-measure. Yet, these indexes are specific for the evaluation of traditional clustering while the segmentation problem, we are dealing with, is somehow different. Indeed, defining appropriate internal and external evaluation metrics for cluster-based segmentation is an open issue. A first proposal of internal indicator, called *stationarity index* is presented in [29]. Applied to single clusters, the stationarity index is an estimate of the "temporal density" in the cluster. This topic could be investigated as part of future work.

- Generality of the proposed framework. As the external evaluation has been conducted on animal trajectories, one could raise the question on whether the scope of the solution is confined to the ecological domain. In reality, the model has been defined in a rigorous and general way, thus is prone to be applied in a variety of domains, such as human mobility analysis.

The results of the external evaluation process can be finally summarized as follows:

- The experiments conducted on the animal dataset used as ground truth show that overall the degree of matching of the SeqScan segmentation with the ground truth is high (Tables 4-6). We recall that we have used the same set of parameters for all of the trajectories ($K = 50, \epsilon = 200, \delta = 20$). Therefore, it is likely that with a finer-grained tuning of the parameters, the quality improves further. Importantly the ground truth is generated independently from the clustering while the evaluation has been conducted in a blind manner ignoring the simulation parameters. This is important for two reasons: it definitely supports the thesis that SeqScan can detect this class of patterns; and that the evaluation is fair.

- For the practical application of SeqScan, the generation of the function f_T, exemplified in Figure 4.10, can be extremely useful to determine a suitable set of values for δ, in the same spirit of Optics [7].
- As exemplified in Figure 4.11 and Figure 4.12, the experiments show that SeqScan is resilient to relatively low sampling rates.
- Finally,the approach proposed in order to support the discovery of periodic locations and behaviors can compete with state-of-the-art techniques in detecting periodical behaviors with noise, while offering a flexible and principled solution.

Chapter 5

From individual to collective behavior analysis

5.1 Overview

Mobility patterns are commonly classified in two broad categories, individual and collective patterns, depending on whether the moving entities of concern are single objects, or groups of objects[30, 6]. In this Chapter we investigate the use of SeqScan for the discovery of collective patterns. Collective patterns include T-pattern [40], flock[10, 88], leader [4, 5], convoy [57, 58]. In particular, a collective pattern of major practical relevance is the *gathering* pattern. A gathering is a group of moving objects that stay together in a geographical area for some time. The study of the gathering pattern poses interesting issues regarding both the definition of a suitable model and the efficiency of the pattern discovery process [95]. This research focuses on the modeling aspects. The contribution of this Chapter can be summarized as follows:

- We introduce a possible classification of gathering patterns distinguishing in particular gatherings with persistent and non-persistent participation;
- We present a first approach to the extraction of gathering patterns with non-persistent participation grounded on SeqScan;
- We evaluate the technique on real data.

The rest of the paper is organized as follows: Section 5.2 presents a possible classification of gathering patterns and related work; Section 5.3 introduces the k-gathering pattern, while Section 5.4 details the pattern discovery algorithm; the experimental evaluation is reported in Section 5.5; conclusive remarks in Section 5.6. Most of the materials reported in this Chapter has been published in [50].

5.2 Understanding the gathering behavior

In broad terms, a gathering can be defined as an assembly of objects that share the same space for some time. A gathering may be the result of some collaborative activity performed at a certain location, e.g. a set of workers engaged in the construction of a building, or it may signify that some social event is taking place such as a concert or a demonstration. A gathering has a temporal extent, i.e. it is created and then dissolves after a period of time, as well as a spatial extent, namely occupies a region of space, though not necessarily the same for the whole period of the gathering. The characterizing feature of gathering is that the group of objects is highly dynamic, objects can join the group and leave it at any time or return back after a while. Gatherings have thus a mutable shape and composition. Despite these common characteristics, the notion of gathering presents different facets and interpretations. A possible classification is presented in the following.

5.2.1 A classification of the gathering behavior

Gathering patterns can be classified as *static* and *dynamic*, depending on whether the assembly takes place in a fixed location or rather in a location that changes slowly in time. Two application scenarios can better illustrate these two situations.

- *Scenario 1: Exhibition.* An exhibition exemplifies a static gathering. The visitors of the exhibition participate at the event, which is held in some place, for a limited, though significant, time. During the visit, individuals are close to other individuals. Moreover, typically visitors are not forced to follow a specific path inside the exhibition, thus can stop at any point for the desired time. As a result the object movement appears random.

- *Scenario 2: Protest march.* A demonstration is an example of dynamic gathering. Again, this is an assembly of objects that can join and leave at different times the event. However, unlike the previous example, the group moves as a whole. That is, although the movement of some participants may appear random, the group remains identifiable as a whole and moves following a direction.

To further characterize a gathering we consider two additional criteria, orthogonal to the distinction between static/dynamic gathering (see Figure5.1):

- *Persistent vs. non-persistent participation.* We say that the participation in the gathering is persistent if there is a set of objects that remain in the group for most of the time. In contrast, the participation is non-persistent if the members can leave at any time and be replaced by other objects.

- *Participation with absence vs. no absence.* Whenever the participation is persistent, a gathering can include members leaving the assembly temporarily before joining the group again. We qualify this form of participation as "with absence".

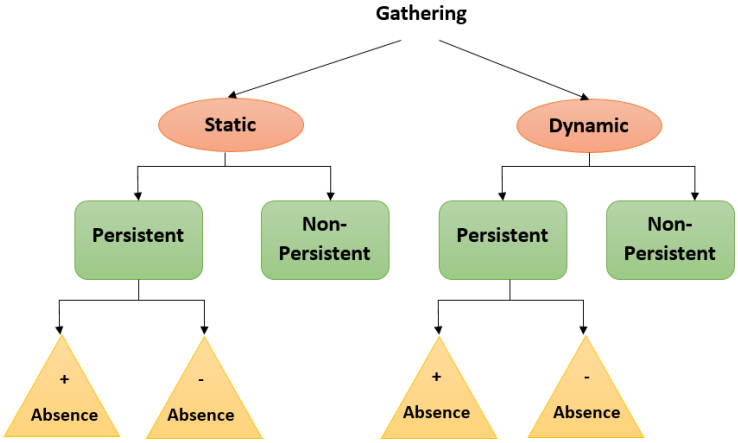


Figure 5.1: Classification of gatherings

5.3 Related work

Preliminarily, it is worth emphasizing that a gathering is not simply a cluster of points close in space and time. In that case, the cluster could contain points belonging to only one or few trajectories and thus the number of objects would be too small for the assembly to be significant. In addition, even though the cluster would contain points from a large number of trajectories, the participation of objects to the assembly could be merely occasional. Therefore for a gathering to take place a sufficient number of objects are to be located in the same region for a sufficient time. In the following we study two major streams of related work the first is concerning the discovery of collective patterns and the second is about the discovery of communities.

5.3.1 Collective pattern discovery

Flock and convoys. Following the definition by Benkert et al. [10], a flock is a set of objects that move together along a path for some amount of time. Therefore, given a set of objects, a group of at least m objects located within a disc of radius r for Δ time can be considered as a "flock". Other works try to optimize the solution of extracting flocks, like In [88] where they suggest the usage of a predefined time

duration and they propose algorithms based on heuristics, to reduce the total number of candidate disks to be combined. Also, in [89], they added a heuristic filter to the flock detection process in order to tolerate the missing or noisy points in the trajectories. Furthermore, the concept of *Herd* was proposed in [53] where they suggest a way to discover how flocks interact with each other over time, and they define four types of evolvements: join/ leave, when a flock becomes merged with/ separated from another one, and expand/ shrink when the number of objects in the flock increases/decreases noticeably.

But, one of the limitation of the flock is the restriction to the disc shape and size while clustering the objects. Hence, another concept is introduced: *convoy*. As the flock, the convoy is defined as a group of objects traveling together for a certain time, but it avoids the restriction on the shapes and size of the pattern, that's why in [58] they introduced the density-connection between objects. The work has been extended in [57] by developing more advanced algorithms to increase the efficiency of extracting the convoys from set of objects trajectories.

However, flocks and convoys they both share a limitation: they require the objects to stay together for consecutive timestamps, an object cannot leave the group for a short time and come back later. This issue was addressed by [73], where they proposed the *swarm* concept. This concept is based on the fact that some objects may diverge temporally from the group to re-join later. They define the swarm as a group of moving objects that travel together in arbitrary shapes, for certain timestamps that may not be consecutive. The swarm concept might seem similar to the concept of *moving cluster* presented by kalnis et al[59]. The latter is based on the idea that a cluster retains its density during the whole required lifetime. But its drawback is that similarity is considered between each two temporally consecutive clusters only, which might lead into a moving cluster that ends up with a totally different set of objects compared to its initial formation.

All the aforementioned concepts cannot be a solution for the problem we are trying to solve, because they find the groups of objects moving together while what we are looking for is to find the group of objects that gather while staying in some region.

In addition, the concept of gathering we are looking for is also different from the *co-location* pattern, i.e. frequent cliques of objects of given type, although both require a set of objects to be nearby [54], but the latter doesn't consider the time aspect. Also, the input in [54] is not a set of trajectories, but a collection of Boolean spatial features, which locate it far from the scope of our work and what we try to do. In [92] they study the problem of finding co-location patterns out of spatial data, but the input is a set of events and not trajectories too. In all of the works related to co-location, only the spatial aspect of objects is taken into consideration, without the temporal one, and thus it cannot present a solution of

the problem we are trying to solve.

Two more concepts, they could be the closest ones to what we are looking for: The *meeting* and the *gathering* patterns. That's why we are going to explain them in more details in what follows.

Meeting: This pattern is presented in [45], it satisfies the essential characteristics of a gathering because it detects behaviors where a sufficient number of objects are located in the same place for a sufficient time. Consider a set D of trajectories. Every trajectory is associated with an object. A meeting consists of $N \leq |D|$ objects that stay within a stationary disk of radius r for a minimum time τ. The meeting has two variants, depending on whether the objects are the same for the whole period (*fixed meeting*) or leave the meeting and are then replaced (*varying meeting*). Figure5.2 and Figure5.3 illustrate the two types of meeting.

The meeting pattern is an example of static gathering pattern discussed in the previous section. If the meeting pattern is qualified as fixed, it means that the participation is persistent for the whole members that, in addition, cannot experience periods of absence. Conversely if the meeting is varying, the participation is non-persistent. In spite of its simplicity, this model relies on a few assumptions that are difficult to meet in practice. In particular the extent of the meeting region, i.e. the radius r, is hard to estimate because it does not have a clear application meaning. Moreover, if no assumption is made on the meeting points, the participants can be distributed arbitrarily in space thus not necessarily inside a circular region.

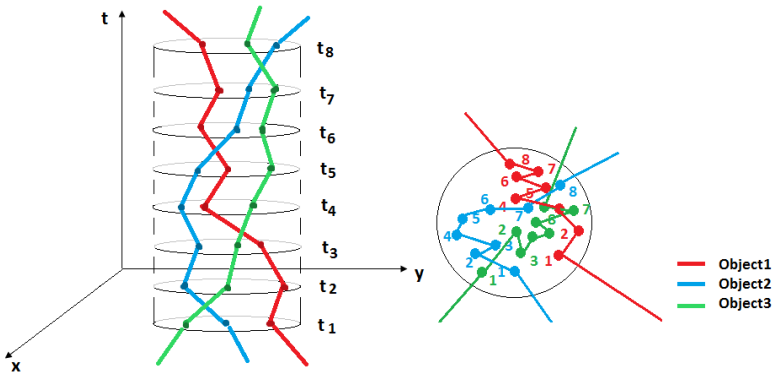


Figure 5.2: Fixed meeting of 3 objects. The three objects are located inside a stationary disk at every instant of the time interval $[t_1, t_8]$

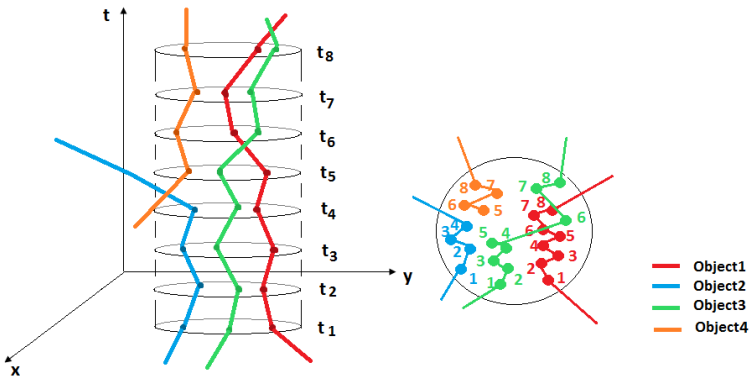


Figure 5.3: Varying meeting. While object1 and object3 remain within the disk for all the meeting duration, object2 leaves the disk definitely at time t_4, and object4 joins it at t_5. At each instant of the time interval at least 3 objects are inside the disk.

Crowd and gathering: A more flexible model - the one that explicitly introduces the term *gathering* - is defined in [95]. In this model, a gathering is basically defined as a set of *participants* in a *crowd*. The crowd is a set of objects that at every instant of an interval of minimum width, forms a cluster (e.g. density-based cluster). These clusters are called *snapshot* clusters. The *participants* are objects that are members of the crowd for significant time. A gathering is a set consisting of a significant number of participants. The pattern is exemplified in Figure 5.4.

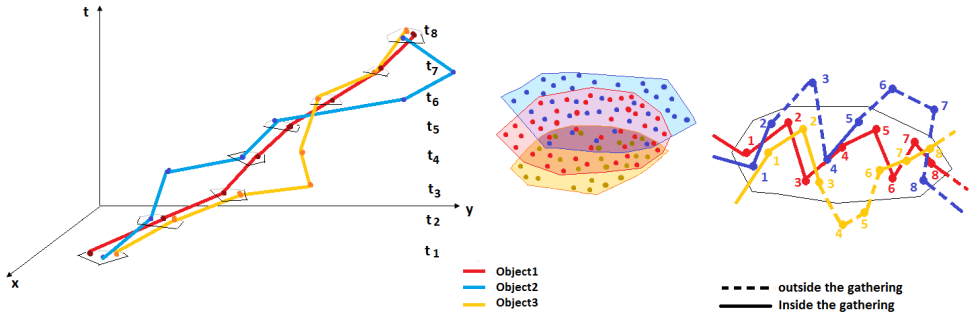


Figure 5.4: An example of gathering based on the model in [95]. The gathering is defined over the interval $[t_1, t_8]$ and contains three objects. The snapshot clusters (with cardinality at least 2) are shown at each instant.

It can be seen that the notion of crowd is defined by a constraint on the density of objects over a time interval. The shape of the dense assembly can thus be arbitrary and change in time. This also implies that the group can move

slowly in time. This model can be classified as a dynamic gathering, moreover the participation is persistent and the participants can experience periods of absence.

A major problem of this model is that it requires in input a high number of parameters. In particular, besides the two density parameters requested by the density-based clustering technique (i.e. DBSCAN), 6 additional parameters are to be specified, in particular: m_c the minimum number of objects of the snapshot cluster; K_c minimum number of consecutive time instants in which the snapshot cluster contains at least m_c objects (duration of the crowd); $K_p \leq K_c$ the number of instants at which an object should appear in a snapshot to be a participant; $m_p \leq m_c$, the minimum number of participants in each snapshot cluster; δ, the maximum Hausdorff distance between consecutive snapshot clusters. Some of these parameters, in particular the Hausdorff distance between clusters, are hard to set. All that adversely affects the usability of the technique.

5.3.2 Community discovery perspective

In the literature, there is a concept of community which, in broad terms, means a set of entities that are closer to each other (in a sense depending on the application domain) than the entities outside the community. Thus, the entities of a community share some common properties. [23, 37]. By observing this definition, it can be said,by an intuitive reasoning, that it is possible for the problem of gathering discovery to be addressed from a community detection perspective.

That's why in this section we are going to show a brief overview on what already exist in the literature concerning this domain, while explaining the similarities and the differences with what we are looking for.

Community detection is in general applied on graphs [37]. A graph consists of nodes connected through edges. According to the application, a node can represent an entity (in case of mobility data, a node may be a point of a trajectory), and if an edge exists between them, it indicates that there is some similarity between these nodes (an edge can be weighted according to the similarity value). A node may have also attributes describing its characteristics.

Many techniques exist in order to discover communities based on the required definition of the latter, like grouping nodes having similar attributes in one community, or finding communities where the nodes inside are densely connected by edges [23]. Some of these techniques use clustering algorithm also to extract the communities, or they are based on the concept of grouping nodes together in order to optimize some quality indexes.A popular quantitative measure for the quality of a given community is the "modularity" defined in [76]. It is the fraction of edges inside the community minus the expected fraction of edges if the graph was random. Its value lies in the range $[\frac{1}{2}, 1[$. As the modularity has a higher value as the communities detected in the graphs are denser and therefore correctly detected.

Another techniques aim to extract communities in such a way that all the nodes of a community have edges following the same structural rule. The approaches adopted to solve such a challenge belong to the graph mining algorithms, like the K-clique[77] algorithm which defines a k-clique-community as the union of all k-cliques (fully connected subgraph of at least k nodes) that can be reached from each other by sharing at least $k-1 nodes$. Many other algorithms representing a relaxant concept of k-clique exist, like the s-plex[64] and the bi-clique[70]...

However, these concepts and techniques for detecting the communities, beside their complexity, do not take into consideration the peculiarities and characteristics of the data, especially for the spatial-temporal ones, which are critical to consider when working with mobility data, hence we cannot confirm that they are suitable for detecting collective behavior for moving objects.

Only some studies have introduced the spatial dimension in such domain, and made the problem more specifically a geosocial community detection, where the connection between 2 entities is characterized by both the social and spatial aspects. One of these techniques is described in [83], where Shakarian et al. consider the problem of finding geographically dispersed communities, and they propose an algorithm which is optimized considering geospatial information in addition to social network information. In another technique [87], they construct a graph from information about gang members, where the nodes are the individuals and the edges are the geosocial similarities between them. The purpose is to find the communities among the gang members. So even though in [83] and [87] they introduced the concept of spatial distance between nodes, but it always accompanies and depends on the social connectivity between individuals.

Furthermore in [15], although they target trajectories in their work, but they aim to find communities of places instead of the moving objects. They define the community as a group of places that are highly connected by people mobility.

Beside the spatial aspect, the temporal dimension is not considered too in such techniques, because studying the spatial-temporal proximity between points belonging to different time instants will be difficult, even with time evolving community detection techniques, because it increases the computation and confuses the analysis process.

In addition, These techniques try to capture the time evolutionary process of communities [21, 75, 85], and to describe how each community forms, evolves, and dissolves for any time period [63]. And furthermore in [84] they were able to detect if a community splits into many other communities or merges with other communities. Thus, these methods focus on how the community evolves with time, so the output depends on time in a sense that on each instant they obtain the corresponding status of the communities formation rather than obtaining a unique output showing all the possible communities that express the existence of

gatherings in some space and time extents.

Following to this reasoning, we can consider that dealing with the problem of gathering from a community discover perspective is far from our scope and hence we will locate our work in the mobility data mining domain.

5.4 Discovering gathering patterns

5.4.1 Overview of the gathering model

The patterns we have seen so far, especially the meeting and the gathering patterns discussed in the previous section, are defined in terms of geometric constraints. Given a dataset of trajectories, the goal is to find the sub-trajectories satisfying such constraints.We argue that those approaches are intrinsically limited because exclusively focused on the geometric dimension of the problem, while the mobility context, in which the gathering takes place, is typically overlooked. As a matter of fact, the models are hard to validate in real applications [69]. We thus propose a different viewpoint. Such a perspective builds on the consideration that a gathering in a certain location consists of objects that autonomously stop at that location for a significant time. That is, a gathering results from the sum of individual behaviors. Also the fact that an object can temporarily leave a stop and thus be absent for a while, can be seen as a characteristic of the individual behavior.

Let's go back to the example of the exhibition, each of its participants has the segment in its trajectory corresponding to his stay in the exhibition representing a stop. This participant reaches the exhibition while driving toward it, then leave it also by driving away toward his home or other places. So by analyzing the participant's trajectory, his staying in the exhibition is considered as a stop. Also, this stop should last at least few hours, as a typical exhibition visitor is expected to stay, or else he is not really considered visiting the exhibition for the purpose of attending it. The same for the case of animals creating a habitat, for each participating animal the sub-trajectory corresponding to his stay in the habitat is considered as a stop in his trajectory.

Therefore the idea is to extract from every trajectory the stop-and-move patterns and use such information to determine whether there exists a sufficient number of stops forming a gathering at a shared location. All that suggests a slightly different formulation of the gathering concept.

Definition 5.1 (Gathering)**.** A gathering is a set of objects that individually stop for significant time in some region of space in spite of periods of absence and that, while in such a region stay in proximity to other objects.

Accordingly, the discovery process can be split in two main steps: the first step

is to identify the stops from the set of individual trajectories the second step is to use such information to find whether the stops form a gathering.

Hence, we can say that the discovery of stops from individual trajectories is key for the discovery of gatherings. SeqScan described in the previous chapters is the best technique for such a mission, since the stay regions (stops) are extracted while handling explicitly the absence from them.

Armed with the notion of stay region, we now turn to detail the process for the discovery of gathering patterns in compliance with Definition5.1.

5.4.2 The discovery process

Consider a dataset $D = \{tr_1, ..tr_n\}$ of spatial trajectories, each one associated with a different object. Assume the trajectories are sampled at time $t_1, t_2, .., t_m$. Given a generic spatio-temporal point we call *point type*, the trajectory the point belongs to. With slight abuse in the notation, we say that D represents the set of point types. The discovery process can be split in two main steps:

(a) Stop extraction, namely to identify the stay regions from every single trajectory

(b) Gathering discovery, that is to find whether stay regions from different objects form a gathering.

Stop extraction

We run SeqScan to extract the sequences of stay regions. In the simplest case, we use the same set of parameters over the whole set of trajectories. An alternative approach is to tune the parameters based on an internal quality index, such as the Stationarity Index presented in [56]. As a result, we obtain the set of $n = |D|$ sequences: $\{\mathcal{S}^i\}_{i\in[1,n]}$ with $\mathcal{S}^i = [s_1, .., s_k]^i$ a series of stay regions. Such a set provides the input for the next step.

Gathering discovery

We search for and analyze conglomerates of stay regions. Stay regions from different trajectories can be seen as sets of points of given type. Therefore if a gathering takes place, it is plausible there exists a dense region of points of different types, close in space and time. This suggests the development of a technique that, in the same line of the spatio-temporal clustering technique *ST-DBSCAN* [14], agglomerates neighbor points based on spatial and temporal proximity criteria.

ST-DBSCAN, as we have seen in the second chapter, extends DBSCAN by taking into consideration the time proximity between points by first filtering the data and retaining only the temporal neighbors and their corresponding spatial values. Two points are temporal neighbors if they are observed in consecutive time units. The substantial difference with ST-DBSCAN is that the points are to be of different types. Note that two gatherings can even occur simultaneously but at different locations, thus, unlike SeqScan, there is no temporal ordering among gatherings. We refer to this model as density-based gathering model.

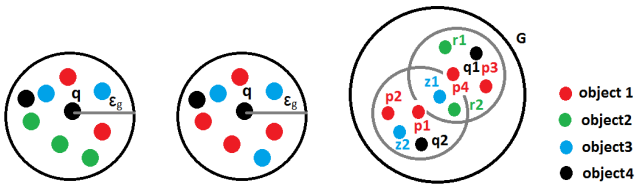
5.4.3 The density-based gathering model

To introduce the model, we need first to redefine a few concepts of density-based clustering. Let ϵ_g be the radius of the (gathering) neighborhood defined with respect to a given distance function, and N_g the minimum number of point types that shall appear in a ϵ_g-neighborhood. In addition, consider the *time window* $\Delta_g \geq 0$. We re-define a core point as a point surrounded by a sufficient number of points of different types. Given e_g and N_g, let us denote $\epsilon_g(q)$ the neighborhood of a generic point $q = (p, t)$ of type $l \in D$.

Definition 5.2 (Core point and border point)**.** *We say that q is a core point if the neighborhood $\epsilon_g(q)$ satisfies the following two conditions:*

- *$\epsilon_g(q)$ contains points of at least $N_g - 1$ types different from l*
- *the temporal distance between every point in $\epsilon_g(q)$ and q does not exceed the time window, i.e. $\forall q_i \in \epsilon_g(q), |t_i - t| \leq \Delta_g$.*

The points that belong to the neighborhood $\epsilon_g(q)$ of a point q, but that are not core points, are border points.



(a) (b) (c)

Figure 5.5: (a-b) The circles delimit the ϵ_g- neighborhood of point q. In (a) q is core point, while in (b) it is not. The color indicates the point type. (c) The outer circle G encloses a 4-gathering

For example, Figure5.5 shows an example of ϵ_g-neighborhood. Consider $N_g = 4$ and assume that the maximum temporal distance between the points in the circle does not exceed the parameter Δ_g. In Figure5.5.(a) the point q is a core point because the points in the area $\epsilon_g(q)$ are of 3 different types. In Figure5.5.(b), the point q is not a core point. Note that, in general, the neighborhood of a core point can contain points of equal type. Moreover if $\Delta_g = 0$ then the points in $\epsilon_g(q)$ are reported at the same instant, thus if a type has some instance in the area, such an instance is unique.

A gathering is basically a cluster of points that not only are dense, but also diverse. Also this gathering must have a duration longer than a defined threshold, if specified. Following DBSCAN [35], we define a gathering as a cluster consisting of density-connected core and border points. In more rigorous terms:

Definition 5.3 (Gathering)**.** *A gathering is a maximal set of density-connected points.*

5.4.4 The k-Gathering algorithm

The algorithm for the discovery of gatherings of at least k objects is reported below. In Step 1 the SeqScan is run on the dataset to extract the set of stay regions. Afterwards, in Step 2 the gathering discovery process is performed. The set Q of points in input results from the union set of the stay regions $\{\mathcal{S}^i\}_{i\in[1,n]}$ obtained at the previous step. The points are represented as pairs *(coord,id)* where *coord* are the spatio-temporal coordinates of the point and *id* the point type (i.e. the trajectory identifier). The algorithm scans the typed points of the input set and for every point checks whether it is surrounded in **space** and **time** by a sufficient number of points of different types, i.e. is a core point (performed by the function *FindNeighbors*). If so the points of the neighborhood are pushed onto a stack and the process iterates over the points of the stack. For each cluster found, if its duration is $\geq \Gamma$ then it is a gathering, and it is added to the set of gatherings $setGatherings$, the output.

Concerning the complexity of K-Gathering, it is dependent on the one of SeqScan described in the previous chapter. If we have m trajectories of n points each, we have to repeat SeqScan for each trajectory in order to get their stay regions. And then for the obtained points we apply the step 2 processing of the algorithm, in the worst case the complexity is $O((n.m)^2$, but practically this will not be the case, since the points will be sorted by temporal order, and in an iteration a point p will be compared only to the points falling in the temporal interval $[t_p - \Delta_g, t_g + \Delta_g]$. Also the same distance measuring method used in Seqscan is implemented as well for K-Gathering which will reduce the computation costs.

Algorithm 3 K-Gathering

Input:
$D = \{tr_1, ..tr_m\}$ *//dataset*
ϵ, N, δ *// SeqScan parameters*
$\epsilon_g, N_g, \Delta_g$ *// Gathering parameters*
Γ *// minimum duration of a gathering*
Output: *setGatherings // set of gatherings*

StayRegions=$\emptyset$; Q=$\emptyset$
//STEP 1
for tr_i in D **do**
    StayRegions=Add(SeqScan($tr_i, \epsilon, N, \delta$))
Q=Union(StayRegions)
//STEP 2
for q in Q **do**
    if q not in a cluster **then**
        $E_g(q)$= FindNeighbors(q,ϵ_g, Δ)
        if |different point types in $E_g(q)$| $\geq N_g$ **then**
            currentCluster = new Cluster
            add all points in $E_g(q)$ to *currentCluster*
            push the points in $E_g(q)$ into a stack s
            while s not empty **do**
                $o = s$.pop()
                $E_g(o)$= FindNeighbors(q,ϵ_g, Δ)
                if |different point types in $E_g(o)$| $\geq N_g$ **then**
                    for u in $E_g(o)$ **do**
                        if u not noise & not in a cluster **then**
                            add u to *currentCluster*
                            push u onto s
            if *currentCluster* not empty **then**
                d=FindDuration(*currentCluster*)
                if $d \geq \Gamma$ **then**
                    setGatherings.add(*currentCluster*)
    else
        Mark q as noise

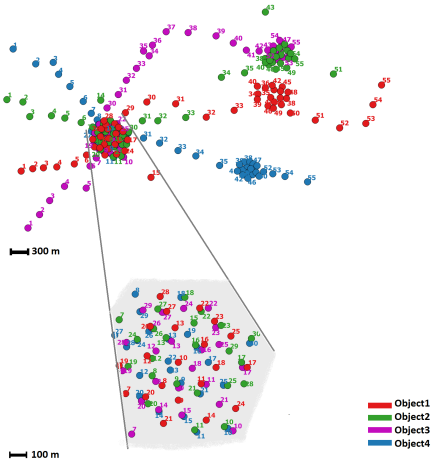


Figure 5.6: Example of static gathering. There are 4 trajectories containing sub-trajectories representing stay regions. Each point is timestamped (from 1 to 55). The zoomed region highlights the static gathering

5.5 Experiments

This section reports the experiments conducted for evaluating the effectiveness of the k-Gathering algorithm. We have performed two main experiments running the algorithm on two different datasets. The first dataset consists of synthetic trajectories. The purpose of the experiment on this dataset is to illustrate the pattern discovery process in different and controlled scenarios. The second dataset consists of real trajectories reporting the movement of a group of animals (roe deer) observed for nearly 1 year. Those trajectories have been collected at low sampling rate and contain noise points. The purpose of the experiment on this dataset is to analyze the behavior of the algorithm on complex structured trajectories. The *k-Gathering* algorithm has been implemented in Python using the Quantum GIS platform. It is added as a part of the MigrO system [28] , this latter will be described in the next chapter.

5.5.1 Experiment 1: effectiveness on synthetic data

We have created two sets of synthetic trajectories, to exemplify a static gathering (Dataset 1) ,and a dynamic gathering (Dataset 2), respectively. In both cases, the sampling rate is fixed. The difference in time between two consecutive points is referred to as *time unit* in the following. Figure5.6 exemplifies the static gathering. The example reports the trajectories of 4 objects that for a certain time remain inside a region moving randomly inside it.

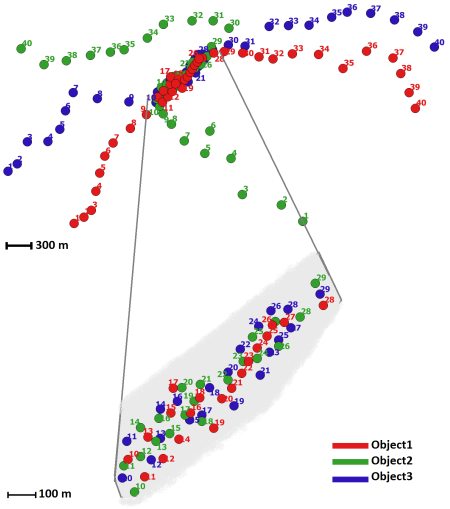
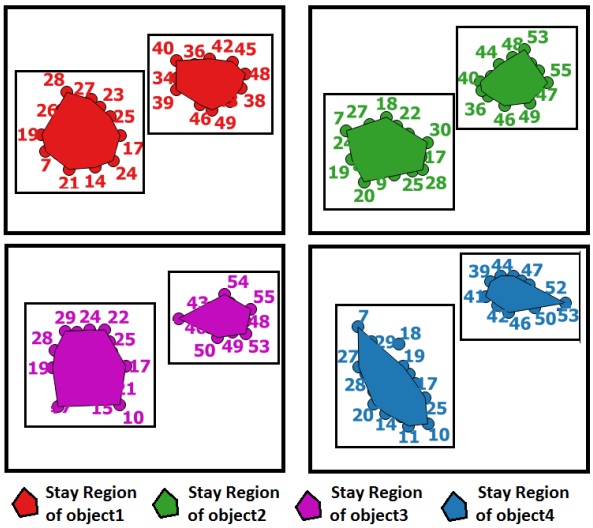


Figure 5.7: Example of dynamic gathering. There are three trajectories of timestamped points from 1 to 40. The zoomed area highlights the dynamic gathering.

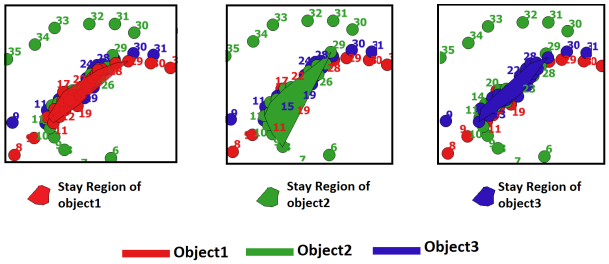
Figure5.7 shows the trajectories of 3 objects that proceed together for a certain time forming a dynamic gathering. In this example, the points are of three different types, moreover at each instant, every object is close to other two forming a clique (i.e. every point is close in space and time to points of at least two different types).

Phase 1: extraction of stay regions

We run the SeqScan algorithm on the trajectories of the two datasets. We obtain the stay regions in Figure5.8.The algorithm is run with the *presence* parameter set to 3 time units. Intuitively, it means that a stay region is recognized as such if the individual remains inside the region at least 3 time units excluding the periods of absence. The density parameters ϵ, N are determined through an iterative process. From Figure5.8.(a), it can be seen that the 4 trajectories consists of 2 stay regions each; Figure5.8.(b) shows that the 3 trajectories contain only one stay region. We emphasize that SeqScan can recognize stay regions that "move".



(a) Stay regions for the 4 trajectories in Dataset 1

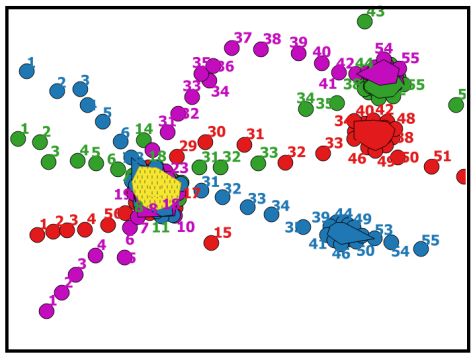


(b) Stay regions for the 3 trajectories in Dataset 2

Figure 5.8: Stay regions extracted from the synthetic trajectories using SeqScan.

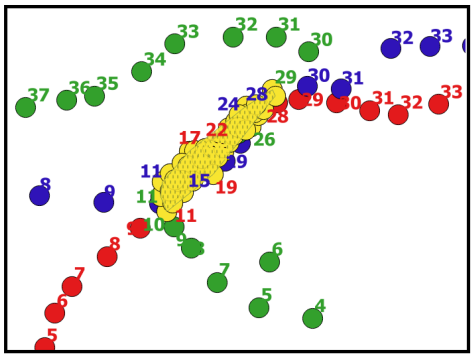
Phase 2: gathering discovery

The gathering discovering process is performed over the set of points obtained from the union set of the stay regions obtained at previous step. The following parameters are used: $\epsilon_g = \epsilon$, minimum number of point types $N_g = 3$ and maximum temporal distance $\Delta_g = 1$ time unit, i.e. two points are "close" in time if they have identical or consecutive timestamps. The minimum gathering duration threshold is set to 0. As a result, we obtain the gatherings reported in Figure5.9. In both examples, the gatherings consist of points correctly typed and located in the expected location.



Gathering id	# Points	Participants
1	63	obj1, obj2, obj3, obj4

(a) Static gathering



Gathering id	# Points	Participants
1	51	obj1, obj2, obj3

(b) Dynamic gathering

Figure 5.9: Outcome of k-Gathering over the synthetic trajectories of the example. The gatherings are approximated by a polygon

5.5.2 Experiment 2: evaluating the effectiveness over real data

The dataset contains the trajectories of 10 roe deer located in Northern Italy. The animals, equipped with a GPS receiver mounted on a collar, have been monitored for more than one year. This dataset is structurally complex. Trajectories are collected at low and irregular sampling rate, i.e. the nominal sampling rate is 1 point every 4 hours. Moreover the data is noisy containing both wrong positions, i.e. GPS errors, and points describing excursions of animals outside their home-ranges.

The points of the trajectories are shown in Figure5.10. A further description of this data is given in the Table5.1. The trajectories are taken from the dataset employed in [27]. As it can be seen, these trajectories form a conglomerate of points. The question is whether such a conglomerate contains a k-Gathering for some value of k.

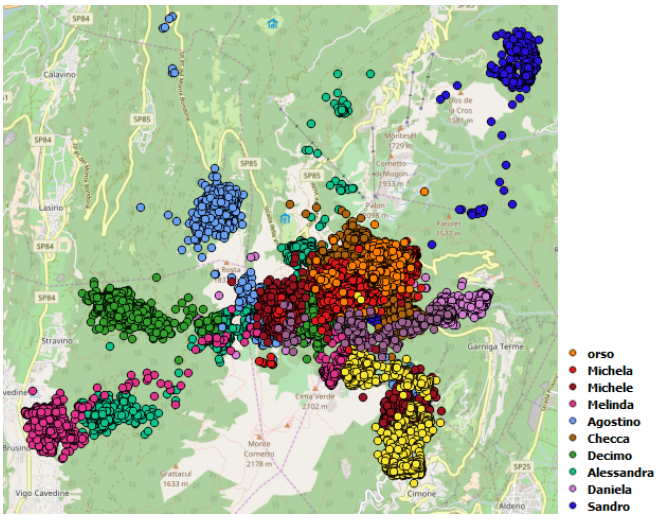


Figure 5.10: Trajectories of 10 GPS-equipped animals. Animals are associated with a nickname and a color

Animal name	Animal id	Number of points	First timestamp	last timestamp
Sondro	774	4094	23/10/2005 20:00	16/12/2007 12:00
Orso Federico	775	890	05/11/2005 12:01	16/08/2006 14:00
Michele	772	2373	20/03/2005 16:02	09/10/2006 10:03
Michela	792	4606	03/11/2005 20:01	02/03/2008 08:01
Melinda	797	2337	27/02/2005 16:03	18/04/2006 04:00
Decimo	791	2695	13/11/2006 00:02	15/03/2008 08:01
Daniela	768	1649	18/10/2005 20:00	29/10/2006 12:00
Checca	767	1365	15/10/2005 20:03	08/12/2006 04:01
Alessandra	782	2642	21/10/2005 20:00	09/02/2007 08:11
Agostino	771	2196	20/03/2005 16:03	27/05/2006 16:02

Table 5.1: Statistics describing the real dataset

As before, we first present the intermediate result obtained from *SeqScan*, and next the gatherings.

Phase 1: extraction of stay regions

Figure5.11 shows the set of stay regions obtained running SeqScan with the following parameters: $\epsilon = 200$m., $N = 20$ points and presence $\delta = 20$ days. The animals thus shall remain inside the stay region for at least 20 days, at the net of absence periods.

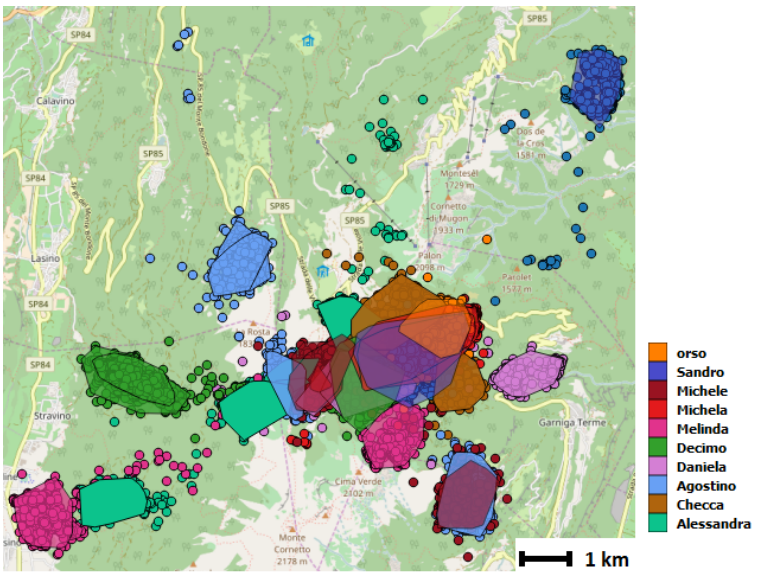
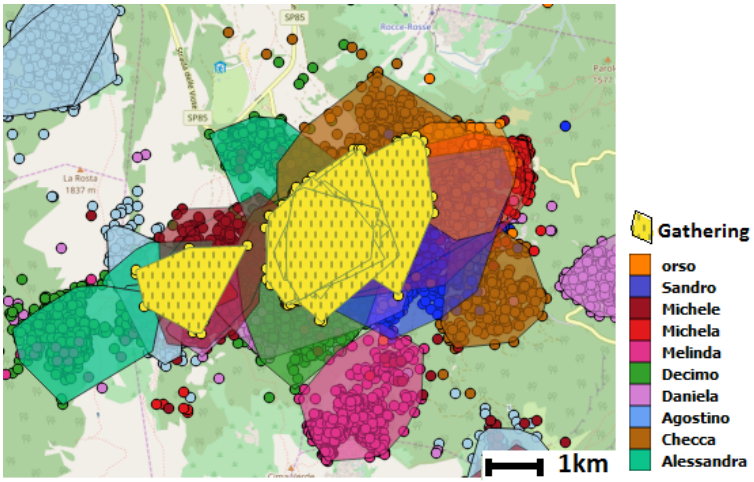


Figure 5.11: The stay regions, represented by polygons, obtained running *SeqScan* on every trajectory. The different colors are associated with the different animals

Phase 2: gathering discovery

In the second phase, to detect the gatherings, we use the following parameters: $\epsilon_g = 500m.$, $N_g = 3$, $\Delta_g = 6$ hours. Thus two animals are considered close in time if the temporal distance is at maximum of 6 hours. The resulting gatherings are represented by the polygons (convex hulls) in Figure5.12. The associated Table provides further details. There are 10 gatherings of 3, 4, 5 animals. Every gathering lasts a few days. It can be seen that one group gathers 6 times. For this group of animals, we have generated the spatio-temporal representation of the trajectories, as shown in Figure5.13. It can be seen - although in more qualitative than quantitative terms - that these trajectories are close in space and time. The presence of 6 gatherings is thus compatible with the visual analysis.



Gathering id	Duration	Participants
1	3 days	Michele, Daniela, Alessandra
2	2 days	Sandro , Orso, Michela
3	1 day	Sandro, Michela, Daniela, Checca, Alessandra
4	1 day	Sandro, Michela, Checca, Alessandra
5	3 days	Sandro , Michela, Decimo
6	2 days	Sandro, Michela, Decimo
7	4 days	Sondro, Michela, Decimo
8	4 days	Sandro , Michela, Decimo
9	2 days	Sandro, Michela, Decimo
10	4 days	Sandro , Michela, Decimo

Figure 5.12: Gatherings obtained running k-Gathering with parameters: $\epsilon_g = 500m., N_g = 3, \Delta_g = 6h$

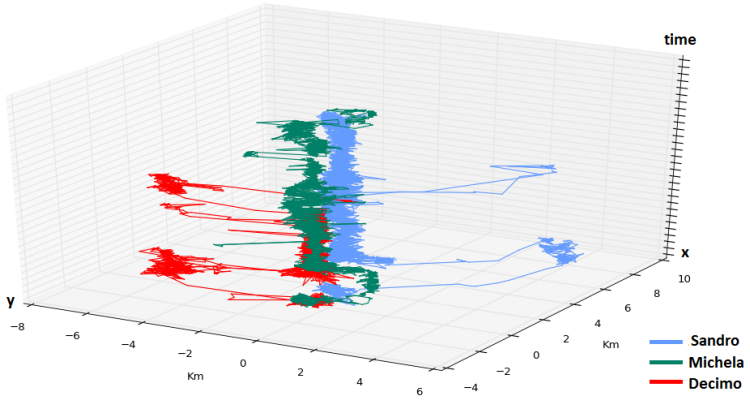


Figure 5.13: Spatio-temporal representation of the trajectories of the 3 animals that most frequently form a gathering

5.6 Concluding remarks

In this chapter, we have presented the k-Gathering technique for the discovery of gatherings of at least k-objects. K-Gathering leverages the individual stops detected by the SeqScan technique. Beyond the parameters requested by SeqScan, the k-Gathering algorithm requires 4 parameters for the detection of gatherings against the 6 parameters requested by [95]. This translates in higher usability.

As for the experiments, in the first one we have used synthetic data and we have illustrated the gathering discovery in different scenarios, our technique has succeeded to detect the gatherings in the expected locations. Also, in the second experiment that has been performed on real data presenting complex structured trajectories, our technique has discovered the gatherings, which are visually compatible with the trajectories analysis.

It is important to highlight the fact that the deployment of the k-Gathering method for the analysis of low sampling-rate trajectories raises interesting issues. Such trajectories are typically not temporally synchronized, i.e. the points are not acquired at the same time. Moreover the temporal gap between consecutive locations can be significantly large and thus the missing points cannot be reasonably obtained by interpolation. In such a setting, the notion of gathering as "staying in the same place at the same time" is blurred because the points at "the same time" cannot be estimated with sufficient accuracy. On the other hand, the techniques presented in literature assume the movement to be nearly continuous, thus are not able to address the challenges posed by this scenario.

Chapter 6

The MigrO system

6.1 Overview

MigrO is a software platform for the extraction of mobility patterns from GPS trajectories based on SeqScan. Initially developed to support the discovery of *stay regions*, it has been subsequently extended to support the discovery of *k-Gatherings* (Chapter 5). It consists of various tools, which have been employed for the evaluation of SeqScan, and also to facilitate the interaction with the domain experts. In this Chapter we overview the architecture of the system and present a use case. The dataset employed for the case study is minimal, but important. It consists of one trajectory recently provided by a biologist from U.S. Geological Survey, based at Las Vegas, to test the potentialities of MigrO for the study of the movement of desert tortoises, a species of turtles living in desert areas.

This chapter is divided in two parts: the first part describes the architecture of the MigrO system. The second part is the usage scenario. Most of the materials reported in this Chapter has been published in [28].

6.2 The MigrO architecture

MigrO is developed in Python as plug-in for the open-source Quantum GIS (QGIS) system (Figure6.1). QGIS is a cross-platform, free and open source software that supports viewing, editing and analysis of geospatial data [1]. MigrO exploits the rich set of functionalities of the hosting system, to offer a powerful platform for the analysis of the mobility behavior. The layered architecture is shown in Figure 6.2. The system requires in input sequences of timestamped points, and returns sequences of stay regions. Stay regions can take the form of labeled points or, in

[1]For more details, the official website is: http://www.qgis.org

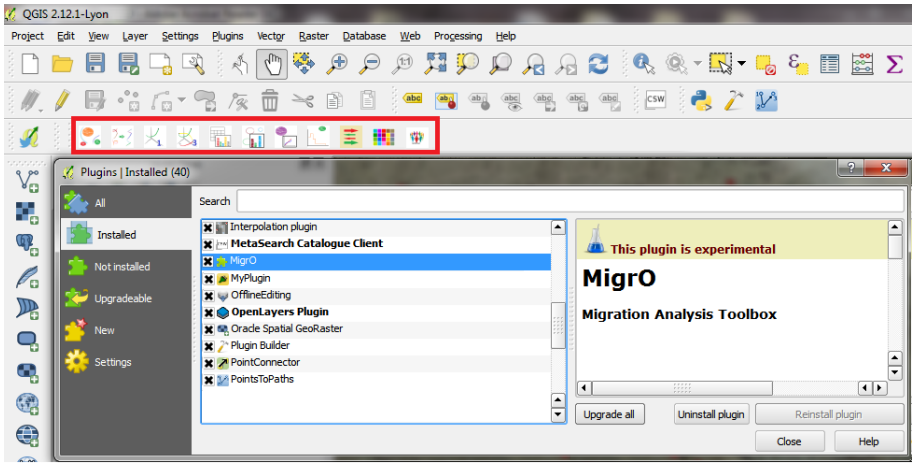


Figure 6.1: MigrO plug-in in QGIS V2.12.1: the red rectangle indicates the MigrO icons

.

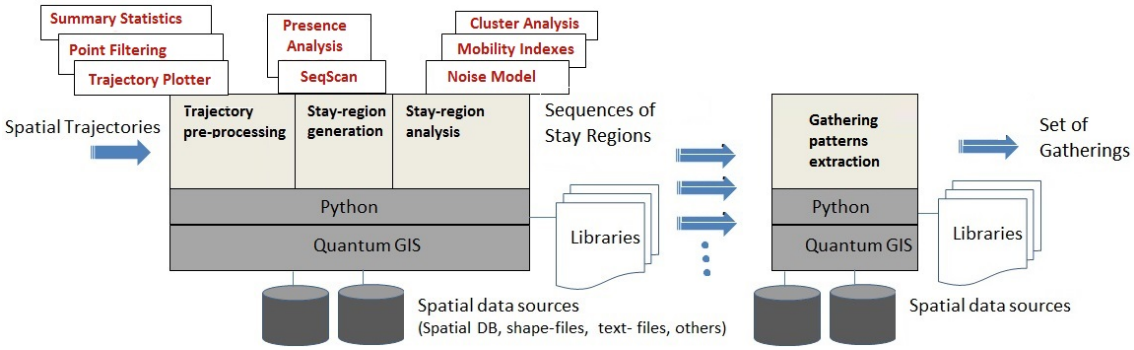


Figure 6.2: MigrO architecture

.

alternative, of symbolic trajectories [49]. The sequences of stay regions obtained from the clustering of multiple trajectories provides the input for the discovery of k-Gathering patterns.

The trajectory data analysis consists of three main phases: trajectory pre-processing, spatio-temporal clustering for stay region generation, and stay region analysis for the assessment of the clustering outcome. Each phase is supported by a number of tools briefly presented in the following. An optional supplementary phase is for the discovery of gatherings.

6.2.1 Phase 1: preprocessing of trajectories

The preprocessing of trajectories has a twofold goal: to provide insights into the data, and to support data cleaning. For this purpose, three tools are available: the trajectory plotter, summary statistics and trajectory filter. These tools are briefly described in the following.

- Trajectory plotter. The trajectory plotter displays the single trajectory as space-time cube enabling the observation of the macro-characteristics of the object's movement.

- Summary statistics. This tool provides statistics on the trajectories dataset, such as the number of points, time and space extents, step length (i.e. distance between two consecutive samples) in space and in time. Statistics are especially important for the choice of the clustering parameters.

- Trajectory filter. This tool filters out the highest sampling frequencies. We recall that the sampling rate has an impact on the clustering operation, therefore controlling the temporal step length is fundamental. Moreover, point filtering can be beneficial for the performance of the clustering algorithm.

6.2.2 Phase 2: stay regions generation

This module contains two tools, for running SeqScan and to to perform the presence analysis.

- Presence analysis. This tool is to facilitate the choice of the δ parameter (i.e. presence threshold). A graph reports the relation between the number of stay regions that can be obtained with a given set of density parameters, and δ. This helps determine the desired level of temporal granularity and therefore the adequate value of δ. It implements the function f_T described in section 3.4.3.

- SeqScan. This is the core component of MigrO, it implements the SeqScan algorithm returning in output the sequence of stay regions.

6.2.3 Phase 3: stay region analysis

A number of tools can be used to analyze the result of the SeqScan clustering:

- Noise analysis. This tool extracts and visualizes local noise and transitions.

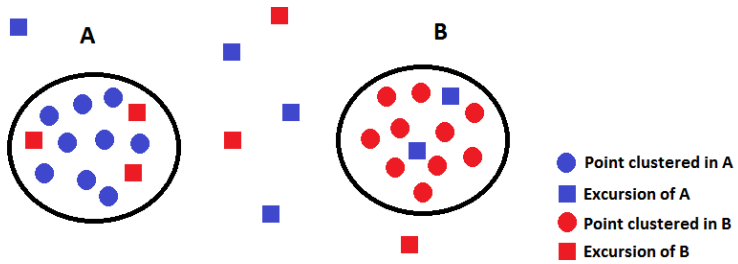


Figure 6.3: Community Index CI. Two stay regions of same object trajectory, A and B with local noise points. Total number of local noise points of A is 6, only 2 falls in the spatial extent of B, hence $CI_{AB} = 0.3$.

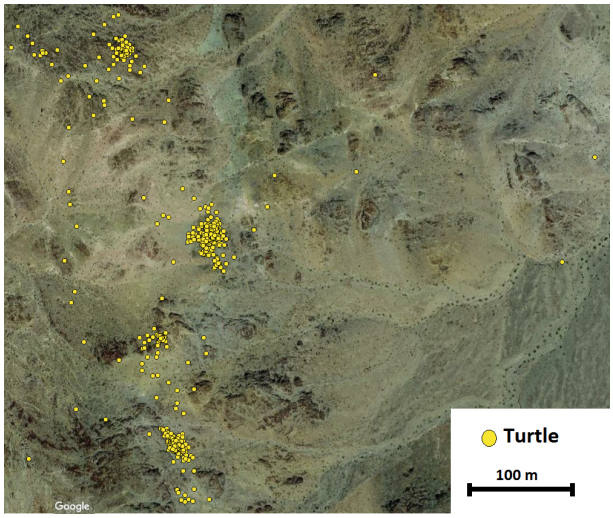
- Mobility indexes. Two indexes, defined in previous work [29], are made available for the evaluation of clustering, called Stationarity index and Commuting index, respectively. *Stationarity index* is the ratio of presence over the duration of a cluster. It indicates the percentage of time the object has effectively spent inside the stay region. *Commuting index* CI helps interpreting the relation between two stay regions. It's a measure of the absences, an object located in the stay region A has experienced to visit stay region B. An example is shown in Figure 6.3.

- Post Statistics. This tool presents some statistics about the resulting clusters, such as the number of clustered points, the spatial distance between the clustered points, the temporal extent of clusters.

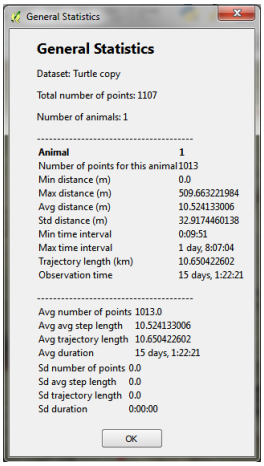
6.3 Usage scenario

The cluster analysis is performed over a trajectory describing the movement of a desert tortoise. The tortoise is equipped with a GPS logger. The location is sampled every 10 minutes from 04:00 to 22:00 daily. The temporal extent of the trajectory is approximately 2 weeks. The dataset suffers from missing points and location errors due to the tortoise's frequent use of deep burrows that block GPS signals.

These characteristics of the data can be detected using the Summary Statistics tool reporting the distribution of the sampling intervals. For example, Figure 6.4(b) shows that the maximum temporal interval between 2 consecutive points is 1 day. Figure6.4(a) shows the trajectory on the map. We can notice that the spatial extent of the trajectory is about few hundreds of meters (maximum distance between 2 points is 509 m according to the statistics).



(a) The sample points of the trajectory (yellow dots). The animal is located next to Soda mountains, in California U.S.A, the whole movement is limited to few hundreds of meters.



(b) Summary statistics

Figure 6.4: The animal trajectory and summary statistics.

The movement can be qualitatively analyzed using the plot in Figure 6.5. By analyzing this plot we can see that there could be at least 3 stay regions of relatively large temporal extent and a shorter stay region.

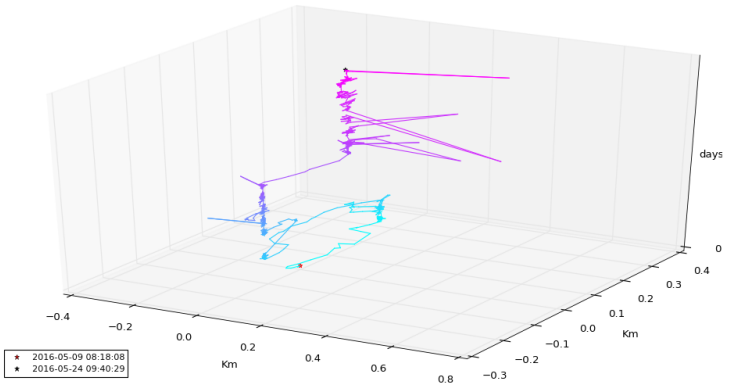


Figure 6.5: Spatio-temporal representation of the trajectory. The spikes that can be observed are likely due to GPS errors

The graph resulting from the presence analysis in Figure 6.6 confirms that with low values of the parameter δ, SeqScan can find 5 stay regions, but only 3 with increased value of presence.

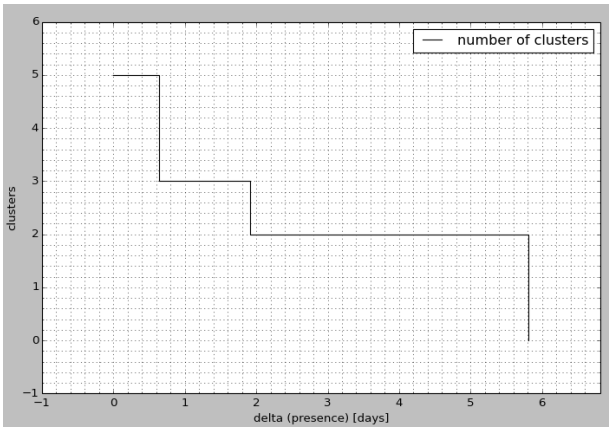


Figure 6.6: Analysis of the presence parameter

We run SeqScan algorithm using as density parameters $\epsilon = 20m$ and n=20 points and set $\delta = 0$. The outcome is displayed in Figure 6.7. The stay regions of the sequence are progressively numbered. It is easy to distinguish the local noise from the transition. The stay regions 2 and 3 do not have local noise, therefore the animal remains inside the cluster for the whole period. Hence the Stationarity Index is maximal in these two clusters, as demonstrated in Figure 6.8(a). Stay regions 1 and 5 have local noise.

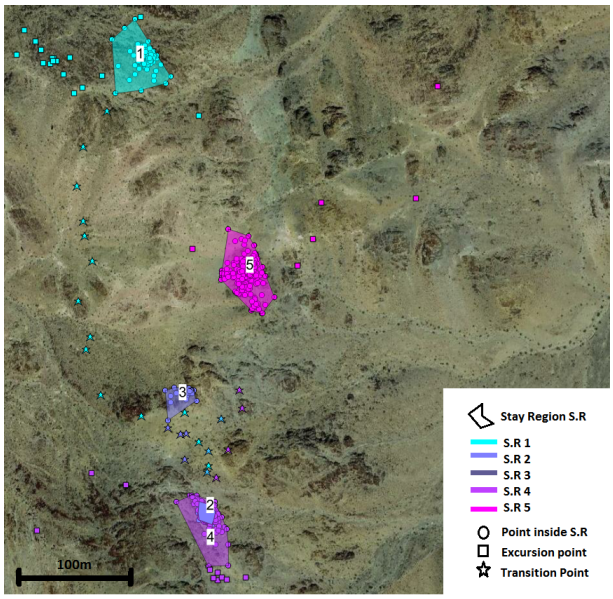


Figure 6.7: Stay regions detected by SeqScan. The parameters of SeqScan used for this case are: $\epsilon = 20m$, $n = 20points$ and $presence = 0days$. Five stay regions are obtained, annotated accordingly.

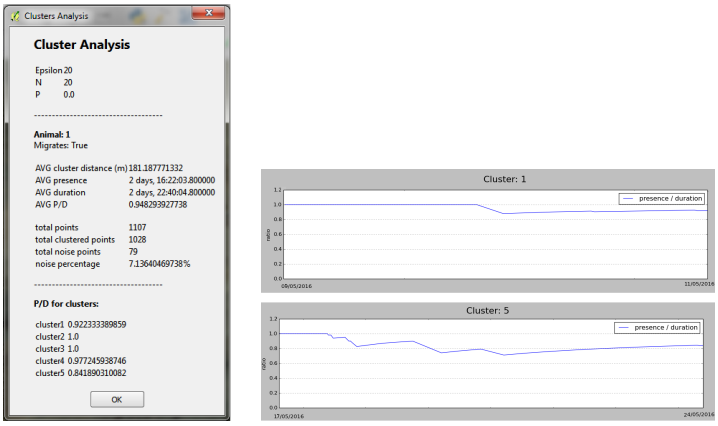


Figure 6.8: Stay regions analysis.(a) Post statistics and (b) graphs of the Stationary Index for regions 1 and 5.

Chapter 7

Conclusion

In this chapter we will briefly summarize the main contribution of this thesis, the lessons learned and discuss possible directions for future work.

7.1 Main contributions

Mobility data is a critical asset for the study of natural phenomena in diverse scientific disciplines, including ecology, as well as an enabling factor for the development of advanced location-aware applications and services. In such a context, the development of analytical techniques especially targeting the extraction of behavioral information from spatial trajectories is a key task, which paves the way to the creation of competitive advantage for both scientific and business organizations. The research conducted in this Thesis falls in such an area, broadly known as *mobility data analytics*. The main contributions of this research can be summarized as follows:

- A rigorous analytical framework has been defined for studying the properties of the recently proposed SeqScan algorithm. The definition of a formal ground has been extremely important for the discovery of new properties such as the distinction between strong and weak spatial separation of clusters. Moreover it has streamlined the transfer of the method to domain specialists.

- An extensive evaluation has been conducted for confronting the clustering with ground truth in an interdisciplinary setting. That has requested a tight collaboration with domain experts, especially to achieve a common understanding of the validation goals and metrics. An interesting issue, which emerged along the way, is the need of validation metrics specifically defined for sequences. This issue is left for future work.

- SeqScan framework has been next applied for the study of additional patterns, such as the gathering pattern, which extends the notion of stop to that of group. This work is at the beginning and can be further extended in future.

- A sw platform is available, providing a set of tools and visualization facilities that can help the data analyst. Clustering analysis is not simply a clustering algorithm, it requires a sw platform including visualization and statistical tools.

7.2 Directions for the future

We have already mentioned the need of quality indexes for evaluating the outcome of SeqScan. Besides external indexes, another set of useful indicators are the internal indexes. Internal indexes allow the evaluation of clustering independently from ground truth, therefore are of primary importance to make SeqScan usable. While a first approach to the definition of internal indices has been proposed in previous work, further research is definitely needed. On the other hand, the use of ground truth in the analysis of trajectories is problematic. Real datasets are noisy, domain knowledge is often not sufficiently robust for the detailed point-by-point verification of the clustering, thus the use of synthetic data become the only viable solutions. But even in that case, the generation of synthetic data is not trivial. The manual creation of trajectories exhibiting a significant behavior is time consuming and thus not affordable over large scales. On the other hand, the development of simulators requires a profound understanding of the mobility phenomenon, which typically is what lacks or is only partially available. Even more complicated is validating collective mobility patterns, such as gathering pattern. Beyond validation, the gathering pattern raises a number of interesting questions, such as how to analyze the relation between objects that participate into the same gatherings, either together or at different time periods. This problem is strictly related to that of time evolving communities, which is another topic left for future work.

Concluding, the results of the Thesis are mature and pave the way to the deployment of the SeqScan technique. The area of ecology is probably the one that most straightforwardly can benefit of the SeqScan framework. Other applications areas can be envisaged. For example, another case study developed during the research regards the analysis of the eye gaze traces, in particular to detect fixations, i.e. the attracting areas where the gaze stops for a while. Confronted with a state-of-the-art technology, the results are interesting, however the lack of robust validation data has greatly hampered the evaluation. Another application con-

cerns the analysis of human mobility through check-ins data. This is a promising application area that can provide useful hints on human mobility.

Bibliography

[1] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for clustering evolving data streams. In *Proc. VLDB*, 2003.

[2] Louai Alarabi. ST-Hadoop: A MapReduce Framework for Big Spatio-temporal Data. In *Proc. of the ACM International Conference on Management of Data*, 2017.

[3] Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information retrieval*, 12(4):461–486, 2009.

[4] Mattias Andersson, Joachim Gudmundsson, Patrick Laube, and Thomas Wolle. Reporting leadership patterns among trajectories. In *Proc. ACM Symposium on Applied Computing*. ACM, 2007.

[5] Mattias Andersson, Joachim Gudmundsson, Patrick Laube, and Thomas Wolle. Reporting leaders and followers among trajectories of moving point objects. *GeoInformatica*, 12(4):497–528, 2008.

[6] Gennady Andrienko, Nathaliya Andrienko, Peter Bak, Daniel Keim, Slava Kisilevich, and Stefan Wrobel. A conceptual framework and taxonomy of techniques for analyzing movement. *Journal of Visual Languages & Computing*, 22(3):213–232, 2011.

[7] Mihael Ankerst, Markus M. Breunig, Hans peter Kriegel, and Jrg Sander. Optics: Ordering points to identify the clustering structure. In *Proc. ACM SIGMOD international conference on Management of data.* ACM Press, 1999.

[8] Boris Aronov, Anne Driemel, Marc Van Kreveld, Maarten Löffler, and Frank Staals. Segmentation of trajectories on nonmonotone criteria. *Transactions on Algorithms (TALG)*, 12(2):26, 2016.

[9] Sugato Basu, Arindam Banerjee, and Raymond J Mooney. Active semi-supervision for pairwise constrained clustering. In *Proc. SIAM international conference on data mining*, pages 333–344. SIAM, 2004.

[10] Marc Benkert, Joachim Gudmundsson, Florian Hübner, and Thomas Wolle. Reporting flock patterns. *Computational Geometry*, 41(3):111–125, 2008.

[11] O. Berger-Tal and S. Bar-David. Recursive movement patterns: review and synthesis across species. *Ecosphere*, 6(9):1–12, 2015.

[12] Pavel Berkhin et al. A survey of clustering data mining techniques. *Grouping multidimensional data*, 25:71, 2006.

[13] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. *When Is "Nearest Neighbor" Meaningful?*, chapter in: International Conference on Database Theory (ICDT99), pages 217–235. Lecture Notes in Computer Science. Springer, 1999.

[14] Derya Birant and Alp Kut. St-dbscan: An algorithm for clustering spatial–temporal data. *Data & Knowledge Engineering*, 60(1):208–221, 2007.

[15] Igo Ramalho Brilhante, Michele Berlingerio, Roberto Trasarti, Chiara Renso, Jose Antonio Fernandes de Macedo, and Marco Antonio Casanova. Cometogether: Discovering communities of places in mobility data. In *Mobile Data Management (MDM), 2012 IEEE 13th International Conference on*. IEEE, 2012.

[16] Maike Buchin, Anne Driemel, Marc J. van Kreveld, and Vera Sacristán Adinolfi. Segmenting trajectories: A framework and algorithms using spatiotemporal criteria. *J. Spatial Information Science*, 3:33–63, 2011.

[17] Maike Buchin, Helmut Kruckenberg, and Andrea Kölzsch. *Segmenting Trajectories by Movement States*, chapter Advances in Spatial Data Handling, pages 15–25. Springer, 2013.

[18] Feng Cao, Martin Estert, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In *Proc. SIAM international conference on data mining*. SIAM, 2006.

[19] H. Cao, N. Mamoulis, and D W. Cheung. Discovery of periodic patterns in spatiotemporal sequences. *IEEE Trans. on Knowl. and Data Eng.*, 19(4):453–467, 2007.

[20] Varun Chandola and Vipin Kumar. Summarization – compressing data into an informative representation. *Knowledge and Information Systems*, 12(3):355–378, 2007.

[21] Yun Chi, Xiaodan Song, Dengyong Zhou, Koji Hino, and Belle L Tseng. Evolutionary spectral clustering by incorporating temporal smoothness. In *Proc. ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007.

[22] Fernando Chirigati, Harish Doraiswamy, Theodoros Damoulas, and Juliana Freire. Data polygamy: The many-many relationships among urban spatio-temporal data sets. In *Proc. of the International Conference on Management of Data*, SIGMOD '16, 2016.

[23] Michele Coscia, Fosca Giannotti, and Dino Pedreschi. A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 4(5):512–546, 2011.

[24] Philippe Cudre-Mauroux, Eugene Wu, and Samuel Madden. Trajstore: An adaptive storage system for very large trajectory data sets. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*. IEEE, 2010.

[25] Maria Luisa Damiani and Fatima Hachem. Segmentation techniques for the summarization of individual mobility data. *WIREs Data Mining and Knowledge Discovery*, 2017.

[26] Maria Luisa Damiani, Fatima Hachem, Hamza Issa, Nathan Ranc, Paul Moorcroft, and Francesca Cagnacci. Cluster-based trajectory segmentation with local noise. *Data Mining and Knowledge Discovery (DAMI)*, 2018. (under review-minor reviews).

[27] Maria Luisa Damiani, Hamza Issa, and Francesca Cagnacci. Extracting stay regions with uncertain boundaries from GPS trajectories: a case study in animal ecology. In *Proc. SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2014.

[28] Maria Luisa Damiani, Hamza Issa, Giuseppe Fotino, Fatima Hachem, Nathan Ranc, and Francesca Cagnacci. Migro: a plug-in for the analysis of individual mobility behavior based on the stay region model. In *Proc. SIGSPATIAL International Conference on Advances in Geographic Information Systems, USA*, 2015.

[29] Maria Luisa Damiani, Hamza Issa, Giuseppe Fotino, Marco Heurich, and Francesca Cagnacci. Introducing 'presence' and 'stationarity index' to study partial migration patterns: an application of a spatio-temporal clustering technique. *International Journal on Geographical Information Science*, 30(5):907–928, 2016.

[30] S. Dodge, R. Weibel, and A-K Lautenschütz. Towards a taxonomy of movement patterns. *Information Visualization*, 7(3-4):240–252, 2008.

[31] Hendrik Edelhoff, Johannes Signer, and Niko Balkenhol. Path segmentation for beginners: an overview of current methods for detecting changes in animal movement patterns. *Movement Ecology*, 4(1):21, 2016.

[32] M. Elfeky, W. Aref, and A. Elmagarmid. Warp: Time warping for periodicity detection. In *Proc. IEEE ICDM*, ICDM, 2005.

[33] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1):12, 2012.

[34] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Michael Wimmer, and Xiaowei Xu. Incremental clustering for mining in a data warehousing environment. In *Proc. VLDB*, 1998.

[35] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, 1996.

[36] I. Farber, S. Guennemann, H. Kriegel, P. Kroeger, E. Mueller, E. Schubert, T. Seidl, and A. Zimek. On Using Class-Labels in Evaluation of Clusterings. In *Proc. 1st International Workshop on Discovering, Summarizing and Using Multiple Clusterings (MultiClust 2010)*, 2010.

[37] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.

[38] Giuseppe Fotino. Tecniche di clustering per l'analisi di pattern di movimento. Master's thesis, University of Milano, 2015.

[39] Junhao Gan and Yufei Tao. Dbscan revisited: mis-claim, un-fixability, and approximation. In *Proc. ACM SIGMOD International Conference on Management of Data*. ACM, 2015.

[40] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. Trajectory pattern mining. In *Proc. of the 13th ACM SIGKDD*. ACM, 2007.

[41] Fosca Giannotti and Dino Pedreschi. *Mobility, data mining and privacy: Geographic knowledge discovery*. Springer Science & Business Media, 2008.

[42] Jean Dickinson Gibbons and Subhabrata Chakraborti. *Non-parametric Statistical Inference*. CRC Press, 2003.

[43] Johannes De Groeve, Nico Van de Weghe, Nathan Ranc, Tijs Neutens, Lino Ometto, Omar Rota-Stabelli, and Francesca Cagnacci. Extracting spatio-temporal patterns in animal trajectories: an ecological application of sequence analysis methods. *Methods in Ecology and Evolution*, 7(3):369–379, 2016.

[44] Joachim Gudmundsson, Patrick Laube, and Thomas Wolle. Movement patterns in spatio-temporal data. *Encyclopedia of GIS*, 726:732, 2008.

[45] Joachim Gudmundsson and Marc van Kreveld. Computing longest duration flocks in trajectory data. In *Proc. ACM International Symposium on Advances in geographic information systems*. ACM, 2006.

[46] Eliezer Gurarie, Russel D Andrews, and Kristin L Laidre. A novel method for identifying behavioural changes in animal movement data. *Ecology letters*, 12(5):395–408, 2009.

[47] R H. Güting, M.H. Böhlen, Martin Erwig, C.S. Jensen, N.A. Lorentzos, M. Schneider, and M.Vazirgiannis. A foundation for representing and querying moving objects. *ACM Transactions on Database Systems*, 25(1):1–42, 2000.

[48] Ralf H. Güting, Thomas Behr, and Christian Düntgen. *SECONDO: a Platform for Moving Objects Database Research and for Publishing and Integrating Research Implementations*. Fernuniv., Fak. fr Mathematik u. Informatik, 2010.

[49] R.H. Güting, F. Valdés, and M. L. Damiani. Symbolic trajectories. *ACM Transactions on Spatial Algorithms and Systems*, 1(2):7:1–7:51, 2015.

[50] Fatima Hachem, Maria Luisa Damiani, and Hamza Issa. Discovering gatherings based on individual mobility patterns: challenges and direction. In *IEEE ICDM Workshop*, 2017.

[51] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[52] Andreas Hotho, Steffen Staab, and Gerd Stumme. Ontologies improve text document clustering. In *Proc. IEEE ICDM*. IEEE, 2003.

[53] Yan Huang, Cai Chen, and Pinliang Dong. Modeling herds and their evolvements from trajectory data. In *International Conference on Geographic Information Science*. Springer, 2008.

[54] Yan Huang, Shashi Shekhar, and Hui Xiong. Discovering colocation patterns from spatial data sets: a general approach. *IEEE Transactions on Knowledge and data engineering*, 16(12):1472–1485, 2004.

[55] Hamza Issa. *Spatio-textual trajectories: models and applications*. PhD thesis, University of Milan, Department of Computer Sciences, 2016.

[56] Hamza Issa and Maria Luisa Damiani. Efficient access to temporally overlaying spatial and textual trajectories. In *IEEE International Conference on Mobile Data Management (MDM)*, 2016.

[57] Hoyoung Jeung, Heng Tao Shen, and Xiaofang Zhou. Convoy queries in spatio-temporal databases. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*. IEEE, 2008.

[58] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S Jensen, and Heng Tao Shen. Discovery of convoys in trajectory databases. *Proceedings of the VLDB Endowment*, 1(1):1068–1080, 2008.

[59] Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. On discovering moving clusters in spatio-temporal data. In *International Symposium on Spatial and Temporal Databases*. Springer, 2005.

[60] Jong Hee Kang, William Welbourne, Benjamin Stewart, and Gaetano Borriello. Extracting places from traces of locations. In *Proc. of the ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*, 2004.

[61] 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, 2001.

[62] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. An online algorithm for segmenting time series. In *Proc. IEEE ICDM*. IEEE, 2001.

[63] Min-Soo Kim and Jiawei Han. Chronicle: A two-stage density-based clustering algorithm for dynamic networks. In *Discovery Science*. Springer, 2009.

[64] Christian Komusiewicz, Falk Hüffner, Hannes Moser, and Rolf Niedermeier. Isolation concepts for efficiently enumerating dense subgraphs. *Theoretical Computer Science*, 410(38-40):3640–3654, 2009.

[65] Dimitrios Kotsakos, Goce Trajcevski, Dimitrios Gunopulos, and Charu C. Aggarwal. Time-series data clustering. In *Data Clustering: Algorithms and Applications*. CRC Press, 2013.

[66] Hardy Kremer, Philipp Kranen, Timm Jansen, Thomas Seidl, Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. An effective evaluation measure for clustering on evolving data streams. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2011.

[67] Hans-Peter Kriegel, Peer Kröger, Jörg Sander, and Arthur Zimek. Density-based clustering. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, 1(3):231–240, 2011.

[68] John Krumm and Dany Rouhana. Placer: semantic place labels from diary data. In *Proc. ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2013.

[69] Patrick Laube. The low hanging fruit is gone: achievements and challenges of computational movement analysis. *SIGSPATIAL Special*, 7(1):3–10, 2015.

[70] Sune Lehmann, Martin Schwartz, and Lars Kai Hansen. Biclique communities. *Physical Review E*, 78(1):016108, 2008.

[71] Luis A. Leiva and Enrique Vidal. Warped k-means: an algorithm to cluster sequentially-distributed data. *Information Sciences*, 237:196–210, 2013.

[72] Z. Li. and J. Han. *Data Mining and Knowledge Discovery for Big Data. Studies in Big Data*, chapter Mining Periodicity from Dynamic and Incomplete Spatiotemporal Data. Springer, Berlin, Heidelberg, 2014.

[73] Zhenhui Li, Bolin Ding, Jiawei Han, and Roland Kays. Swarm: Mining relaxed temporal moving object clusters. *Proceedings of the VLDB Endowment*, 3(1-2):723–734, 2010.

[74] Zhenhui Li, Jiawei Han, Ming Ji, Lu-An Tang, Yintao Yu, Bolin Ding, Jae-Gil Lee, and Roland Kays. Movemine: Mining moving object data for discovery of animal movement patterns. *ACM Trans. Intell. Syst. Technol.*, 2(4):37:1–37:32, July 2011.

[75] Yu-Ru Lin, Yun Chi, Shenghuo Zhu, Hari Sundaram, and Belle L Tseng. Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. In *Proc. international conference on World Wide Web*. ACM, 2008.

[76] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[77] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[78] Andrey Tietbohl Palma, Vania Bogorny, Bart Kuijpers, and Luis Otavio Alvares. A clustering-based approach for discovering interesting places in trajectories. In *Proc. ACM Symposium on Applied Computing*, 2008.

[79] Christine Parent, Stefano Spaccapietra, Chiara Renso, Gennady L. Andrienko, Natalia V. Andrienko, Vania Bogorny, Maria Luisa Damiani, Aris Gkoulalas-Divanis, José Antônio Fernandes de Macêdo, Nikos Pelekis, Yannis Theodoridis, and Zhixian Yan. Semantic trajectories modeling and analysis. *ACM Comput. Surv.*, 45(4):42:1–42:32, 2013.

[80] Slobodan Rasetic, Jörg Sander, James Elding, and Mario A Nascimento. A trajectory splitting model for efficient spatio-temporal indexing. In *Proc. VLDB*. VLDB Endowment, 2005.

[81] Eréndira Rendón, Itzel Abundez, Alejandra Arizmendi, and ElviaM Quiroz. Internal versus external cluster validation indexes. *International Journal of computers and communications*, 5(1):27–34, 2011.

[82] Yehezkel S. Resheff. Online Trajectory Segmentation and Summary With Applications to Visualization and Retrieval. In *Proc. IEEE International Conference on Big Data*, 2016.

[83] Paulo Shakarian, Patrick Roos, Devon Callahan, and Cory Kirk. Mining for geographically disperse communities in social networks by leveraging distance modularity. In *Proc. ACM SIGKDD International Conference on Knowledge discovery and data mining*. ACM, 2013.

[84] Etienne Gael Tajeuna, Mohamed Bouguessa, and Shengrui Wang. Tracking the evolution of community structures in time-evolving social networks. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*. IEEE, 2015.

[85] Lei Tang, Huan Liu, Jianping Zhang, and Zohreh Nazeri. Community evolution in dynamic multi-mode networks. In *Proc. ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008.

[86] Evimaria Terzi and Panayiotis Tsaparas. Efficient algorithms for sequence segmentation. In *Proc. of the Sixth SIAM International Conference on Data Mining, April 20-22, 2006, Bethesda, MD, USA*, 2006.

[87] Yves van Gennip, Blake Hunter, Raymond Ahn, Peter Elliott, Kyle Luh, Megan Halvorson, Shannon Reid, Matthew Valasik, James Wo, George E Tita, et al. Community detection using spectral clustering on sparse geosocial data. *SIAM Journal on Applied Mathematics*, 73(1):67–83, 2013.

[88] Marcos R Vieira, Petko Bakalov, and Vassilis J Tsotras. On-line discovery of flock patterns in spatio-temporal data. In *Proc. ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2009.

[89] Martin Wirz, Pablo Schläpfer, Mikkel Baun Kjærgaard, Daniel Roggen, Sebastian Feese, and Gerhard Tröster. Towards an online detection of pedestrian flocks in urban canyons by smoothed spatio-temporal clustering of gps trajectories. In *Proc. ACM SIGSPATIAL international workshop on location-based social networks*. ACM, 2011.

[90] B.J. Worton. Kernel methods for estimating the utilization distribution in home-range studies. *Ecology*, 1989.

[91] Zhixian Yan, Dipanjan Chakraborty, Christine Parent, Stefano Spaccapietra, and Karl Aberer. SeMiTri: A Framework for Semantic Annotation of Heterogeneous Trajectories. In *Proc. of the International Conference on Extending Database Technology (EDBT)*, 2011.

[92] Jin Soung Yoo and Shashi Shekhar. A joinless approach for mining spatial colocation patterns. *Transactions on Knowledge and Data Engineering*, 18(10):1323–1337, 2006.

[93] Hyunjin Yoon and Cyrus Shahabi. Robust time-referenced segmentation of moving object trajectories. In *Proc. ICDM*, 2008.

[94] Jingjing Zhang, Kathleen M OReilly, George LW Perry, Graeme A Taylor, and Todd E Dennis. Extending the functionality of behavioural change-point analysis with k-means clustering: a case study with the little penguin (eudyptula minor). *PloS one*, 10(4):e0122811, 2015.

[95] Kai Zheng, Yu Zheng, Nicholas J Yuan, Shuo Shang, and Xiaofang Zhou. Online discovery of gathering patterns over trajectories. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1974–1988, 2014.

[96] Yu Zheng. Trajectory data mining: an overview. *Transactions on Intelligent Systems and Technology (TIST)*, 6(3):29, 2015.

[97] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3):38, 2014.

[98] Yu Zheng, Lizhu Zhang, Zhengxin Ma, Xing Xie, and Wei-Ying Ma. Recommending friends and locations based on individual location history. *Transactions on the Web (TWEB)*, 5(1):5, 2011.

[99] Yu Zheng and Xiaofang Zhou. *Computing with spatial trajectories*. Springer Science & Business Media, 2011.

Appendix- Publications of Fatima Hachem

Journal papers

[1] Maria Luisa Damiani, Fatima Hachem, Hamza Issa, Nathan Ranc, Paul Moorcroft, and Francesca Cagnacci. "Cluster-based trajectory segmentation with local noise". Data Mining and Knowledge Discovery,2018. (status: under review- minor revision).

[2] Maria Luisa Damiani and Fatima Hachem. "Segmentation techniques for the summarization of individual mobility data". Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery (2017) (in press)

Conference papers

[1] Fatima Hachem, Maria Luisa Damiani, and Hamza Issa. "Discovering gatherings based on individual mobility patterns: challenges and direction". In Proc. IEEE ICDM Workshops, Nov 2017.

[2] Maria Luisa Damiani, Hamza Issa, Giuseppe Fotino, Fatima Hachem, Nathan Ranc, and Francesca Cagnacci, "Migro: A plug-in for the analysis of individual mobility behavior based on the stay region model," in Proc. ACM SIGSPATIAL, Nov 2015.