Model-based Group Segmentation

Mols, Jorik

Award date:
2021

Department of Mathematics and Computer Science
Algorithms, Geometry and Applications Group

Model-based Group Segmentation

Master Thesis

Jorik Mols

Supervisors:
dr. Kevin Buchin
Aleksandr Popov

Eindhoven, June 2021

Abstract

Trajectory segmentation is the problem of partitioning a trajectory into different movement phases. There is a large body of work on the problem of segmenting individual trajectories, such as *Behavioral Change Point Analysis* (BCPA) which fits a model to statistical features of trajectories to compute likely points of change in behavior. However, there is very little work on algorithms for segmenting a set of trajectories. The aim of this work is to generalize segmentation to multiple trajectories in a model-based context.

We define a graph-like representation of model-based segmentations and an *information criterion* that can evaluate the quality of these representations, based on their complexity and the models fit to the movement phases and breakpoints of these phases. Our representation is a directed acyclic graph, wherein edges represents a movement phase of a group of subtrajectories, and vertices represent changes in movement or grouping behavior. We generalize movement models including BCPA to this group setting, and formalize our optimization objective with these generalized models.

Furthermore, we present three heuristic algorithms to compute such segmentations. A clustering approach is presented which builds clusters of close subtrajectories with similar movement from a trajectory set and greedily selects a subset of the clusters to form a valid segmentation. We relate the cluster selection problem to *Weighted Set Cover* to provide context on the problem and our greedy solution. We also present an incremental method, where we build a complete segmentation for a trajectory by adding trajectories to the segmentation one by one. We define the subproblem of adding a trajectory to a segmentation and describe how our solution to this problem yields a complete segmentation. The last method we present solves the problem in two steps. This method first discovers which sets of subtrajectories are considered to be grouped, by computing the *trajectory grouping structure*, a graph that represents subgroups in a set of trajectories. These subgroups are then segmented separately using an dynamic programming algorithm that generalizes the existing algorithm for model-based segmentation of individual trajectories.

Finally, we present results obtained from experiments on synthetically generated trajectories and real data sets of moving animal and human entities. We used a simple implementation of the heuristic segmentation methods to run these experiments with different methods and models. We conclude our work with a discussion of the results and future work on model-based segmentation of trajectory groups.

Contents

Chapter 1

Introduction

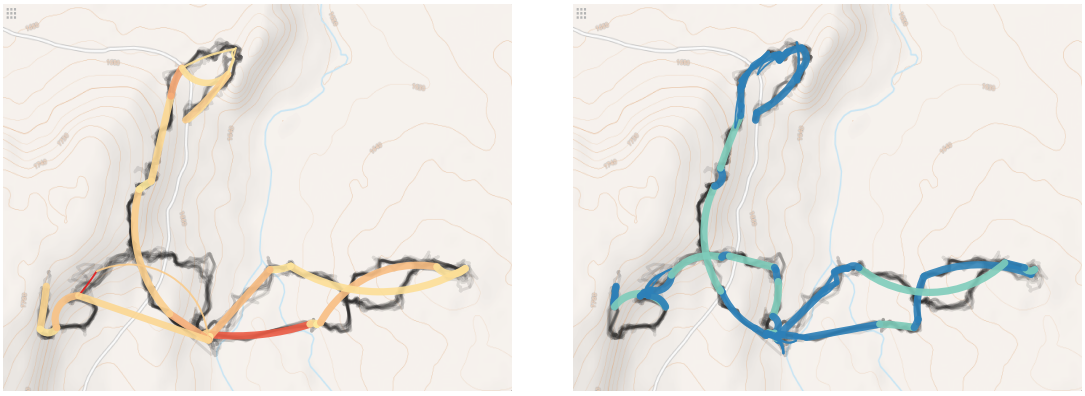


Figure 1.1: Model-based segmentations of multiple baboon trajectories. The result is a directed graph, which follows the course of the trajectories. In this examples it is largely a path, which splits into several parallel edges, when the movement of subgroups require a different model parameter for a good fit. The figures differ in the segmentation algorithm and model that was used (see Section 1.1). On the left: the Brownian bridge movement model, where yellow corresponds to a lower diffusion coefficient than red. On the right: an optimization version of Behavioral Change Point Analysis applied to persistent velocity, where darker blue corresponds to higher autocorrelation.

The spatial tracking of moving objects is becoming easier as technology advances, providing more and more movement data of entities such as cars, animals, people or even hurricanes. Most object tracking happens with the use of GPS, which results in a sequence of timestamped geographic coordinates, representing the movement of an entity over some time period. We refer to this sequence as a *trajectory*. Depending on the tracking hardware that is used, other features such as altitude, velocity or location inaccuracy might also be recorded. There is a recent push in multiple scientific fields where these data are analyzed, to observe not only where these entities are at what times but also to understand why.

The number of devices that can be used for GPS tracking is increasing, with the global use of smartphones being an example of this. This requires novel techniques to analyze this type of data on larger spaces and larger timescales, and in particular with a large number of entities moving at the same time. The analysis of such movement data gives rise to many geometrical problems that we can study algorithmically.

One of these is the *trajectory segmentation* problem. This problem tasks us with dividing the movement of an entity into multiple phases describing different movement states. Finding these

phases and the behavior they describe can lead to insights into movement patterns of tracked entities. Taking ecology as an example, the stop-overs of migrating birds can be identified with segmentation of the migration flight trajectories [2]. This example nicely leads the way for a subsequent question; if we have data of several birds, possibly moving together, can we segment the movement of trajectory groups. In particular, can we represent the movement phases and spatial organization of multiple entities with a segmentation.

We tackle this problem by first investigating segmentation approaches for individual trajectories, then aim to generalize these methods to a group of trajectories. Essentially there are two often studied lines of algorithmic work on the problem of trajectory segmentation. These are criteria-based segmentation and model-based segmentation. In a criteria-based segmentation each *segment*, i.e. a subtrajectory resulting from the segmentation, fulfills a certain (geometric) criterion. These criteria can be a bound on the variance in speed, or staying within a region of a certain radius, for example. Depending on the type of criteria, there are several criteria-based segmentation algorithms, which we will discuss further in the section on related work. Model-based segmentation contrasts the predefined nature of criteria in criteria-based segmentation by instead fitting movement models to the data.

Criteria-based segmentation has been generalized to multiple trajectories using abstract criteria [10] and by clustering based on distance [16]. However, model-based segmentation so far has not been generalized to this multiple trajectory setting. This is what we aim to do in this thesis, which we illustrate in Figure 1.1.

1.1 Related work

From a perspective where multiple trajectories are analyzed, grouping whole trajectories or parts of trajectories is an interesting aspect of the trajectory analysis problem. This grouping is often done based on similarity measures [32], which are able to describe the distance between trajectories and subsequently evaluate their similarity. Common measures include the Fréchet distance [8] and discrete Fréchet distance [18], where the distances of trajectories is measured while allowing a temporal offset. Another common similarity measure is the longest common subsequence (LCSS) [33], where trajectories are interpreted as sequences. Trajectory sequences then are clustered if their sequences are similar in that they can stretch while still having the same order of sequences. Dynamic time warping, a distance-based measure that allows temporal phase shifts, has been used by Haase and Brefeld [23] to analyze similarity between movements in a soccer game.

Grouping subtrajectories based on similarity measures is a large part of most clustering problems [1, 7, 9, 11]. A relevant problem for which clustering methods are often used is map construction [9, 7], where networks are constructed using sets of compact clusters obtained from multiple trajectories. These networks might represent traffic infrastructure or commuting patterns [11] for example. The problem of (k, l)-center clustering asks, given a set of trajectories and a maximum Fréchet distance, whether there are a given number of clusters that represent the trajectories with at most the maximum Fréchet distance. An approximation algorithm for this problem was introduced by Driemel et al. [19] and later improved upon by Buchin et al. [13]. Practically viable algorithms based on these approaches were presented by Buchin et al. [6, 14]. In the context of representing multiple trajectories using clusters, Agarwal et al. [1] describe how clustering subtrajectories can define sets of similar movement based on topological characteristics of the data, and how a selection of clusters can form a decomposition of a trajectory set.

Apart from representing trajectories with sets of clusters, the grouping component that plays a role in the context of multiple trajectories can also be studied. The motivation for this is the ability to tell which trajectories are grouped at certain time intervals. Buchin et al. [12] proposed a method for finding the underlying grouping structure of a set of trajectories, given a closeness measure. This structure can be represented with a geometric graph where each edge resembles a group. This structure is then subsequently used to find maximal groups within the data. While this method only uses the closeness measure as a way of grouping trajectories, *Group Diagrams*, introduced by Buchin et al. [16], allow for any similarity measure to be used, such as

the aforementioned Fréchet distance or equal- or similar time distances. To represent the given trajectories, Group Diagrams use representative segments for each group.

Another movement analysis problem is the segmentation of trajectories, where a trajectory is broken up into multiple segments based on some feature(s) of the trajectory. Informally, these features can represent some facet of the entities behavior, such as a stop-over for migrating birds, or run speeds in tracked running data. Depending on the type of analysis that is to be performed on trajectory data, segmentation methods can answer different research questions. To describe multiple movement patterns in trajectory data, the behavior in sections, or phases, of movement is quantified such that these phases can be expressed in terms of some measure that relates to the behavior being analyzed.

Movement models provide a way of describing such phases using statistical models fit to their movement features. The Brownian motion model and its trajectory application Brownian bridges [24] can be used to describe movement phases in a trajectory based on model parameters. Brownian bridges assume trajectories can be modeled by random walks between two locations with varying variance parameters indicating different behaviors. Statistical models can also be used to report change-points between segments, focusing on where the most likely points of change are for the varying movement characteristics. The Behavioral Change Point Analysis (BCPA) method, which models the underlying movement of a trajectory as a stochastic process in order to find points of behavioral change, was first introduced by Gurarie et al. [21]. This model-based method is able to model any time-series variable that can be obtained from given trajectory data. It uses likelihoods and a sweep-line algorithm to find the points of change in the underlying process, which can then be presented either as a continuous estimation function or a discrete set of change-points. Movement models have been especially helpful in ecologic fields where trajectory data of animals are analyzed, which provides a motivation for the development of new segmentation techniques. These studies often go further than just the segmentations themselves, diving deeper into ecological animal-specific features. Examples of this include the usage of BCPA to discover integration ratios for black cockatoos [27], or how sequential collecting of movement states between BCPA change-points give insights into behavioral classes [31].

An alternative to model-based segmentation is segmentation using criteria on trajectories. Segmentation then occurs such that each subdivided part of a trajectory satisfies some global criterion. These criteria place constraints on topological features of the data, such as trajectory locations fitting inside a disk with a fixed radius or the range of heading angles fits some maximum range. Aronov et al. [5] paint a broader picture of the criteria-based segmentation problem, and provide a method for computing an optimal segmentation for two classes of criteria. Agagnostopoulos et al. [4] have researched how to optimize the computation of distance-based criteria for efficient segmentation methods. A general framework for criteria-based segmentation allowing for different classes of stable criteria is presented by Alewijnse et al. [15]. In the above approaches the behavior in each segment can only be defined by the set of criteria it satisfies. Additionally, they are suitable only for single trajectories. Buchin et al. [10] have generalized this concept of criteria-based segmentation to a collective movement context. The Compact Flow Diagrams they introduce nicely represent segmentations of multiple trajectories with a directed acyclic graph structure wherein trajectories are represented by paths through the graph.

Trajectory data often contains additional attributes alongside a set of timestamps with corresponding location, such as velocity or heading angle. The topic of semantic trajectories and their segmentation aims to augment trajectories with contextual data that is obtained from environmental knowledge of the analyzed trajectory [26]. Behavioral states or criteria can then be defined using these augmented data. Ogawa et al. [29] have proposed an algorithm to segment semantic trajectories using machine learning methodologies, where models are trained on video data of trajectories. This shows that machine learning can play a role in model-based segmentation methods.

The usefulness of any segmentation method might differ depending on the trajectory data to be analyzed, or the features that are to be examined via segmentation of the data. To aid new and existing researchers in getting an overview of segmentation methods for trajectories, several works summarize a set of segmentation methods and list advantages and disadvantages

of each in relation to different properties of trajectory segmentation. Gurarie et al. [22] have outlined the segmentation problem as a whole, listed different segmentation methods and compared results of several methods. A similar overview and comparison is given by Edelhoff et al. [20], where the segmentation problem is generally deconstructed and some methods are evaluated with simulated data to analyze their usefulness with relation to common research questions in trajectory segmentation.

1.2 Contributions

We study the problem of *group segmentation* in a model-based context by defining a segmentation representation that encapsulates the models fit to the data as well as subgroups of the trajectories that are segmented. Examples of such segmentations are depicted in Figure 1.1. This representation is inspired by Compact Flow Diagrams [10] and Group Diagrams [16], which represents segmentations in a graph-like manner. For evaluating segmentations we define an Information Criterion (IC) which serves as a metric for group-based segmentations. Similarly to how Alewijnse et al. [3] have adopted and modified the Bayesian Information Criterion (BIC) to express the IC score of a single-trajectory segmentation, we derive an IC that is able to evaluate segmentations in a multiple trajectory setting. The IC indicates how well the models fit to the representation, while counter-acting overfitting by adding a complexity penalty term. We then employ this IC to define the model-based segmentation problem in a group setting as an optimization problem.

In Chapter 3 we present three heuristic methods for computing a segmentation according to our defined representation, using elements from related work on clustering [12] and single-trajectory segmentation using movement models [3]. These approaches include a clustering method, where we construct clusters of subtrajectories and give them a score based on their estimated information gain. We see that the problem of selecting a subset of clusters to obtain an optimal segmentation in terms of the IC is related to the Weighted Set Cover problem [1]. We then greedily select a set of non-overlapping clusters to obtain a valid segmentation. We present an incremental approach where a complete segmentation of the given trajectory set is built incrementally, adding trajectories one at a time, and merging their segmentations into the complete result at every iteration. We aim to solve the problem of how to add a trajectory to a segmentation representing a set of trajectories such that afterwards the segmentation also represents the given trajectory. The third approach involves two steps that separate the problem into finding a grouping structure and segmenting these groups. The trajectory grouping structure algorithm [12] is implemented to perform the first step, while we use the single-trajectory model-based segmentation algorithm of Alewijnse et al. [3] together with generalized movement models to segment the resulting groups.

Chapter 2

Group segmentation

To segment trajectory groups, we first have to address how to represent such a segmentation and how to evaluate the quality of a segmentation. This chapter provides an overview of the representation and evaluation of segmentations and shows how we tackle these questions.

By explaining how we define and represent our segmentations and evaluate these using different models we lay out the prerequisite material for Chapter 3, where we present several heuristics to the segmentation problem. We first describe the notation that is used for trajectories and segmentations throughout this thesis in Section 2.1. In Section 2.2 we introduce our representation for model-based group segmentation and place it with relation to other representations. In Section 2.3 we show how we can evaluate these segmentations and present model-based segmentation of multiple trajectories as an optimization problem. The models used in the segmentation representation will be discussed in Section 2.4, where we generalize location and movement models to groups and give a short motivation and background on these models.

2.1 Notation

We first describe the terminology and notation of trajectories and then define segmentations and their notation. We define a trajectory τ as a sequence $\langle (z_1, t_1), (z_2, t_2), \ldots, (z_n, t_n) \rangle$, where z_i are locations and t_i are timestamps. We assume the locations z_i are points in the Euclidean plane, however, most of the methods we present would generalize to higher dimensions. We write $\tau(t_i) = z_i$ to denote the location of trajectory τ at time t_i.

Given such a trajectory, we implicitly interpolate movement between the timestamped locations to obtain a continuous trajectory. If we assume entities move between points at a constant velocity, we can linearly interpolate between these points based on the two accompanying timestamps. Under this assumption a trajectory can be interpreted as a polygonal curve. Note that we do not assume constant velocity when fitting movement models to subtrajectories. However, when visualizing trajectories, we will draw them as polygonal curves.

We can indicate a subtrajectory by a time interval and a trajectory as follows: $\tau[t_i, t_j]$ denotes the subtrajectory described by the points $\langle z_i, z_{i+1}, \ldots, z_j \rangle$ and timestamps $\langle t_i, t_{i+1}, \ldots, t_j \rangle$ that belong to trajectory τ. Note that in the work done in this thesis we will only use time intervals that start and end on timestamps for which there is a corresponding location in the 'indexed' trajectories.

The input to our methods is a set of trajectories $\mathcal{T} = \{\tau_1, \tau_2, \tau_3\}$. For such input trajectory sets, we make the assumption that all trajectories in the set share the same set of timestamps. A subtrajectory of a single trajectory in a trajectory set $\mathcal{T}$ can be denoted with $\tau_k[t_i, t_j]$, such that $\tau_k \in \mathcal{T}$. It will also be useful to be able to mention only a subset of the trajectories, such as $T \subseteq \mathcal{T}$. We can then denote a set of subtrajectories on a time interval in a similar fashion, namely as $T[t_i, t_j]$. Formally, we then have $T[t_i, t_j] = \{\tau_k[t_i, t_j] \mid \tau_k \in T\}$. Additionally, for some $T \subseteq \mathcal{T}$ let $T(t_i) = \{\tau_k(t_i) \mid \tau_k \in T\}$, e.g. the set of locations of trajectories in T at time t_i.

A segmentation breaks up a trajectory, or in our case a set of trajectories, into smaller sets of subtrajectories. We call these subtrajectory sets *segments*. A segment s is defined on a time interval for a subset of trajectories in the segmented trajectory set $\mathcal{T}$. We let $t_\alpha(s)$ and $t_\beta(s)$ denote the start and end times of s, respectively. Let $\Gamma(s)$ denote the trajectories in $\mathcal{T}$ whose subtrajectories are contained in s. Note that we always have $\Gamma(s) \subseteq \mathcal{T}$, and $t_1 \leq t_\alpha(s), t_\beta(s) \leq t_n$ as a segmentation is only defined on the points and timestamps of the trajectories in $\mathcal{T}$.

2.2 Representing segmentations

There are different ways to represent a segmentation of multiple trajectories. To get a better understanding of what resulting representations we are interested in, we first discuss segmentation representations from related work, and define a model-based representation which we base our segmentation techniques on.

2.2.1 Related representations

There are several related works that also consider segmentation in a the context of multiple trajectories. These works introduce different ways to represent such segmentations.

Compact flow diagrams An abstract representation which uses directional acyclic graphs is a *Compact Flow Diagram* (CFD) [10], where vertices of a graph represent behavioral states and edges indicate transitions between those states. Specifically, vertices correspond to some criterion. For example, one criterion could be that the velocity is bounded by 10 m/s. Additionally to the vertices representing behavioral states, a CFD also has a start vertex s and end vertex t, and paths from s to t represent the different behavioral states that some trajectory in the set goes through. Every trajectory in the set is represented by one such path, such that the trajectory can be segmented into subtrajectories in such a way that the segments fulfill the criterion of the corresponding states. Note that the time of change is not indicated since neither the vertices or edges have a temporal component. With additional visualization, a CFD can be used to highlight behavioral patterns from a trajectory set. An example of a CFD is shown in Figure 2.1. The behavioral states in this example would thus relate to different criteria, such as speeds within a range, a heading angular range, or staying inside an area with fixed radius.

Group diagrams Group Diagrams (GD) [16] focus on the geometry of the movement by representing multiple trajectories using a geometric graph. In such a graph, the vertices represent timestamped geographical locations and edges represent movement between locations. Consequently, a path in this graph can be interpreted as a trajectory in itself. Similar to Compact

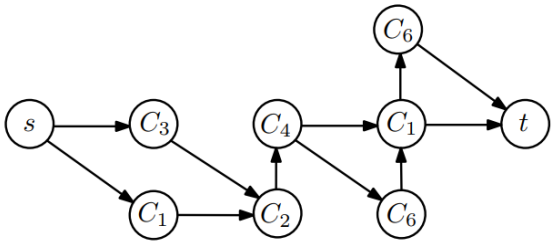


Figure 2.1: An example of a Compact Flow Diagram, where the vertices represent different behavioral states $C_1, \ldots, C_6$ and directional edges between these states indicate state transitions. An example path would be $\langle s \to C_3 \to C_2 \to C_4 \to C_1 \to t \rangle$, which would represent at least one trajectory which follows the state sequence C_3, C_2, C_4, C_1.

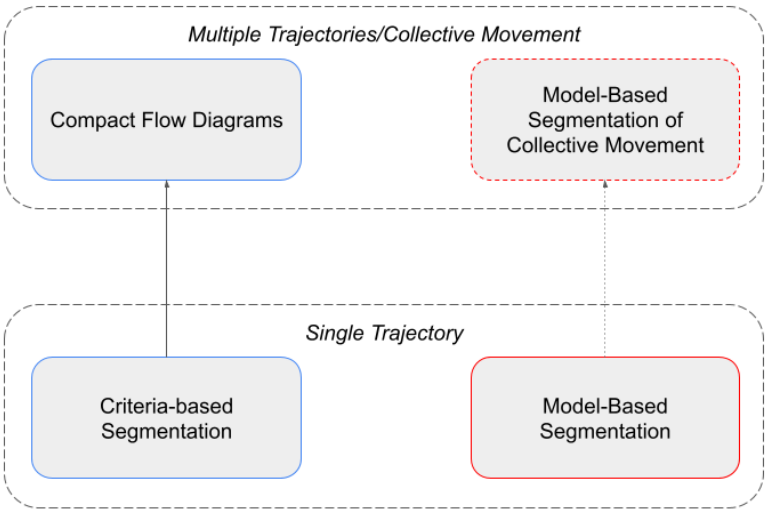


Figure 2.2: A diagram indicating how our work relates to Compact Flow Diagrams and single trajectory based segmentation methods. The arrows indicate generalization from single to multiple trajectory contexts. Where Compact Flow Diagrams generalized single trajectory criteria-based segmentation, our methods generalize the single trajectory model-based variant of segmentation.

Flow Diagrams, Group Diagrams also represent trajectories with paths in a graph, and since paths are in itself similar to trajectories Group Diagrams provide an intuitive representation.

Single trajectory criteria-based segmentation The above representations generalize criteria-based segmentation of individual trajectories to the setting of multiple trajectories. There is extensive work on criteria-based segmentation of single trajectories [5, 15, 2]. These methods are generalized to a group setting by Compact Flow Diagrams and Group Diagrams. In our work, we do not take criteria-based segmentation as a starting point for generalization. Instead, we tackle the generalization of model-based segmentation to a group setting by starting from single trajectory model-based segmentation methods introduced by Alewijnse et al. [3].

Trajectory grouping structure Another relevant related data structure that expresses groups in a set of trajectory is the *trajectory grouping structure* [12]. The aim of this structure is to find maximal groups based on certain constraints that tell whether a set of subtrajectories form a group. As an intermediate step in this algorithm, a graph is created where each edge represents one or more subtrajectories that are close, based on a closeness threshold ε. Since we will be using this part of the algorithm in one of the segmentation approaches we introduce in Chapter 3, we describe it in more detail in the following paragraphs. Although the complete work on the trajectory grouping structure results in a definition and representation that differs from the intermediate graph result which we are interested in, we will be referring to this intermediate graph as the "grouping structure" to still make it clear where it originates from.

The goal is to find a set $\mathcal{R}$ of subtrajectories we call groups such that the entire trajectory set $\mathcal{T}$ is covered by non-overlapping groups in $\mathcal{R}$. A group $r \in \mathcal{R}$ is defined as a set of trajectories $T_r \subseteq \mathcal{T}$ on a time interval $[t_\alpha(r), t_\beta(r)]$, where $t_\alpha(r)$ and $t_\beta(r)$ are the start and end time of group r. A more concise description for r then is the set of subtrajectories $T_r[t_\alpha(r), t_\beta(r)]$.

In order for $\mathcal{R}$ to be a valid grouping structure for $\mathcal{T}$, we define the following constraints. For two groups $r_1, r_2 \in \mathcal{R}$ that have overlapping time intervals, we must have $T_{r_1} \cap T_{r_2} = \emptyset$. Conversely, if we have $T_{r_1} \cap T_{r_2} \neq \emptyset$ then the time intervals must not overlap. Finally, each trajectory $\tau \in \mathcal{T}$ must be represented by a sequence of non-overlapping groups in $\mathcal{R}$ such that the entirety of τ is

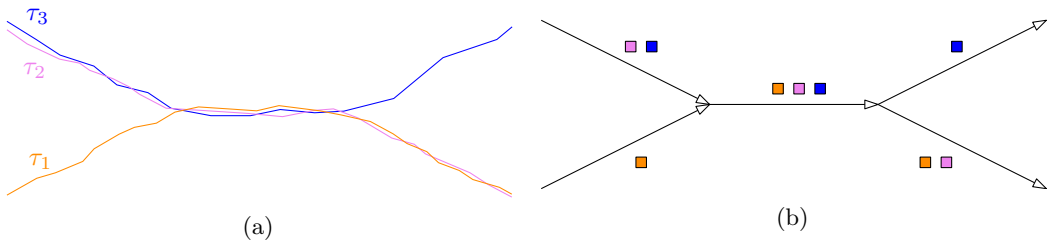


Figure 2.3: A visualized example of the intermediate result of the trajectory grouping structure method [12], where a graph representing close subtrajectories on different time intervals is constructed. (a) A set of color-coded trajectories $\{\tau_1, \tau_2, \tau_3\}$. (b) The graph generated from the trajectories in (a). The directed edges represent the groups. The colored boxes next to the edges represent the trajectories "contained" in those groups.

covered by the groups in the sequence. If a grouping structure satisfies these conditions, it is a valid grouping structure for $\mathcal{T}$.

The existing trajectory grouping structure algorithm uses a closeness parameter ε that determines how close trajectories must be in order for them to be considered grouped. This method groups two trajectories $\tau', \tau'' \in \mathcal{T}$ with a single link restriction. If the distance $d(\tau'[t], \tau''[t])$ between τ' and τ'' at time t is at most ε then they are considered directly ε-connected at time t. If there is a sequence of trajectories $\tau' = \tau_0, ..., \tau_k = \tau''$ where for all i, τ_i and τ_{i+1} are directly ε-connected at time t, then τ' and τ'' are considered to be ε-connected at time t.

Given the parameter ε, for increasing time t_i the algorithm will record 'connect' and 'disconnect' events for pairs of trajectories. Two trajectories $\tau_1, \tau_2 \in \mathcal{T}$ will connect at time t_i if $d(\tau_1[t_{i-1}], \tau_2[t_{i-1}]) > \varepsilon$ and $d(\tau_1[t_i], \tau_2[t_i]) \leq \varepsilon$. They disconnect if the reverse holds, e.g. $d(\tau_1[t_{i-1}], \tau_2[t_{i-1}]) \leq \varepsilon$ and $d(\tau_1[t_i], \tau_2[t_i]) > \varepsilon$. A visualization of an example trajectory grouping structure graph is shown in Figure 2.3.

We process the list of all connect and disconnect events for all pairs of trajectories in chronological order. For increasing time t, we keep track of all groups at time t with a graph $G_{\mathcal{R}} = (V, E)$. Each vertex $v_i \in V$ represents trajectory $\tau_i \in \mathcal{T}$, and there is an edge $e_{i,j} = (v_i, v_j) \in E$ if and only if τ_i and τ_j are directly ε-connected. We then keep track of all the connected components $C \in \mathcal{C}(G)$ in the graph, and give them a timestamp $t(C)$ indicating the time at which they last changed. The connected components at that point represent the different groups, since a path between two vertices v_i, v_j means that τ_i and τ_j are ε-connected. We let T_C denote the set of trajectories represented by component C. A connect event with trajectories τ_i, τ_j adds the edge $e_{i,j}$, while a disconnect event removes it. If a connect event at time t' joins two components C_1, C_2, we can add two groups to our resulting group structure $\mathcal{R}$; $T_{C_1}[t(C_1), t']$ and $T_{C_2}[t(C_2), t']$. We then let $C' = C_1 \cup C_2$ denote the newly formed component, and set its timestamp $t(C') := t'$. In the case of a disconnect event for trajectories τ_i, τ_j, if the component C for which $v_i, v_j \in C$ is broken up into two components C_1, C_2 then we add the group $T_C[t(C), t']$ to $\mathcal{R}$. We update the list of components with C_1, C_2 replacing C, and set $t(C_1), t(C_2) := t'$. When all events have been processed, we have obtained a grouping structure $\mathcal{R}$ for trajectory set $\mathcal{T}$.

2.2.2 Model representation

Segmentation techniques are often used to visually explore data sets of moving objects. We focus on *group segmentation*, where a set of trajectories is segmented into different phases of movement and groups. Our starting point for a representation includes Compact Flow Diagrams and Group Diagrams to obtain a representation that accommodates the usage of models in our segmentation. As seen in these related representations, using graphs generally allows for an underlying grouping structure to be intuitively described. Our work differs from the work on Compact Flow Diagrams and Group Diagrams in that we use models to describe the behavioral states and change points,

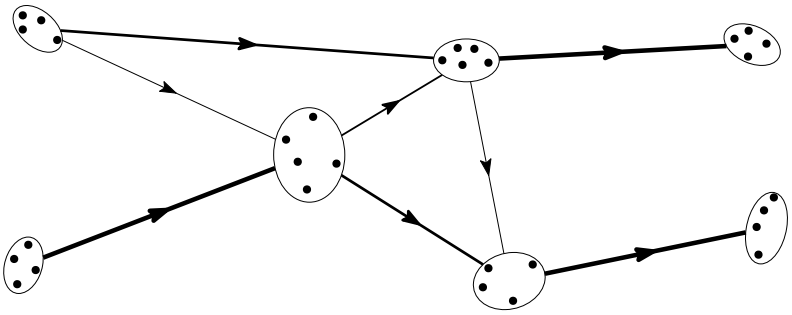


Figure 2.4: A visually represented model-based segmentation graph $G_{\mathcal{S}} = (V, E)$. The ellipses containing points represent the vertices $v \in V$, where the shape of the ellipsis of each vertex v represents the location model fit to the point set $P(v)$ with parameter $\overline{\theta}(v)$. While the behavior in the movement models fit to the edges $e \in E$ is not visualized, the thickness of the line belonging to each edge e represents the amount of subtrajectories represented by e. Larger values of $|M(e)|$ are represented with thicker lines.

and as such we have model parameters to describe how our models can be fit to the data.

Given a set of trajectories $\mathcal{T}$, we let a directed acyclic graph $\mathcal{S} = (V, E)$ represent a segmentation of $\mathcal{T}$. As before, we assume every trajectory $\tau \in \mathcal{T}$ contains n observations. We then switch the functions that vertices and edges have in a Compact Flow Diagram, letting edges in our graph describe movement with a model parameter component and letting vertices describe the modeled locations in between sections of movement. We thus make the distinction between movement models which model the movement characteristics of one or more subtrajectories, and location models which model the location distribution of one or more points. Using this setup we are also able to assign timestamps to vertices with which we can deduce the time intervals on which segments are defined. Formally, for a vertex $v \in V$ we use $\overline{\theta}(v)$ to denote the location model parameters assigned to v and for an edge $e \in E$ we use $\overline{\sigma}(e)$ to denote the movement model parameters assigned to e. We use vector notations for $\overline{\theta}$ and $\overline{\sigma}$ to emphasize that there might be more than one model parameter depending on the model used. We also let $P(v)$ denote the points represented by v, and let $M(e)$ denote the subtrajectories that are represented by e. Note that in this case $P(v) = T[t_i]$ for some $T \subseteq \mathcal{T}$ and $0 \leq i \leq n$, and $M(e) = T[t_i, t_j]$ for some $T \subseteq \mathcal{T}$ and $0 \leq i < j \leq n$. We can use the location model parameters to place the vertices onto a 2D plane[1] to clearly indicate where the segments in the segmentation start and end geographically. We do not assume our graphs to be planar, as edges may intersect. Figure 2.4 shows an example where the different parts of the representation are visualized. Note the explicit depiction of the location models and the thickness of segments, which represents the amount of subtrajectories contained in each edge.

Our model-based representation abstracts from the models used in its vertices and segments. As long as a location or movement model is able to express the location or movement of multiple subtrajectories in terms of a set of parameters, it can be used in this representation. For placing vertices in geographical locations found by the segmentation, we do not necessarily require explicit x, y coordinates from the location model parameters, but rather an indication of a central location obtainable from the parameters.

2.3 Evaluating segmentations

To evaluate segmentations in the representation we have defined in Section 2.2.2, we use an information criterion (IC). First we describe the IC and its usage, then we formulate the segmentation

[1]For example, a circular normal distribution is defined on central x, y coordinates with a variance parameter. These coordinates can then be used to place the vertex.

problem as an optimization problem with the IC.

2.3.1 IC definition and usage

To evaluate a segmentation as defined by our model-based representation, we want to have some measure with which we can estimate the quality of a segmentation. Two segmentation features that we can relate to its quality are its complexity and how well the models fit to its locations and movement. A well-known measure for finding an optimal model within a given set of models is the *Bayesian information criterion* (BIC), which balances a model's complexity with its log-likelihoods to obtain a value that when minimized, corresponds to the optimal model. This balance is a solution to model overfitting, which is a problem that applies to model-based segmentation. For a model with likelihoods $\hat{L}$, number of observations n and complexity k, the BIC is defined as:

$$BIC = k\ln(n) - 2\ln(\hat{L}) \tag{2.1}$$

A variation of this IC is *Akaike's information criterion* (AIC), which differs from the BIC in interpreting the complexity of a model, and is used as a more statistical way of determining which model to select.

$$AIC = 2k - 2\ln(\hat{L}) \tag{2.2}$$

We first describe how we define our model log-likelihoods and model complexity with relation to our segmentation representation defined in Section 2.2.2. The likelihood of a model is often formulated in terms of some model parameters. For an edge $e \in E$ in a segmentation, we use some movement model to determine the log-likelihood of the subtrajectories in e, denoted as $M(e)$, given model parameters $\overline{\sigma}(e)$. We then write the log-likelihood of edge e as $LL(M(e) \mid \overline{\sigma}(e))$, and similarly write $LL(P(v) \mid \overline{\theta}(v))$ for the log-likelihood of vertex v with points $P(v)$ and model parameter $\overline{\theta}(v)$. An important observation to be made is that the movement models only model the subtrajectories between locations, and that the location models are modeling points that are not modeled by the movement models. From here we assume independence between the movement and location models. To summarize the log-likelihoods of all the models in the segmentation we can then simply sum the log-likelihoods of the movement and location models. The complexity of a segmentation can be stated in terms of its number of vertices $|V|$ and/or its number of edges $|E|$.

Now consider some segmentation $\mathcal{S}$ which segments trajectory set $\mathcal{T}$ where trajectories have n points. Let $C(\mathcal{S})$ be some function of $|V|$ and $|E|$. We then apply the IC on our segmentation representation in a similar fashion to the form of the IC defined in [3]. This gives us the following formula:

$$\mathrm{IC}(\mathcal{S}) = \ln n \cdot C(\mathcal{S}) - \sum_{v \in V} 2 \cdot LL(P(v) \mid \overline{\theta}(v)) - \sum_{e \in E} 2 \cdot LL(M(e) \mid \overline{\sigma}(e)) \tag{2.3}$$

In the BIC (and AIC in a similar fashion) the complexity term consists of the model complexity multiplied with some penalty weight. This complexity penalty weight is different for both IC's, and since we will be generalizing our movement models to groups the complexity weights may no longer match with the log-likelihoods that are obtained from movement and location models. We thus leave the complexity penalty weight as a variable parameter p. Additionally, there is a relation between $|V|$ and the location model log-likelihoods, as well as between $|E|$ and the movement model log-likelihoods. Let $C(V)$ and $C(E)$ denote the complexity function $C(\mathcal{S})$ separated into two parts for V and E. We can now express these relations in the IC by splitting up the IC into two parts IC_{loc} and IC_{mov} for the locations (vertices) and movements (edges) respectively:

$$\mathrm{IC}_{loc}(\mathcal{S}) = p \cdot C(V) - 2 \cdot \sum_{v \in V} LL(P(v) \mid \overline{\theta}(v)) \tag{2.4}$$

$$\mathrm{IC}_{mov}(\mathcal{S}) = p \cdot C(E) - 2 \cdot \sum_{e \in E} LL(M(e) \mid \overline{\sigma}(e)) \tag{2.5}$$

Note that we use the same complexity penalty weight p for both the location and movement parts. It is possible to have two different factors, which gives more separate control over the location and movement complexity in the segmentation, however we simply use equal values since exploring data with different penalty factors lies outside this work's scope. We can sum the two parts to get a complete IC of a segmentation which we use to evaluate our segmentations:

$$\mathrm{IC}(\mathcal{S}) = \mathrm{IC}_{loc}(\mathcal{S}) + \mathrm{IC}_{mov}(\mathcal{S}) \tag{2.6}$$

2.3.2 Optimizing the IC

We now use the IC to define the segmentation problem as an optimization problem similar to [3]. If, for some trajectory set $\mathcal{T}$, we find the segmentation which minimizes the IC, we have found an optimal segmentation according to the IC. We define this problem formally as MultiTrajectoryModelSegmentation, or MTMS:

MTMS($\mathcal{T}, n$): Given a trajectory set $\mathcal{T}$, where each $\tau \in \mathcal{T}$ consists of n time-stamped locations, and a complexity weight p, find an optimal segmentation $\mathcal{S}$ such that for all segmentations $\mathcal{S}'$ of $\mathcal{T}$:

$$\mathrm{IC}(\mathcal{S}) \leq \mathrm{IC}(\mathcal{S}')$$

The definition of the optimization problem does not help us directly in finding an optimal solution, but we can interpret solutions or segmentations with lower IC scores as better solutions. The formal definition of this problem provides us with an optimization objective we aim to fulfill with model-based segmentation techniques.

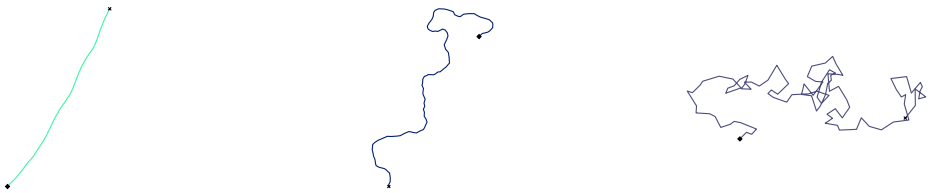
2.4 Generalizing models to groups

The information criterion we have defined in Equation 2.6 for our segmentation representation uses the log-likelihoods of multiple points or subtrajectories. For a vertex v in a segmentation $\mathcal{S}$ with representation graph $G_{\mathcal{S}}$ we now need to compute $\mathrm{LL}(P(v) \mid \overline{\theta}(v))$, and for an edge e we need to compute $\mathrm{LL}(M(e) \mid \overline{\sigma}(e))$. For our purposes, we do not believe modeling a set of points with a location model requires complex methods, since we are mostly focused on some measure of closeness for the vertices in a representation graph. A simple location model is described in Section 2.4.3. However, expressing the log-likelihood of multiple subtrajectories requires movement models that are generalized to groups.

Multiple movement models have been presented with which an entity's movement can be described. We consider the Brownian Bridge Movement Model (BBMM) and the Behavioral Change Point Analysis (BCPA) to test our methods with. In the field of trajectory analysis and segmentation, these models have previously been used to describe the movement behavior of different animals by finding change-points in the fitted model parameters [3][21]. In Sections 2.4.1 and 2.4.2 we will discuss how to apply these models in a group setting. For both BBMM and BCPA we will formulate the log-likelihood of a subtrajectory such that in describing the general model-based segmentation technique we can use these likelihoods regardless of which model is being used. The step-by-step process of how we obtain the generalized log-likelihoods will be from the perspective of a trajectory set $\mathcal{T}$ rather than from the perspective of edges in segmentation representation graphs, since the locations of trajectories at different timestamps need to be explicitly mentioned in the models.

2.4.1 Brownian Bridge Movement Model

Brownian motion is an existing framework for describing animal movement. This type of movement closely relates to random walks, where we can express a path in terms of a diffusion coefficient. A



(a) $\sigma_m^2 = 0.120$ (b) $\sigma_m^2 = 3.808$ (c) $\sigma_m^2 = 12.60$

Figure 2.5: Three randomly generated trajectories with varying behavior to showcase the effect of different diffusion coefficient values in the Brownian bridge movement model. The diffusion coefficient values under each trajectory maximize the BBMM log-likelihoods $LL(\tau \mid \sigma_m^2)$ for each trajectory τ. A higher value of σ_m^2 results in more erratic behavior.

path or trajectory between two points can also be described with Brownian motion using Brownian bridges. These Brownian bridges are an established component in trajectory analysis and will be useful for segmentation purposes as we will see further on in this section. First we will expand upon the ways we can model our (sub)trajectories using these Brownian bridges, and how we reach a likelihood estimation for a segment. For most formulas below, the reasoning from [24] is followed.

Let us first consider the simple example of a single bridge between two points. Consider two points a and b observed at times t_a and t_b respectively. We can then use a Brownian bridge process to model the location distribution of the entity at any time $t_a \leq t_z \leq t_b$. The location distribution of an entity moving from a to b with Brownian motion is additionally defined by a diffusion coefficient σ_m^2 which describes the general deviation of the entity. Figure 2.5 gives a visual example of how the diffusion coefficient can describe different trajectories. Let $t_{ab} := t_b - t_a$ denote the time it takes for the entity to move from a to b, and let $t_{az} := t_z - t_a$ denote the time it takes for the entity to move from begin point a to some location at time t_z. The observed location's mean $\mu(t)$ and circular variance $\sigma^2(t)$ of the normal distribution at time t_z are given by:

$$\mu(t_z) = a + (b - a) \cdot \frac{t}{t_{ab}} \tag{2.7}$$

$$\sigma^2(t_z) = \frac{t(t_{ab} - t)}{t_{ab}} \sigma_m^2 \tag{2.8}$$

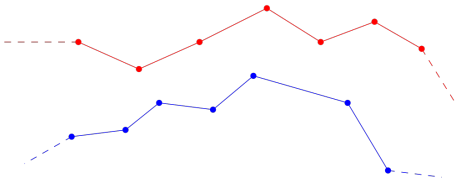
To account for measurement errors in the observed locations, points a and b can be considered as random variables. We use the same approach, and let these correspond to circular normal distributions, $\mathcal{N}(a, \delta_a^2 \mathbf{I})$ and $\mathcal{N}(b, \delta_b^2 \mathbf{I})$ for the starting and end position respectively. To accommodate these uncertainties of the bridge edge points, we have to change the formula of variance $\sigma^2(t_z)$ slightly:

$$\sigma^2(t_z) = t_{ab}\alpha(1 - \alpha)\sigma_m^2 + (1 - \alpha)^2 \delta_a^2 + \alpha^2 \delta_b^2 \tag{2.9}$$

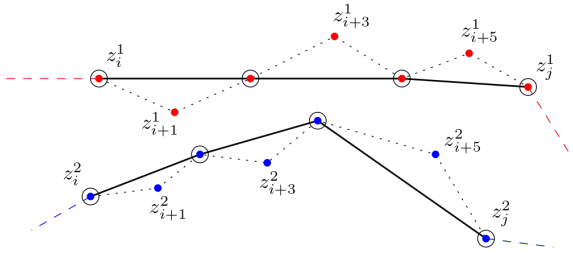
$$\alpha = t_{az}/t_{ab} \tag{2.10}$$

Using the above, we can also obtain a likelihood for an independently observed location z between two points a and b, given a diffusion coefficient. This allows us to describe the Brownian movement behavior of a so-called *Brownian bridge*. Let z be a point observed at time t_z between two points a and b. The likelihood of the Brownian bridge formed between a and b with observed location z can be written as:

$$L(a, b, z, \sigma_m^2) = \frac{1}{2\pi\sigma^2(t_z)} \exp\{\frac{-[z - \mu(t_z)][z - \mu(t_z)]^\top}{2\sigma^2(t_z)}\} \tag{2.11}$$



(a) Example subtrajectories of two different entities, displayed in red and blue.



(b) The underlying Brownian bridge movement of the trajectories in (a), where only uneven observations are considered. The bridges are shown by the bold black lines, the intermediate independent observations are labeled with z_k^1 and z_k^2 for $k \in odd(i,j) = \{i+1, i+3, i+5\}$.

Figure 2.6: An example of the structure of Brownian bridges in a segment with multiple bridges.

Note that the diffusion coefficient σ_m^2 is used implicitly in $\sigma^2(t_z)$. Writing σ_m^2 as one of the parameters helps us indicate what diffusion coefficient value we assign to Brownian bridges later on. For a more detailed description of how the likelihood function for a Brownian bridge is obtained, see [24].

To describe the Brownian movement of a subtrajectory with multiple bridges we only consider the odd observations for Brownian bridge links. For some subtrajectory $\tau[t_i, t_j]$, we then consider only those points z_k with $i < k < j$ where $k - i$ is odd. This gives us $\lfloor \frac{(j-i)-1}{2} \rfloor$ bridges, for which we can take the product of their likelihoods (since their likelihoods are independent) to get a total likelihood for $\tau[t_i, t_j]$. To reduce notation complexity, let $odd(i,j)$ denote the integers k for which $i < k < j$ and $k - i$ is odd. Figure 2.6a and Figure 2.6b provide a visual example of how to divide a subtrajectory into several bridges using observations z_k with $k \in odd(i,j)$. We obtain the following formula for the likelihood of a subtrajectory modeled with Brownian bridges:

$$L(\tau[t_i, t_j], \sigma_m^2) = \prod_{k \in odd(i,j)} \frac{1}{2\pi\sigma_k^2(t_k)} \exp\{\frac{-[z_k - \mu_i(t_k)][z_k - \mu_k(t_k)]^\top}{2\sigma_k^2(t_k)}\} \tag{2.12}$$

where:

$$\begin{aligned} \mu_k(t_k) &= z_{k-1} + \alpha_i(z_{k+1} - z_{k-1}) \\ \sigma_k^2(t) &= (t_{k+1} - t_{k-1})\alpha_k(1-\alpha_k)\sigma_m^2 + (1-\alpha)^2\delta_{k-1}^2 + \alpha_k\delta_{k+1}^2 \\ \alpha_k &= \frac{(t_k - t_{k-1})}{t_{k+1} - t_{k-1}} \end{aligned}$$

We now have defined the likelihood for a single subtrajectory and are left with defining the likelihood for a group of trajectories. Note that we consider a 'group' to be a set of subtrajectories that are considered as grouped (this classification of groups will be mentioned in segmentation methods later on and is not important for this chapter on movement models) and as such we wish to define the Brownian bridge likelihood of this group given a diffusion coefficient. For simplicity, we consider some set of trajectories $\mathcal{T} = \{\tau_1, \ldots, \tau_m\}$ which is assumed to be a group from start to finish. For a set of subtrajectories, given that they are on the same time interval, the following notation is very similar. The total likelihood of the model parameter $L(\mathcal{T}, \sigma_m^2)$ will apply the diffusion coefficient σ_m^2 to every trajectory in $\mathcal{T}$. We use Equation 2.12 and its underlying approach for the likelihood computation.

If each of the trajectories $\tau \in \mathcal{T}$ has their own likelihood $L(\tau, \sigma_m^2)$, then we can add these likelihoods together when determining the likelihood of all the trajectories in the group. We

assume independence between the observations of the different trajectories, and we use the fact that we can take the sum of log-likelihoods instead of a product of likelihoods. This gives us the Brownian Bridge log-likelihood for a trajectory group $\mathcal{T}$, which we write as $LL_{\text{BBMM}}(\mathcal{T}, \sigma_m^2)$ to indicate that we are using the logarithms of the likelihood terms.

$$LL_{\text{BBMM}}(\mathcal{T}, \sigma_m^2) = \sum_{\tau \in \mathcal{T}} \log(L(\tau, \sigma_m^2)) \tag{2.13}$$

Finally, let us relate this log-likelihood to the IC. If e is some edge in a segmentation $\mathcal{S}$ with representation graph $G_{\mathcal{S}}$, then if Brownian bridges were used in segmentation $\mathcal{S}$ we simply have $LL(M(e) \mid \overline{\sigma}(e)) = LL_{\text{BBMM}}(M(e) \mid \overline{\sigma}(e))$. Note that due to the segmentation using Brownian bridges we can implicitly assume $\overline{\sigma}(e)$ represents only a diffusion coefficient σ_m^2 when using LL_{BBMM}.

2.4.2 Behavioral Change Point Analysis

The Behavioral Change Point Analysis (BCPA) as presented in [21] is a framework for finding changes in movement behavior. It essentially tries to find points where the behavior of a trajectory changes. The BCPA abstracts from the feature of the trajectory that it is modeling with a stochastic process. For example, variables such as the trajectory's velocity, heading angle or time delay between points can all be modeled by such a process to find points where these variable likely change. Note that we say "likely", since the BCPA algorithm by itself can either select the most likely change-point or give an output that averages parameter values for every observed location of the trajectory data. The algorithm performs a windowed sweep over a trajectory, meaning it considers a small time interval for increasing times while performing likelihood estimation to evaluate whether there is a change-point in the current window. In this regard BCPA already differs much from the Brownian bridges in the sense that by itself it is already capable of showing when the movement behavior in a trajectory changes, while the Brownian bridges simply describe movement behavior. The movement behavior itself is then described by BCPA in terms of an autocorrelation coefficient which serves as a parameter to the model. In this section we will explain how we use the BCPA model to obtain log-likelihoods for single subtrajectories, as well as how to generalize the BCPA model to obtain log-likelihoods for subtrajectory groups.

Whereas Brownian bridges used the observed points of movement data, with the BCPA model we have to choose what time-series variable of any given trajectory we wish to model. Two variables regularly used with the window-sweep BCPA algorithm are the persistent velocities or the turning speeds of an entity. However, since our segmentation model lies within a group context, we also will consider variables that make use of possible grouping aspects. We consider using the distance to the centroids or the distance to the k-nearest neighbor at each point of a trajectory. Here the centroid at some time t_i for trajectories T is the average point of the point set $T[t_i]$. These two variables only make sense in a group context however, as for a single trajectory they would both be zero at all times. We discuss these variables further down this section.

Let us first describe the persistent velocities and turning speeds. For some trajectory τ with timestamped points $\langle (z_1, t_1), (z_2, t_2), \ldots, (z_n, t_n) \rangle$, let $\Phi(t_i)$ be the absolute compass direction. The velocity $V(t_i)$ and turning angle $\Psi(t_i)$ are then written as:

$$V(t_i) = \frac{|z_i - z_{i-1}|}{t_i - t_{i-1}} \tag{2.14}$$

$$\Psi(t_i) = \Phi(t_i) - \Phi(t_{i-1}) \tag{2.15}$$

The persistent velocity V_p and turning speed V_t at time t_i are then defined as:

$$V_p(t_i) = V(t_i) \cdot \cos(\Psi(t_i)) \tag{2.16}$$

$$V_t(t_i) = V(t_i) \cdot \sin(\Psi(t_i)) \tag{2.17}$$

We use $V_p(\tau)$ and $V_t(\tau)$ to describe the sequence of persistence velocities or turning speeds of a (sub)trajectory τ. The difference between high and low V_p values is visualized in Figure 2.7, where

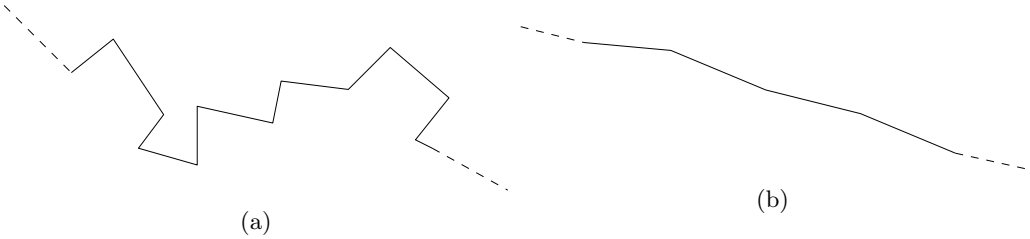


Figure 2.7: Two example trajectories highlighting the visual difference between different values of the persistent velocities $V_p(\tau)$. The trajectory in (a) has V_p values close to zero as it contains sharp turns where the velocity the trajectory had in its previous direction is almost entirely lost. In (b) we see the opposite, where almost straight lines result in the trajectory keeping its momentum in a single direction. This trajectory consequently has high V_p values.

two trajectories with low and high V_p values are shown. Of these two variables, the persistence velocity is considered to be a robust fit for the Gaussian processes the BCPA uses to model the movement features and as such is used the most often [21]. In theory, we could also try and obtain a likelihood formula that takes two autocorrelation values as arguments, and maximize our segment likelihoods with two parameters in our segmentation methods later on. These two parameters would then encapsulate both the persistent velocities and turning speeds. However, $V_p(\tau)$ and $V_t(\tau)$ are not necessarily independent, depending on the data τ is sampled from. Since finding the relationship between V_p and V_t is neither the focus of this work or within its scope, we opt to consider only one of the two at any time in our BCPA applications.

In a setting where we have some trajectory set $\mathcal{T}$ consisting of m trajectories, the centroid $\tilde{z}_i$ at any point in time t_i can be defined as:

$$\tilde{z}_i = \frac{1}{m} \sum_{\tau \in \mathcal{T}} \tau[t_i] \tag{2.18}$$

Here the points $\tau[t_i]$ are point vectors being summed and multiplied by a scalar, giving us the average point or centroid $\tilde{z}_i$. The distance for each trajectory τ at time t_i is then given by $dist(\tilde{z}_i, \tau[t_i])$. The distances to the k-nearest neighbor are defined similarly. For a trajectory τ at time t_i, let $\langle \tau_1[t_i], ..., \tau_{m-1}[t_i] \rangle$ be the sequence of points belonging to the other trajectories in $\mathcal{T}$, sorted on their distances to τ at time i, namely $\text{dist}(\tau[t_i], \tau_j[t_i])$ for $1 \leq j \leq m-1$. The distance to the k-nearest neighbor for τ at time t_i is then written as $dist(\tau[t_i], \tau_k[t_i])$ for a given $1 \leq k \leq m-1$.

From here we give a short description of how we can calculate log-likelihoods that indicate how well the model fits to a subtrajectory, and generalize this notion to groups. This log-likelihood, which we will write as $LL_{\text{BCPA}}(\tau[t_i, t_j])$ for some subtrajectory $\tau[t_i, t_j]$, is expressed in terms of a parameter called the autocorrelation coefficient ρ. The log-likelihood describes how well an autocorrelation coefficient value ρ fits to the subtrajectory. Although the implication of different values of ρ might not be immediately intuitive, a general interpretation is that higher ρ values indicate the modeled variables (persistent velocities, turning speeds or others) stay consistent over time. The autocorrelation coefficient thus essentially drops when a subtrajectory is considered where the modeled variable wildly varies. Let W denote the underlying Gaussian process we are modeling the model parameters of some subtrajectory $\tau[t_i, t_j]$ with. Note that we do the following steps with explicit notation of subtrajectories on some time interval $[t_i, t_j]$ instead of "complete" trajectories. This will help connect the formulas used here with the usage of the BCPA model in later sections where log-likelihoods are used mostly for subtrajectories. We use $\hat{\mu}$ to indicate the mean of the persistence velocities $V_p(\tau)$, and $\hat{\sigma}$ to indicate the standard deviation of $V_p(\tau)$. Finally, let $\delta_i := t_i - t_{i-1}$ to reduce notation complexity. We can write the probability density

function of W at some time t_i, written as W_i, with ρ as the autocorrelation coefficient as:

$$f(W_i \mid W_{i-1}, \hat{\mu}, \hat{\sigma}) = \frac{1}{\hat{\sigma}\sqrt{2\pi(1-\rho^{2\delta_i})}} \exp\Big(\frac{(W_i - \rho^{\delta_i}(W_{i-1} - \hat{\mu}))^2}{2\hat{\sigma}^2(1-\rho^{2\delta_i})}\Big) \tag{2.19}$$

Using the above probability density function, with W modeling $V_p(\tau)$, we can simply write the likelihood for a subtrajectory $\tau[t_i, t_j]$ with autocorrelation coefficient ρ as:

$$L(\tau[t_i, t_j], \rho, \hat{\mu}, \hat{\sigma}) = \prod_{k=i}^{j} f(W_k \mid W_{k-1}, \hat{\mu}, \hat{\sigma}) \tag{2.20}$$

Finally, the likelihood for a group of subtrajectories $\mathcal{T}[t_i, t_j]$ and a given autocorrelation coefficient ρ we define as the sum of the log-likelihoods of the BCPA fitted to each subtrajectory $\tau[t_i, t_j] \in \mathcal{T}[t_i, t_j]$:

$$LL_{\text{BCPA}}(\mathcal{T}[t_i, t_j], \rho) = \sum_{\tau \in \mathcal{T}} \log(L(\tau[t_i, t_j], \rho)) \tag{2.21}$$

Similarly to the log-likelihood of the Brownian bridges, for edge e in a representation graph $G_{\mathcal{S}}$ we can indicate the log-likelihood $LL(M(e) \mid \overline{\sigma}(e)) = LL_{\text{BCPA}}(M(e) \mid \overline{\sigma}(e))$ if the BCPA model was used in $\mathcal{S}$. When we write LL_{BCPA} we implicitly assume $\overline{\sigma}(e)$ represents only an autocorrelation coefficient ρ.

2.4.3 Location model

In this section we will describe how to obtain a log-likelihood function for a segmentation in the context of locality, by providing a location model for vertices in our representation graph that uses an ellipsoid normal distribution on sets of trajectory points. More formally, if v is a vertex in the representation graph $\mathcal{S}$, we want to express $LL(P(v) \mid \overline{\theta}(v))$. If $\mathcal{T}$ is the set of trajectories segmented by $\mathcal{S}$, it is easy to see that $P(v)$ matches some set of points $T[i]$ at time t_i, for some set $T \subseteq \mathcal{T}$. To be able to formally define the location model we use, we have to be explicit in our notation of points within $T[i]$, and thus we continue with this style of notation described in Section 2.1.

We can consider the set of points $T[i]$ at time t_i to be sampled from some distribution. Let this distribution be a multivariate normal distribution on random variables X and Y, representing the x and y coordinates of the trajectory locations, with means μ_x, μ_y and variances σ_x^2, σ_y^2 respectively. The correlation coefficient of the two variables is defined as:

$$\rho = \frac{\text{cov}(X, Y)}{\sigma_x \sigma_y} \tag{2.22}$$

where $\text{cov}(X, Y)$ is the covariance of X and Y, in turn defined as:

$$\text{cov}(X, Y) = \frac{\sum (x_i - \mu_x)(y_i - \mu_y)}{n - 1} \tag{2.23}$$

We can use the correlation coefficient in the following density function for the bivariate distribution on X, Y:

$$f(x, y \mid \mu_x, \mu_y, \sigma_x, \sigma_y, \rho) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} e^{-\frac{1}{2(1-\rho^2)}\left[(\frac{x-\mu_x}{\sigma_x})^2 - 2\rho(\frac{x-\mu_x}{\sigma_x})(\frac{y-\mu_y}{\sigma_y}) + (\frac{y-\mu_y}{\sigma_y})^2\right]} \tag{2.24}$$

Given our sample point set $T[i]$ at time t_i, with measured means $\hat{\mu}_x$ and $\hat{\mu}_y$, we can compute the likelihood of this sample set for given variances $\hat{\sigma}_x, \hat{\sigma}_y$ (and $\hat{\rho}$, which can be computed from $T[i]$, $\hat{\sigma}_x$ and $\hat{\sigma}_y$) as:

$$L(T[i] \mid \hat{\sigma}_x, \hat{\sigma}_y) = \prod_{(x,y) \in T[i]} f(x, y \mid \hat{\mu}_x, \hat{\mu}_y, \hat{\sigma}_x, \hat{\sigma}_y, \hat{\rho}) \tag{2.25}$$

It is important to note that in this likelihood computation, the likelihood for each point set $T[i]$ uses the estimated mean of only the locations in $T[i]$. The log-likelihood of this location model can then be expressed using only $\hat{\sigma}_x$ and $\hat{\sigma}_y$ as parameters. Switching back to the representation graph $\mathcal{S}$ perspective; for a vertex v with points $P(v)$, we can write the log-likelihood of the location model with parameters $\overline{\theta}(v)$ as:

$$LL_{\text{LOC}}(P(v) \mid \overline{\theta}(v)) = \log L(P(v) \mid \overline{\theta}(v)) \tag{2.26}$$

Again, note that we assume implicitly that $\overline{\theta}(v)$ represents model parameters $\hat{\sigma}_x$ and $\hat{\sigma}_y$ when using LL_{LOC}.

A general note on this model is that by only using one value for both its horizontal and vertical variance, equivalent to having $\hat{\sigma}_x = \hat{\sigma}_y$, we can simplify the model to a circular normal distribution instead. This also leaves only one parameter in the log-likelihood function to be estimated when maximizing the log-likelihood.

Chapter 3

Heuristic group segmentation methods

In Chapter 2 we have defined a representation for our model-based segmentations of multiple trajectories, and we have defined an information criterion to evaluate these segmentations. In this chapter we introduce three heuristic approaches which tackle the optimization problem defined in Section 2.3.2.

Two of the heuristic methods use a previously developed method for segmenting single trajectories using movement models which are not generalized for groups [3]. We can apply this algorithm on a set of subtrajectories which we assume to be a group, using the generalized movement models to obtain log-likelihoods instead. In Section 3.1 we give a short description of this algorithm and its usage in our segmentation methods.

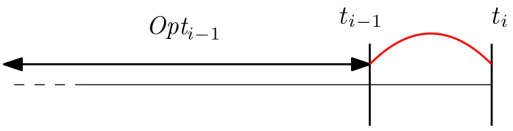
The generalization of the segmentation problem to multiple trajectories involves finding sets of subtrajectories that are similar in terms of movement and closeness. The segments in our representation can be compared to clusters, which closely relates the grouping part of our segmentation problem to the field of trajectory clustering. We describe an approach in Section 3.2 that involves clustering subtrajectories. The subtrajectory clustering problem has been related to the Set Cover problem [1], which forms the general idea behind the clustering segmentation approach we propose.

To see whether combining segmentations to create one segmentation for a trajectory set is a viable approach, we designed an incremental segmentation approach in Section 3.3. With this approach we combine the segmentations obtained by segmenting given trajectories separately, while trying to minimize the IC of the overall segmentation. We utilize the fact that the IC for a segmentation can be obtained by summing the IC values of different parts of the segmentation, allowing us to quickly compute changes in the IC of a segmentation when merging or adding segments.

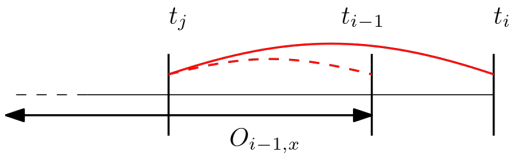
Finally, in Section 3.4 we solve the segmentation problem with a two-step approach. In the first step we focus on obtaining a graph structure that represents the underlying grouping behavior of the input trajectory set. The second step can then segment the edges in this graph, which represent sets of subtrajectories, assuming that these subtrajectories are grouped in the time interval represented by each edge. For the first step we use an existing method for finding this trajectory grouping structure, defined in [12]. In the second step we apply the model-based single trajectory segmentation algorithm in [3], using the generalized movement models from Section 2.4 to apply the algorithm to groups.

3.1 Segmenting groups

An existing method of segmenting a single trajectory with movement models is presented in [3]. Similar to how we evaluate our segmentation representation, this approach defines an optimization



(a) Appending Opt_{i-1} with the single link between timestamps t_{i-1} and t_i (red arc) to obtain the optimal segmentation for entry $O_{i,x}$.



(b) Extending the last segment of $O_{i-1,x}$ to t_i to obtain the optimal segmentation for entry $O_{i,x}$. Here j is the start time of the last segment of $O_{i-1,x}$. The last segment of $O_{i-1,x}$ (dashed red arc) is replaced with the extended segment (red arc) on interval $[t_j, t_i]$.

Figure 3.1: A visualization of how $O_{i,x}$ can be calculated as one of two options.

goal to minimize a given information criterion (IC) over segmentations of some trajectory τ. The optimal segmentation for subtrajectory $\tau[t_1, t_i]$, named Opt_i, is computed for increasing values of i. The last segmentation Opt_n is the optimal segmentation for the full trajectory $\tau[t_1, t_n]$ according to the IC.

To compute Opt_i for some time t_i the algorithm simultaneously builds a two-dimensional table O where the dimensions are the timestamps $t_1, ..., t_n$ and a discrete set of parameters $\sigma_1, ..., \sigma_m$. The set of discrete parameters is computed in a pre-processing step where for all possible time intervals $[t_i, t_j]$ for $1 \leq i < j \leq n$, the model parameter that maximizes the log-likelihood for subtrajectory $\tau[t_i, t_j]$ is added to the set of parameters. We let k denote the number of parameters in this set. Entry $O_{i,x}$ then contains the optimal IC for a segmentation of $\tau[t_1, t_i]$ whose final segment has a corresponding movement parameter value of σ_x. The segmentation represented by Opt_i is simply the minimal IC value found among all $O_{i,x}$ for $1 \leq x \leq k$. For a table entry $O_{i,x}$, there are two possible options:

1. Append: The optimal segmentation for $O_{i,x}$ is equal to the optimal segmentation Opt_{i-1} appended with a new segment $\tau[t_{i-1}, t_i]$ appended at the end, with an assigned movement model parameter of σ_x. The new IC obtained by appending is computed as:

$$\text{IC}_{ap}(O_{i,x}) = \text{IC}(Opt_{i-1}) + p - 2 \cdot LL(\tau[t_{i-1}, t_i] \mid \sigma_x)$$

2. Extend: The optimal segmentation for $O_{i,x}$ is equal to the segmentation of $O_{i-1,x}$, where the last segment is extended with the link $\tau[t_{i-1}, t_i]$. The new IC obtained by extending is computed as:

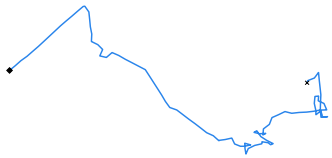
$$\text{IC}_{ex}(O_{i,x}) = \text{IC}(O_{i-1,x}) + p - 2 \cdot LL(\tau[t_{i-1}, t_i] \mid \sigma_x)$$

Either the append or the extend option must give an optimal segmentation for entry $O_{i,x}$ and the choice between these options can be checked just using already filled entries of O and Opt. In Figure 3.1 these options are visualized. Recall that p is used to indicate the complexity penalty weight. The IC for entry $O_{i,x}$ can be computed as the minimum IC of the append and extend options:

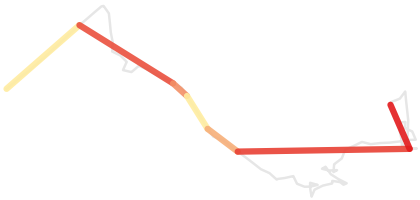
$$\text{IC}(O_{i,x}) = \min(\text{IC}_{ap}(O_{i,x}), \text{IC}_{ex}(O_{i,x})) \tag{3.1}$$

The problem is then solved with a dynamic programming approach to fill out the table and ultimately get an optimal segmentation for τ, according to a given IC. This algorithm computes the optimal segmentation Opt_n in $O(nk)$ time.

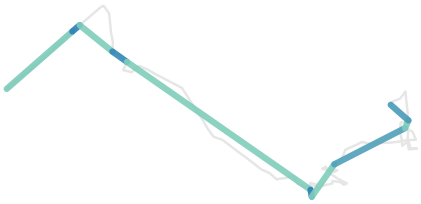
This algorithm can by itself be used for single trajectories, given a movement model to calculate the log-likelihoods used in Equation 3.1. A few examples of such segmentations, using the Brownian bridge, BCPA with V_p and BCPA with V_t movement models, are shown in Figure 3.2. However, with the movement models we have generalized to groups in Section 2.4, we can also use



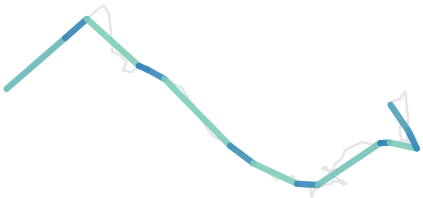
(a) A generated trajectory τ with $n = 100$ points.



(b) A segmentation of τ using the approach from Section 3.1 with Brownian bridges as a movement model, and complexity weight $p = 10$. Segments have a color ranging from yellow (low diffusion coefficient) to red (high diffusion coefficient).



(c) A segmentation of τ using the approach from Section 3.1 with BCPA modeling the persistent velocities $V_p(\tau)$, and complexity weight $p = 3$. Segments have a color ranging from teal (low autocorrelation) to dark blue (high autocorrelation).



(d) A segmentation of τ using the approach from Section 3.1 with BCPA modeling the turning speeds $V_t(\tau)$, and complexity weight $p = 3$. Segments have a color ranging from teal (low autocorrelation) to dark blue (high autocorrelation).

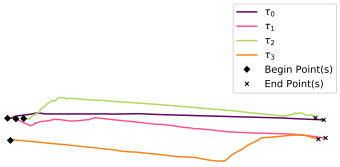
Figure 3.2: Segmentations of a single generated trajectory using the method described in Section 3.1.

this algorithm to segment a group of (sub)trajectories. Given such a set $T \subseteq \mathcal{T}$, each Opt_i then simply represents the optimal segmentation for $T[t_1, t_i]$.

Note that if we use this algorithm with a set of trajectories or subtrajectories, we assume that they form a group from start to end. If the trajectories or subtrajectories given are not close or have different movement behavior, the resulting segmentation of this algorithm might poorly represented all the trajectories in the set. In the heuristic methods described in the next few sections, we will use this algorithm strictly on single trajectories or trajectories which we assume to be grouped. We use $\mathcal{S}_{DP}(T)$ to indicate the segmentation of some (sub)trajectory set T, or $\mathcal{S}_{DP}(\tau)$ for the segmentation of a single (sub)trajectory τ, whenever we use this algorithm in the remainder of this work.

3.1.1 Maximizing likelihoods with model parameters

In Sections 2.4.1 and 2.4.2 we have described how to compute the Brownian bridge and BCPA log-likelihoods for trajectory groups, or edges in our segmentation representation. We also have defined a simple location model to compute log-likelihood for its vertices in Section 2.4.3. The information criterion defined in Section 2.3.1 does not specify what movement model and location model parameters are used to calculate the log-likelihoods of vertices and edges in the segmentation representation. Since we want higher log-likelihoods which result in a lower IC score, we ideally would have parameters such that the log-likelihood for any given vertex or edge is maximized. These parameters then represent location or movement most accurately. A method which searches for parameters to maximize a likelihood is often called Maximum Likelihood Estimation (MLE). In this section we will shortly go over ways we estimate the best fitting parameters in our models.



(a) A set $\mathcal{T} = \{\tau_0, \tau_1, \tau_2, \tau_3\}$ of four generated trajectories with $n = 100$ points each.

(b) A segmentation of $\mathcal{T}$ using the approach from Section 3.1 with Brownian bridges as a movement model, assuming that the trajectories in $\mathcal{T}$ are grouped.

Figure 3.3: A set of trajectories being segmented using the method described in Section 3.1, while it is assumed that they form a group from start to end.

In related work on single-trajectory segmentation using Brownian bridges, finding the optimal diffusion coefficient such that the likelihood of a group segment is maximized is solved by using a golden section search [3]. This method is able to optimize σ_m^2 such that a maximum of the likelihood function is found, and since in a single-trajectory context there can only be one maximum likelihood value this method works for parameter estimation for Brownian bridges. However, in our case the likelihood function of a group is a sum of multiple likelihood functions, and there can be multiple maximum likelihood values for different values of σ_m^2. As such, we apply a different one-sided bounded search algorithm for finding the optimal diffusion coefficient for a group segment.

We run into the same issue when optimizing the autocorrelation coefficient for group segments in the BCPA model. Since we know the value of the autocorrelation coefficient are bounded in the interval $[0, 1]$, we can use a search algorithm for finding maximums bounding the domain in $[0, 1]$.

The location model we defined takes two parameters, $\hat{\sigma}_x$ and $\hat{\sigma}_y$ to describe the horizontal and vertical variance. Although one could be tempted to use the fastest algorithm to optimize these parameters for a set of points, the amount of points for which optimization needs to occur is at most the amount of trajectories in the segmented set, $|\mathcal{T}|$. Especially when studying animal movement, the amount of trajectories often do not seem to be so large as to cause the optimization of the location models to have a higher impact on running time than the optimization of the movement models.

3.2 Clustering method

The segments in the model-based representation can be related to clusters, in the sense that they contain subtrajectories that are similar based on movement or closeness measures. To obtain a complete segmentation for a trajectory set, we have to select a number of clusters such that the full trajectory set is covered by the selected clusters. In related work on subtrajectory clustering, the cluster selection problem was related to the Set Cover problem [1]. A Set Cover problem instance $(\mathcal{U}, \mathcal{C})$ describes a universe U with elements $\{x_1, \ldots, x_n\}$ and a set of subsets $\mathcal{C} = \{C_1, \ldots, C_m\}$, where $C_i \subseteq \mathcal{U}$ for $1 \leq i \leq m$. The required solution to this problem is a set $S \subseteq \mathcal{C}$ that satisfies $\bigcup_{C \in \mathcal{S}} C = \mathcal{U}$. We will relate our cluster selection problem to the Set Cover problem as well, and use this relation to build a solution from the cluster set.

In Section 3.2.1 we will go into more detail on the construction and definition of the clusters in the cluster set $\mathcal{C}$, and what choices we make regarding the relevance of a cluster. Here we also discuss the potential size of $\mathcal{C}$, and give a formula for a cluster's score using several aspects from the IC definition. Section 3.2.2 describes how exactly we cover the trajectory set, or $\mathcal{U}$, with a

selection of clusters that represents a valid segmentation. The details of the relation to the Set Cover problem and the greedy solution are explained in Section 3.2.3.

3.2.1 Constructing relevant clusters

Given a trajectory set $\mathcal{T}$, we define a cluster c as a set of subtrajectories on some time interval. Let the set of the trajectories corresponding to a cluster c be written as T_c. We then write $t_\alpha(c)$ and $t_\beta(c)$ to indicate the start and end time of c, such that c is defined on the interval $[t_\alpha(c), t_\beta(c)]$. Using the notation convention described in Section 2.1, we can indicate the set of subtrajectories of c on its time interval with $T_c[t_\alpha(c), t_\beta(c)]$. A segmentation of a trajectory set $\mathcal{T}$ can then be expressed in terms of a set of non-overlapping clusters. Note that this set can also contain clusters containing a single subtrajectory.

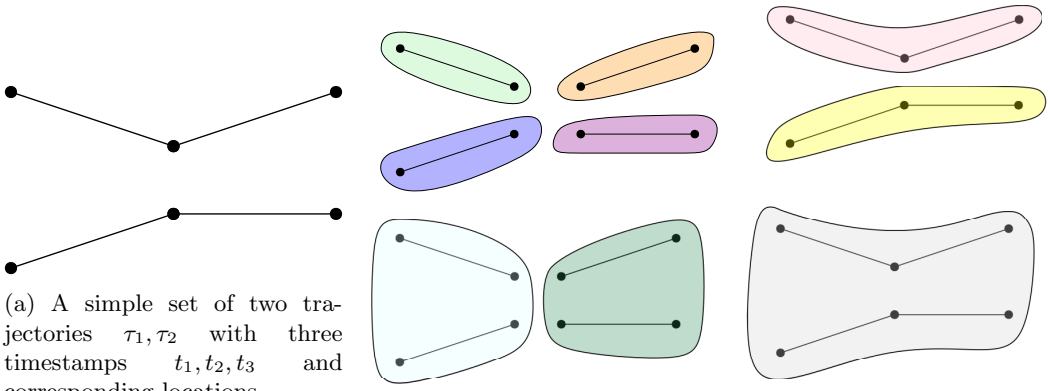
To construct these clusters, let us consider some time interval $[t_i, t_j]$. Let c denote a potential cluster with $t_\alpha(c) = t_i$ and $t_\beta(c) = t_j$. If T_c is some subset of $\mathcal{T}[t_i, t_j]$, we consider c to be a relevant cluster if the following all hold:

1. The movement of the subtrajectories in T_c is similar.
2. The set of start points of the subtrajectories in T_c satisfies some closeness measure.
3. Similarly, the set of end points satisfies some (other) closeness measure.

The above requirements still leave some details open to interpretation. While in [1] the similarity along the subtrajectories is determined using Fréchet distance, we wish to express the similarity with generalized movement models. One way of expressing this is performing a check whether the maximized log-likelihood of the movement model on the subtrajectories in T_c is above a certain threshold γ. More formally, we require $LL(T_c[t_i, t_j] \mid \overline{\sigma}) \geq \gamma$, where $\overline{\sigma}$ represents the movement model parameters which maximize the log-likelihood for $T_c[t_i, t_j]$.

Next, the closeness measure for the start and end points can either be a location model or a simple distance threshold. We opt for a distance threshold which removes the need to maximize location model likelihoods as well as movement model likelihoods for all possible clusters. With some point distance threshold δ, we can either require single or full linkage of the start or end points. Let us consider the point set $T_c[t]$ for some time t, which, without loss of generality, is either the start or end time of some potential cluster c. If we let the points in this set represent vertices of a graph, where two distinct points $z_i, z_j \in T_c[t]$ share an edge if $dist(z_i, z_j) \leq \delta$, single linkage requires this graph to be connected. Full linkage requires the graph to be a clique, implying that for every two distinct points z_i, z_j we have $dist(z_i, z_j) \leq \delta$. When using this method in later sections, we will only use single linkage. For some specific applications of this method, where a bit more context on the grouping characteristics of the entities is available, using full linkage might prove more useful.

When looking at a time interval $[t_i, t_j]$, we have now determined what clusters we want to consider for a segmentation on this time interval, based on the three constraints described above. To reduce the number of total clusters to consider for a segmentation of trajectory set $\mathcal{T}$, with $|\mathcal{T}| = m$, we construct a cluster set $C[t_i, t_j]$ on time interval $[t_i, t_j]$ that contains at most m clusters. Let $T := \mathcal{T}[t_i, t_j]$ denote all subtrajectories on time interval $[t_i, t_j]$. We create a cluster c containing a random subtrajectory $\tau[t_i, t_j] \in T$, setting $T_c := \{\tau[t_i, t_j]\}$ and removing $\tau[t_i, t_j]$ from T. Then we try to add other single subtrajectories $\tau'[t_i, t_j] \in T$ to $T_c[t_i, t_j]$ and check whether c still satisfies the constraints for relevant clusters. If $\tau'[t_i, t_j]$ can be added to c without breaking the constraints, we let $T_c[t_i, t_j] := T_c[t_i, t_j] \cup \{\tau'[t_i, t_j]$ and remove $\tau'[t_i, t_j]$ from T. When all subtrajectories have been considered or added to the cluster we add c to $C[t_i, t_j]$ and repeat this process by creating another cluster with one of the remaining trajectories in T and checking whether we can add single subtrajectories to this new cluster. In the event that a cluster c is created with some subtrajectory $\tau[t_i, t_j] \in T$ and no other subtrajectory can be added without breaking the constraints, which includes the case where the maximized log-likelihood of the movement model $LL(\tau[t_i, t_j] \mid \overline{\sigma})$ is below the threshold γ, we still add c to $C[t_i, t_j]$. If $C[t_i, t_j]$ does not



(a) A simple set of two trajectories τ_1, τ_2 with three timestamps t_1, t_2, t_3 and corresponding locations.

(b) All clusters that can be created from τ_1 and τ_2, colored differently for clarity. Note that we have not visualized clusters on only one timestamp, such as the set of two points $\{\tau_1[t_1], \tau_2[t_1]\}$.

Figure 3.4: An illustration of clusters constructed from two example trajectories.

contain some subtrajectory $\tau[t_i, t_j]$ there is a possibility that this subtrajectory is not covered by the final segmentation due to the way we select clusters for the segmentation. Following these steps, we obtain at least one, and at most m clusters for any time interval $[t_i, t_j]$.

For a trajectory set $\mathcal{T}$ with trajectories having n points and corresponding timestamps, there are $O(n^2)$ time intervals to consider. Let $C = \{C[t_i, t_j] \mid 1 \leq i < j \leq n\}$, where each $C[t_i, t_j]$ is the set of relevant clusters obtained for time interval $[t_i, t_j]$ with the method above. Since each $C[t_i, t_j]$ contains at most m clusters, we obtain a total of $O(n^2 m)$ relevant clusters for $\mathcal{T}$. A small example of two trajectories and their corresponding possible clusters are shown in Figure 3.4a and Figure 3.4b, respectively. The large number of clusters to consider can be scaled down by first segmenting each trajectory separately using the movement model applied in the clustering algorithm, obtaining a segmentation $\mathcal{S}(\tau)$ for each $\tau \in \mathcal{T}$. Let $br(\mathcal{S}(\tau))$ be the set of timestamps obtained from all vertices in the segmentation of τ. In other words, $br(\mathcal{S}(\tau))$ represents the timestamps where the behavior of τ changes according to the segmentation. Now let $U = \bigcup_{\tau \in \mathcal{T}} br(\mathcal{S}(\tau))$. If we only consider creating clusters on time intervals $[t_i, t_j]$ with $t_i, t_j \in U$ and $t_i < t_j$, then the amount of relevant clusters can be decreased significantly depending on the complexity of the segmentations of each $\tau \in \mathcal{T}$. We will use this tactic in the experiments on real data to speed up the computation time.

3.2.2 Cluster selection

We now have a set $\mathcal{C}$ consisting of $O(n^2 m)$ clusters. Note the time intervals of different clusters in $\mathcal{C}$ may overlap. We can produce a valid segmentation from $\mathcal{C}$ by selecting a subset $\mathcal{S} \subseteq \mathcal{C}$ for which the following requirements hold:

1. For any two clusters $c_1, c_2 \in \mathcal{S}$ for which there is time overlap, e.g. $t_\alpha(c_1) < t_\alpha(c_2) < t_\beta(c_1)$ or $t_\alpha(c_2) < t_\alpha(c_1) < t_\beta(c_2)$, the corresponding trajectory sets T_{c_1} and T_{c_2} must be disjoint $T_{c_1} \cap T_{c_2} = \emptyset$.

2. Conversely, if two clusters $c_1, c_2 \in \mathcal{S}$ share at least one trajectory such that $T_{c_1} \cap T_{c_2} \neq \emptyset$, their time intervals must not overlap.

3. For every trajectory $\tau \in \mathcal{T}$, there must exist a set of clusters $\mathcal{S}(\tau) \subseteq \mathcal{S}$ that together represent τ. It should be possible to write this set $\mathcal{S}(\tau)$ as a sequence of clusters $c_1, \ldots, c_k$ where the following hold:

   (a) Each cluster $c_i \in \mathcal{S}(\tau)$ contains a subtrajectory of τ, such that we have $\tau[t_\alpha(c_i), t_\beta(c_i)] \in T_{c_i}[t_\alpha(c_i), t_\beta(c_i)]$.

(b) For each pair of consecutive clusters c_i, c_{i+1} in the sequence we have $t_\beta(c_i) = t_\alpha(c_{i+1})$, such that there are no gaps between clusters in the sequence.

Since there can be multiple ways to select a cluster set $\mathcal{S}$ to represent a valid segmentation of $\mathcal{T}$, we have to make choices between different clusters containing different sets of subtrajectories on varying time intervals. Following the reasoning behind the information criterion, we want to balance the resulting segmentation complexity with the log-likelihoods of location and movement models. We can give clusters a score based on their contribution to the IC of the resulting segmentation. However, we cannot encode the effect in terms of complexity that picking some cluster c has on the complete segmentation, yet we still have to combat overfitting by continuously letting smaller clusters with better fitted models be picked. Picking a larger cluster that contains a large number of subtrajectories and covers a large time interval will most likely result in less complexity in the overall segmentation, therefore countering overfitting. We use the log-likelihood $LL(T_c[t_i, t_j] \mid \overline{\sigma})$ together with the size of the cluster, defined by both its size $|T_c|$ and length $t_\beta(c) - t_\alpha(c)$, to express the estimated usefulness of a cluster with a scoring function $F(c)$. This heuristic measure allows us to make decisions on which cluster to select in a segmentation.

We consider a scoring function which adds parameterized size and length bonuses to the log-likelihood value to get a cluster score. For better readability, let $LL(c)$ denote the maximized movement model log-likelihood of a cluster c, and $t_{total} = t_n - t_1$ denote the total duration of trajectories in $\mathcal{T}$. Then we have:

$$F(c) = LL(c) + p_{size}\frac{|\mathcal{T}(c)|}{|\mathcal{T}|} + p_{length}\frac{t_\beta(c) - t_\alpha(c)}{t_{total}} \tag{3.2}$$

The size and duration score parameters p_{size} and p_{length} in $F(c)$ work similarly to the IC complexity weight p used in Equations 2.4 and 2.5, giving some control over the segmentation construction in terms of segment size and length. However, this also requires two parameters to be tweaked for different data sets which makes experimentation on data difficult.

3.2.3 Weighted Set Cover and greedy solution

Now that we have formulated a scoring function in Equation 3.2, we are still left with the problem of which clusters to select from a cluster set C, based on their scores. As seen in [1] we can relate the Weighted Set Cover problem to this cluster selection problem. Recall that the Set Cover problem has already been introduced in the introduction of Section 3.2. Here we first introduce the Weighted Set Cover problem, then introduce the relation to our cluster selection problem, and finally describe how we use a greedy solution to heuristically solve the cluster selection problem.

To convert an instance $(\mathcal{U}, \mathcal{C})$ of the Set Cover problem into an optimization problem, we apply a weight function to the sets in $\mathcal{C}$, denoted as $w : \mathcal{C} \to \mathbb{R}^+$. We then have a Weighted Set Cover problem instance, where the optimal solution is a set $S \subseteq \mathcal{C}$ which minimizes the total weight of its picked sets $\sum_{C \in S} w(C)$ while satisfying $\bigcup_{C \in \mathcal{S}} C = \mathcal{U}$.

We now let $\mathcal{U}$ denote all the points of the trajectories in a trajectory set $\mathcal{T}$, such that $\mathcal{U} = \bigcup_{\tau \in \mathcal{T}} \{z_i \mid (z_i, t_i) \in \tau\}$. We still let $\mathcal{C} = \{C_1, \ldots, C_m\}$, but each C_i now represents a set of points corresponding to the subtrajectories in some cluster c_i. This can be written as $C_i = \{z \in \tau \mid \tau \in T_{c_i}\}$. Additionally, let the weight of a set $C_i \in \mathcal{C}$ be the score of corresponding cluster c_i, e.g. $w(C_i) = F(c_i)$. In the previous section we have outlined the requirements for the selected cluster set in order to obtain a valid segmentation. A set of clusters S for which those requirements hold clearly satisfies the set cover problem, as all trajectories are fully covered by the clusters in $\mathcal{S}$. However, a solution to the Weighted Set Cover problem does not incorporate the non-overlapping aspect we demand from selected clusters. This means that a solution the Weighted Set Cover problem does not necessarily solve the cluster selection problem, as the requirements may not be met.

The optimal solution to a Weighted Set Cover instance can be approximated with a greedy approach, picking the set with the lowest weight each iteration. We do the same, adding the cluster $c \in \mathcal{C}$ with the highest score, such that $F(c) \geq F(c')$ for all $c' \in \mathcal{C}$, to the cluster set

$\mathcal{S}$ each iteration. However, to adhere to the valid cluster selection constraints described in the previous section, we remove clusters that overlap with c from $\mathcal{C}$ when c is picked. Clusters c and c' overlap if the following two statements hold:

1. $T_{c'} \cap T_c \neq \emptyset$
2. $(t_\alpha(c) < t_\alpha(c') < t_\beta(c)) \vee (t_\alpha(c') < t_\alpha(c) < t_\beta(c'))$

We continue the greedy selection of clusters this way until $\mathcal{C}$ is empty. At that point it may still be the case that there are subtrajectories left uncovered by $\mathcal{S}$. We add all uncovered subtrajectories as new single-subtrajectory clusters to $\mathcal{S}$ to be able to fully cover $\mathcal{T}$. This gives us a heuristic solution to the segmentation problem using a clustering approach.

3.3 Incremental method

Given the complex nature of the segmentation problem for multiple trajectories, we explore an incremental approach, where we build a segmentation from smaller segmentation parts. In Section 3.1 we have seen a segmentation approach that finds the optimal segmentation by temporally iterating over the given trajectory. Instead of finding the optimal segmentation at every time t_i, we consider finding the optimal segmentation for larger subsets $T \subseteq \mathcal{T}$, given the segmentation of T minus some trajectory τ'. More formally, we define this subproblem as follows. Given a trajectory τ, a trajectory set T and its segmentation $\mathcal{S}(T)$, can we find a segmentation $\mathcal{S}(T \cup \{\tau\})$ in a reasonable time.

Starting with one single-trajectory segmentation, we add other such segmentations to it repeatedly, eventually resulting in a segmentation for the full trajectory set. To realize this method, we formalize a way to add a single-trajectory segmentation to a multi-trajectory segmentation. In the remainder of this section, we will refer to segmentations S as sets of segments representing movement. However, keep in mind that the segmentation representation we are building towards with this approach is not defined as such a set.

Given a segmentation S that represents one or more trajectories and a segmentation S_τ that only represents trajectory τ, the goal is to create a new segmentation S' that represents the segmentation S combined with τ. Let H denote a chronologically ordered set of timestamps for which S contains one or more vertices. Assume we have a segmentation S_τ that represents only τ, which is simply a graph representing a path. For each time $t \in H$, do the following:

1. Let $S' := S \cup S_\tau$. This is the segmentation where τ is not grouped with any of the trajectories represented in S at any time.
2. Let $S(t)$ denote the segments $s \in S$ that have starting time $t_\alpha(s) = t$ and some end time $t_\beta(s)$. Consider some t for which $S(t) \neq \emptyset$. For each segment $s \in S(t)$ we then compute the information criterion value of a modified segmentation of S' where the subtrajectory $\tau[t_\alpha(s), t_\beta(s)]$ is added to s, while the remainder of τ is now represented by a set of simple paths in S_τ. Note that there can be multiple simple paths since adding $\tau[t_\alpha(s), t_\beta(s)]$ to s splits of the path formed by S_τ. Figure 3.5 shows the merging of the subtrajectory of τ with s. In multiple iterations of this algorithm S_τ might be split more often or even completely merged with S. Additionally, the merge might create a new vertex in S_τ if the there is no vertex in S_τ with time equal to $t_\alpha(s)$ or $t_\beta(s)$. This event is also shown in Figure 3.5. We then compute the IC score of S' after this potential merge which we write as $\mathrm{IC}_s(S')$. Let $T_{del}(s)$ be the segments that will be deleted due to the merge, and let $T_{add}(s)$ be the segments that are added. We interpret the merging of subtrajectory $\tau[t_\alpha(s), t_\beta(s)]$ with s as the deletion of s combined with the addition of segment $s \cup \tau[t_\alpha(s), t_\beta(s)]$, therefore these segments are contained in $T_{del}(s)$ and $T_{add}(s)$ respectively. Then, we compute the IC score of S' before the merge of τ into s as:

$$\mathrm{IC}_{s,before}(S') = \sum_{\tau_i \in T_{del}(s)} -2 \cdot LL(\tau_i \mid \overline{\sigma}_i) + p \tag{3.3}$$

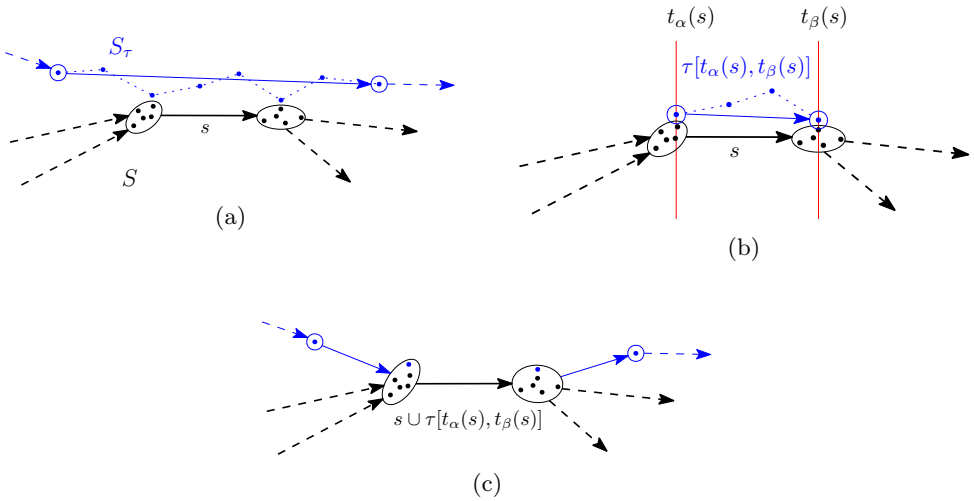


Figure 3.5: A visual explanation of how some subtrajectory $\tau[t_\alpha(s), t_\beta(s)]$ is merged into a segment $s \in S(t)$. (a) Segmentation $S' = S \cup S_\tau$. Component S_τ is drawn in blue, S in black. The intermediate points of the segment belonging to S_τ are connected with a dotted blue line. (b) The time interval $[t_\alpha(s), t_\beta(s)]$ (red lines) forms a subtrajectory on τ, which is then checked for closeness of starting points and end points to s. (c) The subtrajectory $\tau[t_\alpha(s), t_\beta(s)]$ is merged into s, leaving two (new) segments in S_τ.

In a similar fashion, the IC score after the merge is computed as:

$$\mathrm{IC}_{s,after}(S') = \sum_{\tau_i \in T_{add}(s)} -2 \cdot LL(\tau_i \mid \overline{\sigma}_i) + p \tag{3.4}$$

Here we implicitly let $\overline{\sigma}_i$ denote the optimal model parameter assigned to subtrajectory τ_i. For segments that will be deleted, namely s and one or more segments in S_τ, we already know their parameters from S' before the merge. The movement model is refitted to $s \cup \tau[t_\alpha(s), t_\beta(s)]$, while new segments in S_τ are left with the parameters they were assigned before they are split up. In total, this requires us to only recalculate the movement model parameters for one subtrajectory set. We then write the change in IC score for the merge of s as $\mathrm{IC}_{s,change}(S') = \mathrm{IC}_{s,after} - \mathrm{IC}_{s,before}$.

3. We then consider the segment s for which the IC score change caused by merging decreased the most. More formally, we have $s_{max} := argmin_{s \in S(t)} \mathrm{IC}_{s,change}(S')$.

4. If $\mathrm{IC}_s(S') \leq \mathrm{IC}(S')$, modify S' such that the subtrajectory $\tau[t_\alpha(s_{max}), t_\beta(s_{max})]$ is added to s_{max}, and the subtrajectories of $\tau[t_{prev}, t_\alpha(s_{max})]$ and $\tau[t_\beta(s_{max}), t_{next}]$ are represented by new segments $s' \in S_\tau$. Here t_{prev} is the first timestamp before $t_\alpha(s)$ in S_τ, and t_{next} is the timestamp of the first timestamp after $t_\beta(s)$ in S_τ.

5. Let $\mathrm{IC}(S') := \mathrm{IC}_{s,after}(S')$.

The above steps can then be repeated with $S := S'$ and $x := x(s_{max})$, until all times in H have been processed. Doing this for all trajectories then gives a segmentation representing the given trajectory set $\mathcal{T}$.

3.4 Two-Step method

In the IC defined in Section 2.3, both the log-likelihoods of locations at vertices and movement at edges define the quality of any given segmentation. We can divide the objective of minimizing

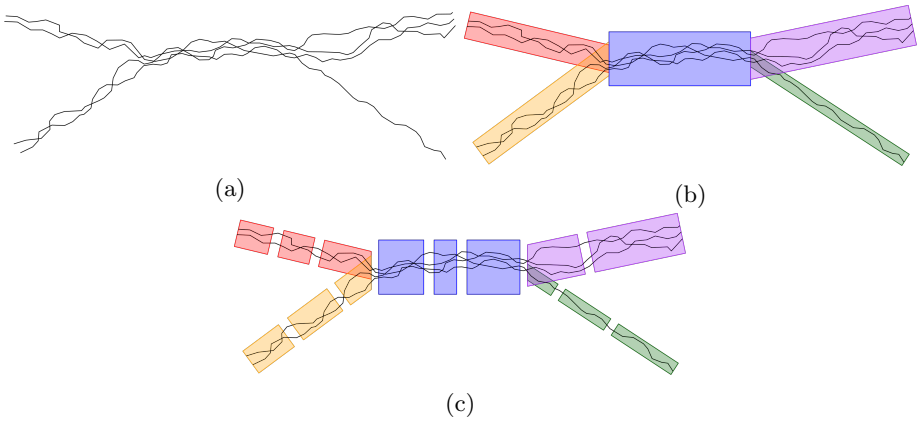


Figure 3.6: A visual example of a two-step approach for segmentation. In (a) we see an example set of trajectories. A grouping structure for these trajectories is shown in (b). The different color boxes represent different groups, where wider boxes indicate more subtrajectories contained in the group. After obtaining this grouping structure, we then further segment the groups based on movement, shown in (c).

this IC into finding what trajectories are grouped on which time intervals and further segmenting these groups. With this strategy in mind we present a two-step method for segmenting a set of trajectories. In Figure 3.6 we visually outline the separation of the grouping and movement segmentation of such a two-step approach. We use the *trajectory grouping structure* algorithm described in Section 2.2.1 to find groups in the given trajectory set.

Given a grouping structure $\mathcal{R}$, the second step consists of segmenting the sets of subtrajectories corresponding to groups $r \in \mathcal{R}$. For each group $r \in \mathcal{R}$, the set of its subtrajectories is denoted by $T_r[t_\alpha(r), t_\beta(r)]$. We use the generalized model-based segmentation algorithm described in Section 3.1 to get the segmentations for each group. Recall that we use $\mathcal{S}_{DP}(T_r[t_\alpha(r), t_\beta(r)]$ to indicate the resulting segmentation of the group segmentation algorithm for input $T_r[t_\alpha(r), t_\beta(r)]$. The segmentation $\mathcal{S}$ for the complete trajectory set $\mathcal{T}$ is then simply obtained with the union of the group segmentations:

$$\mathcal{S} = \bigcup_{r \in \mathcal{R}} S_r \tag{3.5}$$

Chapter 4

Experiments

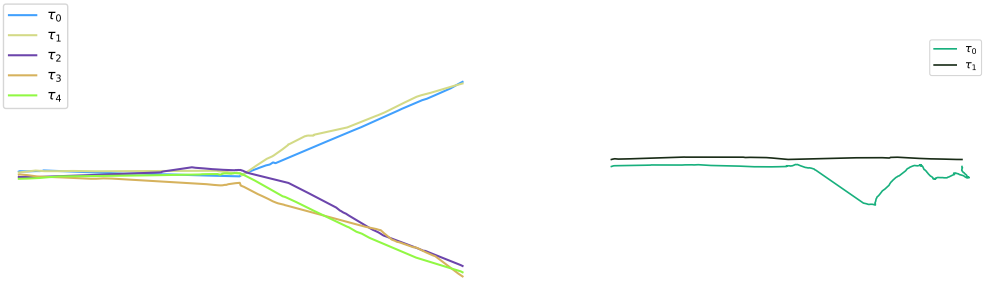
We have experimentally evaluated the heuristic methods we proposed for solving the model-based segmentation problem for multiple trajectories. In this chapter we outline these results while commenting on the advantages and disadvantages of the approaches. The visualization of resulting segmentation representations is done by drawing colored lines of varying width to represent segments. The color of these lines is used to indicate the movement model parameter fit to each particular segment. A simple Python implementation of the approaches was used to run experiments on trajectory data that was either generated or sampled from real data sets. We first present several experiments on synthetic data which serve to compare different approaches using different movement models and highlight differences in their resulting segmentations in Section 4.1. Additionally, we compare their actual running times in our implementation to obtain an estimate of their relative time performance. Secondly we run some of the approaches on real data sets, including a group of wild baboons [28] and a set of skiing trajectories obtained from a GPS tracking phone app. The resulting segmentations of these experiments we will compare visually and based on their IC score.

4.1 Comparing approaches with synthetic data

To get a clear overview of the effectiveness of each segmentation approach, we first segment smaller, synthetic trajectory data to quickly compare approaches. The trajectories are generated using a trajectory generation algorithm described by Technitis et al. [30]. Trajectories generated with this method normally move randomly from a fixed start point to a fixed end point, given a maximum velocity and a specified amount of locations. We vary the location of the start and end point around fixed points to generate multiple trajectories that together resemble group movement from one location to another. With these specifications we are able to set the number of trajectories m and amount of locations per trajectory n and generate a corresponding trajectory set $\mathcal{T}$.

We have designed two simple test cases that should test whether the approaches are able to recognize grouping changes and movement behavior changes. The first example set is generated in a "Y"-shape, indicating a change in the grouping structure around the point where the group splits up into two paths. This trajectory set, denoted as $\mathcal{T}_Y$ is shown in Figure 4.1(a). The second example set is generated with the idea that a "good" segmentation should be able to pick up movement changes within a group and segment the change appropriately. For this example we generated two trajectories τ_0, τ_1 which move straight over two parallel lines up until the halfway point in time, say $t_{\frac{1}{2}}$. From there, trajectory τ_0 keeps moving straight to its end point, while τ_1 also moves to its endpoint but with different, more erratic movement. This trajectory set which we denote as $\mathcal{T}_M$ is shown in Figure 4.1(b).

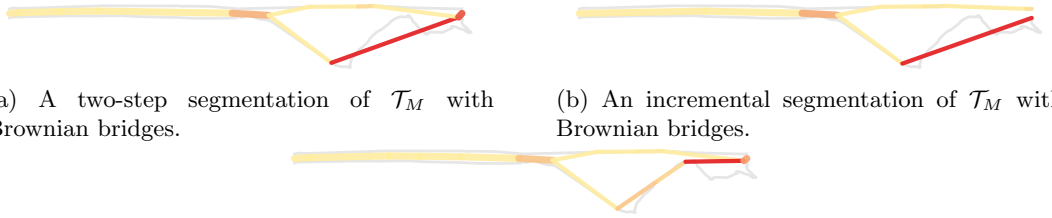
We use several combinations of segmentation approaches and movement models to segment sets $\mathcal{T}_Y$ and $\mathcal{T}_M$. We first focus on $\mathcal{T}_M$. For this set we do not use the centroid distances and k-nearest neighbor distances BCPA variables, as the centroid distances are equal for both traject-



(a) Trajectory set $\mathcal{T}_Y$ which was generated in a "Y"-shape.

(b) Trajectory set $\mathcal{T}_M = \{\tau_0, \tau_1\}$. The behavior of τ_1 changes at the halfway point.

Figure 4.1: Two sets of generated trajectories with the purpose of verifying whether our segmentation approaches are able to pick up grouping and movement changes.



(a) A two-step segmentation of $\mathcal{T}_M$ with Brownian bridges.

(b) An incremental segmentation of $\mathcal{T}_M$ with Brownian bridges.

(c) A clustering segmentation of $\mathcal{T}_M$ with Brownian bridges.

Figure 4.2: Segmentations of $\mathcal{T}_M$ using the BBMM in the three approaches. The diffusion coefficient σ_m^2 of each segment is visualized with a color in the range yellow to red. Yellow represents lower diffusion coefficient values and red represents higher diffusion coefficient values.

ories and the only possible k-nearest neighbor distance with $k = 1$ does not behave differently. The segmentations of $\mathcal{T}_M$ using BBMM are shown in Figure 4.2, the segmentations using BCPA with persistent velocities V_p are shown in Figure 4.3. From this point forward, the segments in segmentations will be visualized with a color indicating the model parameter value. For segmentations made with BBMM, yellow indicates a lower diffusion coefficient and red a higher diffusion coefficient. To easily distinguish between BBMM and BCPA segmentations, the segments for BCPA segmentations use a different color range; teal indicates a low autocorrelation coefficient, while dark-blue indicates a higher autocorrelation coefficient. We observe the trajectories in $\mathcal{T}_M$ are segmented as expected, being grouped on the first half and for the most part segmented further separately.

Now we consider trajectory set $\mathcal{T}_Y$. The segmentations made using approaches with Brownian bridges are shown in Figure 4.4. For each of the approaches, we see that the splitting of the trajectories is reflected in the segmentations. The part where this split happens is also described with higher diffusion coefficients, which indicates more erratic behavior. This clearly stems from the change in direction, as the trajectories move rather straight along their paths outside the split.

Next, we apply the approaches to $\mathcal{T}_Y$ using BCPA with different model variables. First we consider the persistent velocities V_p, which is the variable traditionally modeled with BCPA. The segmentations using this model are shown in Figure 4.5. All three approaches are able to pick up on the splitting of the group and segment accordingly, with varying degrees of complexity. The main difference we see in the clustering segmentation is the slightly increased complexity of the

(a) A two-step segmentation of $\mathcal{T}_M$ with BCPA modeling persistent velocities V_p.

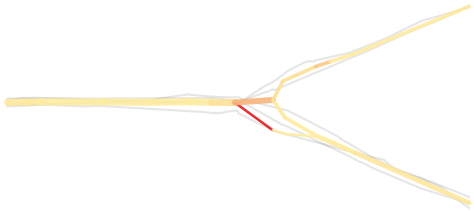
(b) An incremental segmentation of $\mathcal{T}_M$ with BCPA modeling persistent velocities V_p.

(c) A clustering segmentation of $\mathcal{T}_M$ with BCPA modeling persistent velocities V_p.

Figure 4.3: Segmentations of $\mathcal{T}_Y$ with BCPA modeling persistent velocities V_p. The autocorrelation coefficient ρ of each segment is visualized with a color in the range teal to dark-blue. Teal represents lower autocorrelation coefficient values and dark-blue represents higher autocorrelation coefficient values.

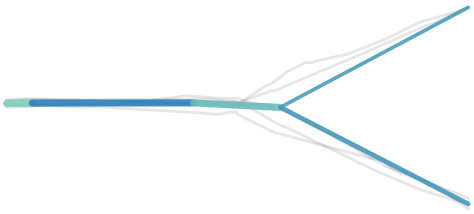
(a) A two-step segmentation of $\mathcal{T}_Y$ with Brownian bridges.

(b) An incremental segmentation of $\mathcal{T}_Y$ with Brownian bridges.

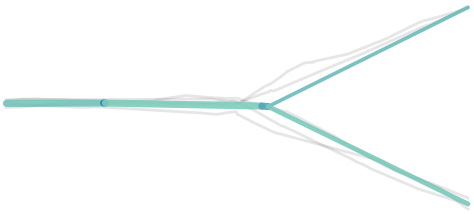


(c) A clustering segmentation of $\mathcal{T}_Y$ with Brownian bridges.

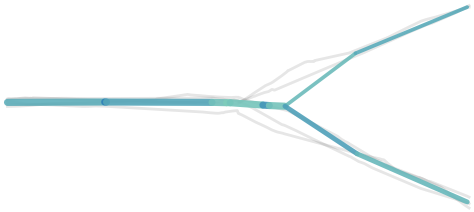
Figure 4.4: Segmentations of $\mathcal{T}_Y$ using the BBMM in the three approaches. The diffusion coefficient σ_m^2 of each segment is visualized with a color in the range yellow to red. Yellow represents lower diffusion coefficient values and red represents higher diffusion coefficient values.



(a) A two-step segmentation of $\mathcal{T}_Y$ with the BCPA model and persistent velocities V_p as model variable.



(b) An incremental segmentation of $\mathcal{T}_Y$ with the BCPA model and persistent velocities V_p as model variable.



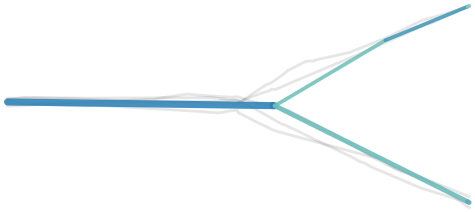
(c) A clustering segmentation of $\mathcal{T}_Y$ with the BCPA model and persistent velocities V_p as model variable.

Figure 4.5: Segmentations of $\mathcal{T}_Y$ using the BCPA model with the persistent velocities V_p as a model variable in each of the approaches. The autocorrelation coefficient ρ of each segment is visualized with a color in the range teal to dark blue, where teal represents a low autocorrelation coefficient and the dark blue represents a high autocorrelation coefficient.

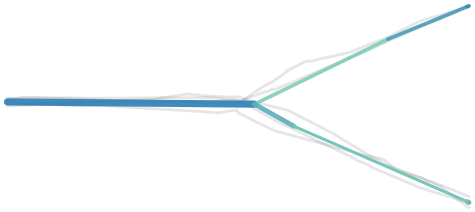
segmentation compared to the two-step and incremental approaches. Recall that the complexity parameters p_{size} and p_{length} used in the clustering approach do not directly translate to a complexity weight used in an (estimated) IC score for the constructed segmentation (see Equation 3.2). We set the complexity weight to the same value in all segmentations in Figure 4.5, but are given slightly more complexity by the clustering approach. Although this may not be too noticeable with this set of trajectories, it emphasizes that the scoring function really only approximates the complexity part of the IC.

The second BCPA model variable we consider is the distance to the centroids. The resulting segmentations are shown in Figure 4.6. In terms of grouping, all approaches are able to find the y-shaped grouping in the trajectory set, and the segment autocorrelation coefficients make sense for the model parameter. The other distance-based variable we used is the k-nearest neighbor distances, with $k = 2$ for this example. These segmentations can be found in Figure 4.7, and are able to distinguish the different grouping phases. The autocorrelation coefficients are once again acting as expected in this scenario with the 2-nearest neighbor BCPA model variable being used.

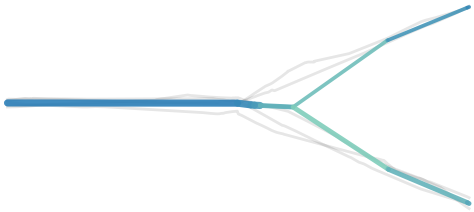
To test the efficiency of our algorithms, we ran the implementation of each approach on a generated data set of ten trajectories with 500 points, measuring the total computation time. These times are reported in Table 4.1. The difference in times between using Brownian bridges or using BCPA likely stems from the log-likelihood computation requiring knowledge of V_p values at a larger time interval, while the Brownian bridges directly compute likelihoods from locations. The clustering approach is noticeably slower, due to the large amount of candidate clusters that have to be constructed.



(a) A two-step segmentation of $\mathcal{T}_Y$ with the BCPA model and centroid distance as model variable.



(b) An incremental segmentation of $\mathcal{T}_Y$ with the BCPA model and centroid distance as model variable.

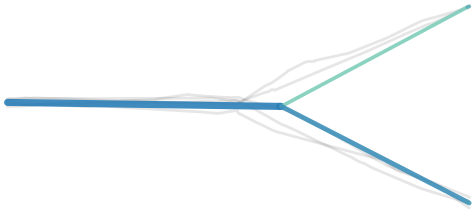


(c) A clustering segmentation of $\mathcal{T}_Y$ with the BCPA model and centroid distance as model variable.

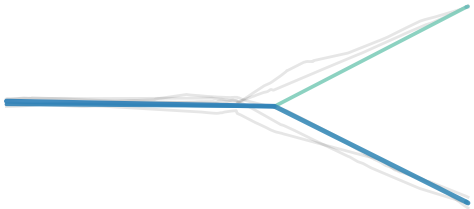
Figure 4.6: Segmentations of $\mathcal{T}_Y$ using the BCPA model with the distances to the centroids as a model variable in each of the approaches. The autocorrelation coefficient ρ of each segment is visualized with a color in the range teal to dark blue, where teal represents a low autocorrelation coefficient and the dark blue represents a high autocorrelation coefficient. The top-right segment in each segmentation has a lower autocorrelation coefficient. This can be explained by this segment only containing two subtrajectories, while the segment below contains three. This means that the subtrajectories in the top-right segment move further away from the centroid than the trajectories below, resulting in a lower autocorrelation coefficient.

Computation times				
Segmentation method	BBMM	BCPA (Centroid distances)	BCPA (k-NN)	BCPA (V_p)
Two-step	9.995s	32.708s	32.996s	157.142s
Incremental	13.822s	36.105s	35.919s	204.064s
Clustering	427.720s	199.578s	184.531s	1094.283s

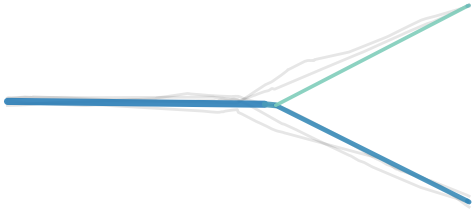
Table 4.1: The computation times of different combinations of segmentation algorithms and movement models. Each method was used to segment a trajectory set with $m = 10$ and $n = 500$. Running times are given in seconds.



(a) A two-step segmentation of $\mathcal{T}_Y$ with the BCPA model and 2-nearest neighbor distance as model variable.



(b) An incremental segmentation of $\mathcal{T}_Y$ with the BCPA model and 2-nearest neighbor distance as model variable.



(c) A clustering segmentation of $\mathcal{T}_Y$ with the BCPA model and 2-nearest neighbor distance as model variable.

Figure 4.7: Segmentations of $\mathcal{T}_Y$ using the BCPA model with the 2-nearest neighbor distance as a model variable in each of the approaches. The autocorrelation coefficient ρ of each segment is visualized with a color in the range teal to dark blue, where teal represents a low autocorrelation coefficient and the dark blue represents a high autocorrelation coefficient. We see the same visual effect as in the centroid distances segmentations, where the top-right segment has a lower autocorrelation coefficient than the bottom-right segment. For the two subtrajectories in the top-right segment the 2-nearest neighbor distances keep increasing, while the 2-nearest neighbor distances for the three bottom-right subtrajectories do not change much, resulting in a higher autocorrelation coefficient in the bottom-right.

4.2 Experiments on real data

To further evaluate our methods we ran several of the segmentation algorithms on two real data sets. First we segment a data set of multiple baboon trajectories, moving through a wildlife reserve over the span of a few days [17, 28]. Then we segment a data set consisting of geese migration flights [25]. To evaluate our approaches on these data sets, we compare their IC scores and explore the segmentations visually to find good and bad decisions made by the segmentation approaches.

4.2.1 Segmentation visualization

In this section we visualize segmentations similarly as in Section 4.1, where BBMM segments are drawn with a yellow-red color palette and BCPA segments in a teal-dark blue color palette, on top of transparent black lines following the actual trajectories. Lighter segment colors indicate lower parameter values for these models, and darker colors indicate higher parameter values. For segmentations using BCPA, the parameters (autocorrelation coefficient ρ) must lie in the range $[0, 1]$. The correct color for a parameter in that range can then be obtained by linearly interpolating between the palette colors. The diffusion coefficients in the BBMM can take any positive value, with larger values occurring when the scale of the trajectories is larger. If some x is the maximum diffusion coefficient in the segmentation we want to visualize, the color for any segment can be obtained with linear interpolation on the range $[0, x]$. In cases of parameter value outliers, this can result in a large gap between the colors in a segmentation, which we will see an example of later.

4.2.2 Segmenting Baboon trajectories

We attempt to make the visualization of the segments align more with the data by drawing arcs that follow the movement that a segment captures. The difference between visualizations with and without arcs is shown in Figure 4.8(a) and Figure 4.8(b), where we show an example segmentation on the baboon data set with and without arcs. Depending on how much a visualization is improved with arc segments we will decide whether to use arcs in the other visualizations in this section. Additionally, it is possible to draw the location models fitted to vertices as ellipses. This would make vertices where trajectory locations have a larger spread stand out among other vertices. We have implemented a simple version of this, but found that visualizing these ellipses causes too much clutter. Figure 4.8(c) shows an example segmentation with the location models visualized.

From the baboon data set, fifteen trajectories with the most consistent reports of locations over time were selected. For each of these trajectories we sampled a subtrajectory on a fixed time interval during the seventh day of tracking. The resulting trajectory set $\mathcal{T}_{\text{baboons}}$ contained $m = 15$ trajectories with $n = 1000$ points each. The reason for subsampling was twofold: it decreased computation time which helped in running multiple experiments quickly and reduced the amount of backtracking occurring in the resulting trajectories. We use the term backtracking to describe subtrajectories which move back along a previously traversed path. Segments that capture the entire back-and-forth movement in one segment will become harder to visualize as these segments will start and end at similar points, no longer visually representing the movement of the subtrajectories.

Figure 4.9 shows a two-step segmentation of the baboon data, using Brownian bridges. Additionally, in Figures 4.9(b) and (c) we have highlighted how two trajectories in the set are represented by the segmentation. Observe that most of the spikes are matched by the diffusion coefficients of the segments, except for one spike in (c). This segmentation seems to be rather complex, with segment parameters following optimal values very erratically, almost on a per-bridge level. This is a disadvantage of the two-step approach, since the grouping structure stays the same regardless of the complexity weight. To verify this, we increased the complexity penalty weight to try and obtain a segmentation that was less complex, but increasing this parameter to very high values did not result in changes in the segmentation output. In (b) and (c), note that the parts where

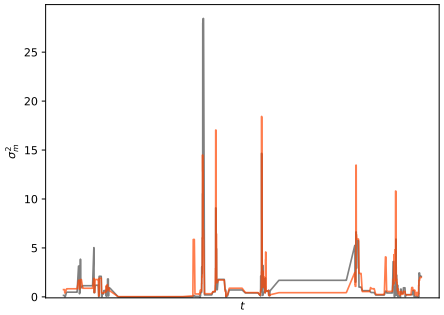
(a) An example segmentation of the baboon data set using BCPA as a movement model with straight lines representing segments. Note that the movement of the subtrajectories in the segments does not always match a straight line.

(b) The same segmentation as in (a) with arcs instead of straight lines representing segments. For the majority of the segments, these arcs follow the movement of the subtrajectories in each segment. However, a clear example of an arc not following this movement is the segment in the bottom left, where the movement is skipped by a straight line. These "faulty" arcs are caused by implementation issues we did not resolve.

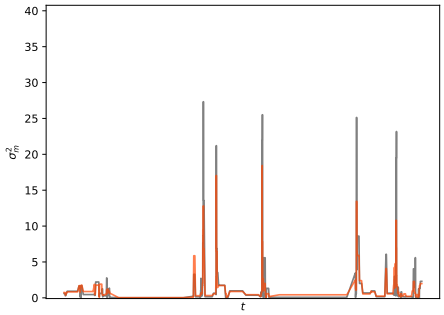
(c) The same segmentation as in (a) now with arcs and with location model ellipses drawn transparently. The ellipses are drawn based on the parameters used in the location models, σ_x, σ_y.

Figure 4.8: A comparison of different visualization methods for segmentations, based on a baboon data segmentation which used the BCPA model.

(a) The baboon data set segmented with the two-step approach using Brownian bridges.



(b) A graph comparing the segment diffusion coefficients (red line) and the optimal diffusion coefficients (black line) for trajectory τ_{10} in the baboon set. If these lines are close for some time interval, the segmentation represents the trajectory well on that interval. The optimal diffusion coefficients with relation to each segment or edge e is the diffusion coefficient that maximizes the likelihood for subtrajectory of τ_{10} over the time interval corresponding to e.



(c) Similar to (b), the optimal diffusion coefficients and the coefficients of the segments representing another one of the trajectories, τ_{12}, in the baboon set.

Figure 4.9: Two-step segmentation of the baboon data set using the BBMM.

Figure 4.10: The baboon data set segmented with the two-step approach using BCPA with persistent velocities V_p.

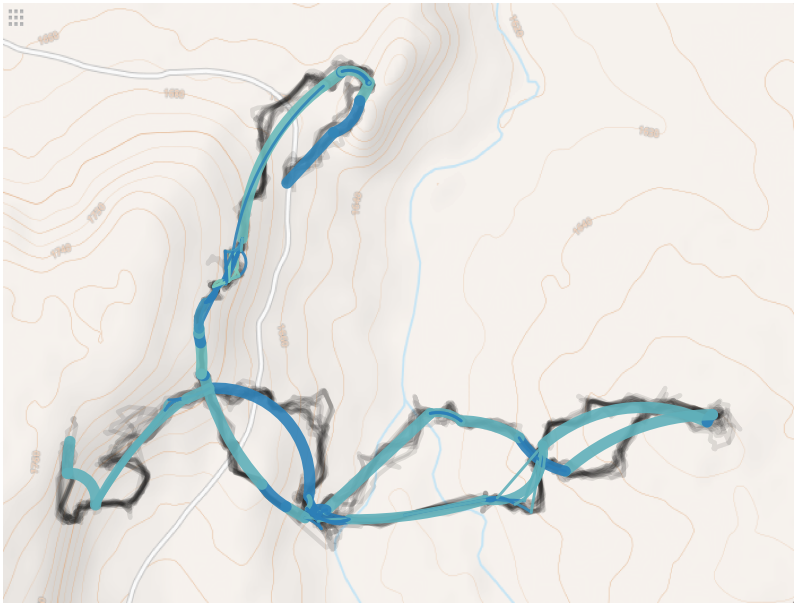


Figure 4.11: The baboon data set segmented with the incremental approach using BCPA with persistent velocities V_p.

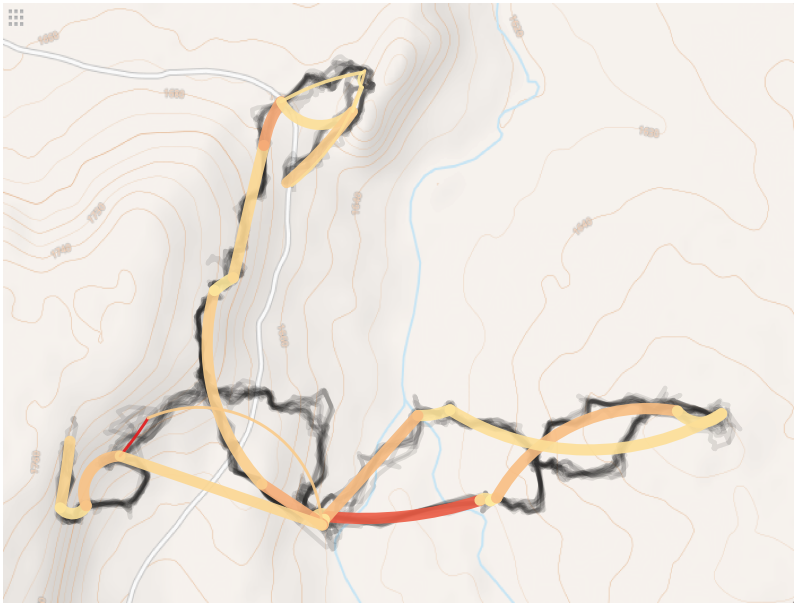


Figure 4.12: The baboon data set segmented with the incremental approach using Brownian bridges.

the optimal and segment coefficients stay close to zero for some time correspond to two large time shifts in the data, originating from temporal intervals in the tracking of the baboons.

In Figure 4.10 the two-step approach using the BCPA model with persistent velocities segments the baboon set. We can see the trajectories move somewhat erratically around the point where the river splits, also backtracking shortly at times. Due to this behavior the two-step approach has split the group in which the baboons crossed the river up into multiple subgroups. The autocorrelation coefficients in this segmentation are either very low or very high, which is an interesting result. We suspect that fitting the generalized BCPA model to groups that do not have similar movement (a consequence from the first step in the two-step algorithm) results in the high contrast of autocorrelation values. For example, the incremental approach using BCPA with persistent velocities shown in Figure 4.11 has more nuanced autocorrelation coefficients assigned to its segments.

We furthermore segmented the baboon data set with the incremental approach using Brownian bridges and BCPA with centroid distances. The segmentation using BBMM is shown in Figure 4.12. The diffusion coefficients seem to align with the behavior that is seen in the trajectories. For example, the small segment with high diffusion coefficient on the left falls in line with a small group of baboons which move in the other direction from the group, erratically moving around a small area, then returning to the main group. The trajectories underneath the large red segment at the bottom look somewhat straight, but since there is a skip in time between GPS observations here the high diffusion coefficient instead models the clutter on the left of the segment.

We will now look at segmentations using distance-based BCPA variables. Since the two-step approach already groups the trajectories, we do not expect modeling distance-based variables in the second step to give meaningful results. We therefore use the incremental approach to evaluate the use of these variables. In Figure 4.13 we see an incremental segmentation of the baboon set with centroid distances used in the BCPA model. Figure 4.14 shows an incremental segmentation using BCPA with 4-nearest neighbor distances. While the autocorrelation coefficients of the segments are harder to interpret the results with, the behavior in relation to the rest of the group should be captured for each trajectory. From the two segmentations we can see that the centroid distances and 4-nearest neighbor distances act similarly, since the autocorrelation coefficients of the two segmentations are similar for the same parts in the trajectory set.

Figure 4.13: The baboon data set segmented with the incremental approach using BCPA with centroid distances.

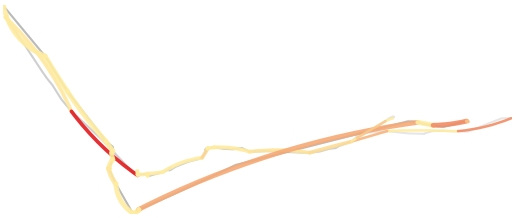
Figure 4.14: The baboon data set segmented with the incremental approach using BCPA with 4-nearest neighbor distances.

Figure 4.15: A set of $m = 5$ geese migration flight trajectories, with $n = 1880$ points each. While there are five geese in this set, we can clearly see two groups for the majority of the trajectory data. Additionally, the small black spots on the inside of the corner have been verified to be stop-overs where the geese rest or feed. These stop-overs would ideally be picked up on by our segmentation methods.

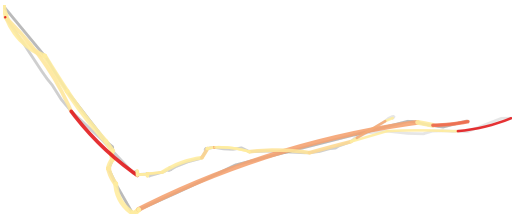
4.2.3 Segmenting geese trajectories

From the geese data, we obtained the longest trajectory set for which all points were time-aligned between trajectories. This set can be seen in Figure 4.15. The segmentations using the BBMM can be seen in Figure 4.16. In all three approaches, the stop-overs in the geese trajectories are not recognized as different segments. These stop-overs contain points that are very close together already, as the geese almost do not move compared to the larger flights they take in between. The diffusion coefficients in the Brownian bridges that fit some movement well scale with the length of the bridges themselves. For example, a large portion of flight that deviates only slightly will have a larger diffusion coefficient maximizing the log-likelihood than a very small section of movement as seen in the stop-overs, even if the movement is erratic on this smaller scale. While the movement of the baboons constantly stayed on the same "scale", in the geese data there is a stark contrast between movement lengths. The Brownian bridges and BBMM as a movement model are not that well suited to this type of data. The segmentation approaches are able to group the trajectories in the two groups that the geese clearly fly in.

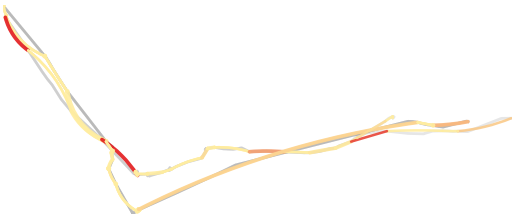
The segmentations using BCPA with different variables can be seen in Figures 4.17. Modeling persistent velocities with BCPA seems to have a positive effect on the segmentation of the stop-overs, except for the incremental approach, for which we obtained a less complex segmentation. The BCPA modeling distance-based variables was not able to capture the stop-overs at all. Segmentations with centroid distances are shown in Figure 4.18 and segmentations with 2-nearest neighbor distances are shown in Figure 4.19. Since the distances to the centroids or 2-nearest neighbors do not change much during a stop-over due to there being almost no movement, the segmentations fit larger segments to the data covering these stop-overs. The usefulness of distance-based BCPA variables most likely depends on the purpose of the segmentation or the context of the data. When trying to capture behavioral differences (regardless of grouping, as we are interested



(a) The geese data segmented with the two-step approach using the BBMM.

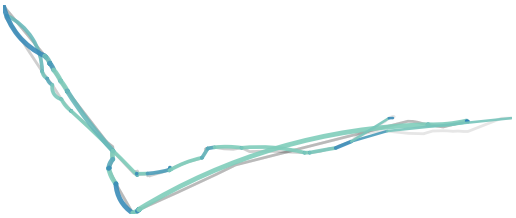


(b) The geese data segmented with the two-step approach using the BBMM.

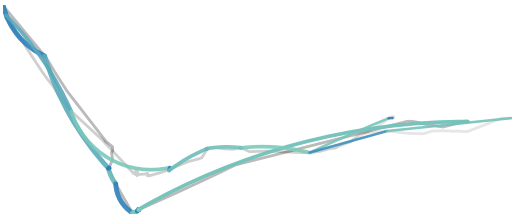


(c) The geese data segmented with the two-step approach using the BBMM.

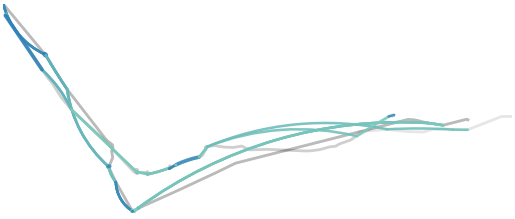
Figure 4.16: Segmentations of the geese data using the BBMM as a movement model. Note that the stop-overs are not segmented differently.



(a) The geese data segmented with the two-step approach using BCPA with persistent velocities V_p.

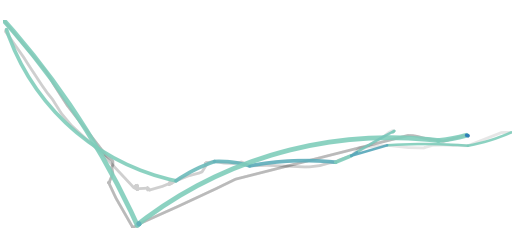


(b) The geese data segmented with the incremental approach using BCPA with persistent velocities V_p.

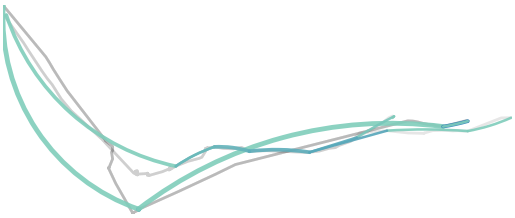


(c) The geese data segmented with the clustering approach using BCPA with persistent velocities V_p.

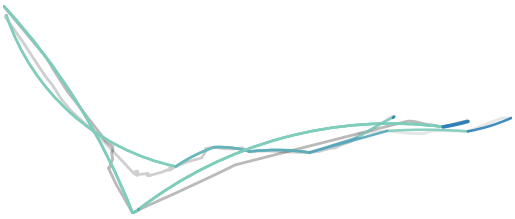
Figure 4.17: Segmentations of the geese data using BCPA with persistent velocities V_p. The stop-overs are somewhat captured by the two-step and clustering approach, while the incremental approach does not segment most of them.



(a) The geese data segmented with the two-step approach using BCPA with centroid distances.



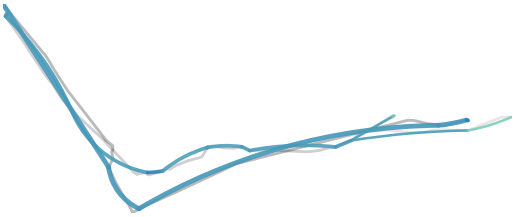
(b) The geese data segmented with the incremental approach using BCPA with centroid distances.



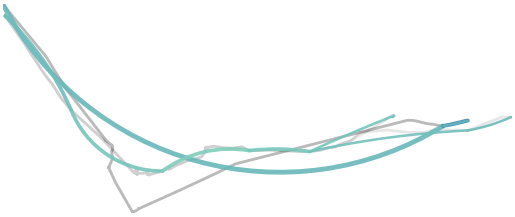
(c) The geese data segmented with the clustering approach using BCPA with centroid distances.

Figure 4.18: Segmentations of the geese data using BCPA with centroid distances. The stop-overs are not captured by these segmentations.

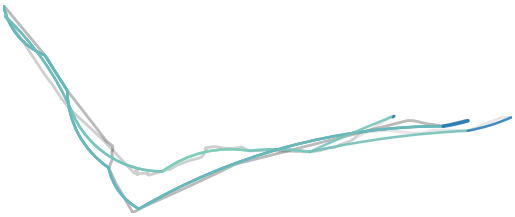
specifically in the stop-overs here) these distance-based variables are not helpful.



(a) The geese data segmented with the two-step approach using BCPA with 2-nearest neighbor distances.



(b) The geese data segmented with the incremental approach using BCPA with 2-nearest neighbor distances.



(c) The geese data segmented with the clustering approach using BCPA with 2-nearest neighbor distances.

Figure 4.19: Segmentations of the geese data using BCPA with 2-nearest neighbor distances. The stop-overs are not captured by these segmentations.

Chapter 5

Discussion & future work

In this thesis we have introduced a model-based representation for the segmentation of a group or a set of trajectories. This representation captures both how subtrajectories group together as well as characteristics of the movement and when these change. As such it generalizes the segmentation of individual trajectories, but also adds a layer of complexity since subgroups need to be identified.

We also propose an approach to evaluate these segmentations using an information criterion that integrates both how well a movement model fits to a group, and to what extent the point at which the model changes can be mapped to a location (model). The penalty of the information criterion is the complexity of the representation. Together, the segmentation representation and the information criterion cleared the way for us to define the algorithmic problem of segmenting trajectory groups using models formally, while abstracting from the choice of movement or location models. This allows us to use the algorithms developed for this problem with a movement model of choice, and we demonstrate how to use group segmentations with the Brownian bridge movement model and Behavioral Change Point Analysis.

Unfortunately, the algorithmic problem most likely cannot be solved optimally by an efficient algorithm, since even just clustering points, which occurs as a subproblem, is NP-hard. The IC score of a segmentation is defined both by its grouping aspects (location models) and log-likelihoods of movement phases (movement models), which leaves a large number of possible segments or segmentation options. We therefore proposed three heuristic methods to solve the segmentation problem; a two-step, an incremental and a clustering approach. They explore the large space of different segmentations by considering smaller divisions, based on grouping (two-step method) or movement and grouping (incremental, clustering).

We evaluated these heuristics for the movement models mentioned above on various synthetic, animal movement and skiing data sets. While the two-step approach is able to consistently group trajectories based on a closeness threshold, this takes away from grouping happening on a movement basis. In other words, where the incremental and clustering approaches can choose to group subtrajectories because they moved similarly, the two-step approach does not have this choice, and groups subtrajectories solely on distance. This does show in the information criterion scores, which often were higher for the two-step approach than for the other approaches. In cases where the closeness threshold is known for the given trajectory set, the two-step approach is a useful method for segmenting the data. Overall, the two-step approach gave segmentations of the real data sets that were visually interesting and intuitive.

The incremental and clustering approaches require significant amounts of parameter tuning. While the incremental approach works by directly trying to increase the IC score of its segmentation, we have found unexpected segmentation behavior due to the way we interpret complexity in our models, which we tried to counteract by slightly modifying the IC computation in the approach. The clustering approach could be improved by further studying the relevance of potential clusters, which we describe using a scoring function based on their size and length. We have observed that the scoring parameters – which should have some indirect relation to the complexity penalty weight – are difficult to tune accordingly to the data. We propose that having a better

understanding of what clusters to select and how to quantify their relevance, can help improve the quality of the clustering segmentations further. Currently, an important limitation for the clustering approach is its efficiency, which in particular limited the size of the data sets it could be used for.

In the model-fitting we made the simplifying assumption that we could add up the log-likelihoods. However, this corresponds to multiplying likelihoods, which would only be correct when the movements are independent. Arguably this is not a problem when determining the model parameter for a given segment, assuming that a parameter that is suitable for the individual trajectories is also suitable for the group. However, it may cause a problem when deciding whether to group trajectories, since no matter whether subtrajectories are grouped or not, their likelihoods are added up. A similar problem arises when combining movement and location models, since we again simply sum the log-likelihoods, while they are not necessarily independent. We expect that better models –and with this a better information criterion– for group segmentation may considerably improve the results of our algorithms.

We focused on the Brownian bridge movement model and an optimization version of Behavioral Change Point Analysis (BCPA). BCPA can use different movement characteristics, and in many cases the persistent velocity is used as a characteristic. Using group-specific characteristics like distance to the centroid or to some neighbor gave interesting results, which warrant further investigation.

So far, we use a very simple visualization, essentially just drawing the straight-line graph corresponding to the segmentation. However, since the locality of segments is only tied to its start and end points, the in-between movement of its subtrajectories is often not represented by a straight line from the start to the end point. Drawing arcs that best fit the in-between subtrajectories addresses this issue to a certain degree, but not in a robust way. Furthermore, if entities retrace their previous movement in their trajectories, segments might start and end on very close points while their subtrajectories move towards a far away area and back. Furthermore, when segments are very long, we again end up with an abstract representation of the movement. It is an interesting open problem how to visualize the segmentation of a group in a way that it is easy to interpret and extract useful information from segmentations.

Bibliography

[1] Pankaj K. Agarwal, Kyle Fox, Kamesh Munagala, Abhinandan Nath, Jiangwei Pan, and Erin Taylor. Subtrajectory clustering: Models and algorithms. In *Proc. 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, SIGMOD/PODS '18, page 75–87, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3196959.3196972. 2, 4, 19, 22, 23, 25

[2] Sander P.A. Alewijnse, Kevin Buchin, Maike Buchin, Andrea Kölzsch, Helmut Kruckenberg, and Michel A. Westenberg. A framework for trajectory segmentation by stable criteria. In *Proc. 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 351–360. ACM, 11 2014. doi:10.1145/2666310.2666415. 2, 7

[3] Sander P.A. Alewijnse, Kevin Buchin, Maike Buchin, Stef Sijben, and Michel A. Westenberg. Model-based segmentation and classification of trajectories. *Algorithmica*, 80(8):2422–2452, 2018. doi:10.1007/s00453-017-0329-x. 4, 7, 10, 11, 19, 22

[4] Aris Anagnostopoulos, Michail Vlachos, Marios Hadjieleftheriou, Eamonn Keogh, and Philip S. Yu. Global distance-based segmentation of trajectories. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 34–43, New York, NY, USA, 2006. Association for Computing Machinery. doi:10.1145/1150402.1150411. 3

[5] Boris Aronov, Anne Driemel, Marc van Kreveld, Maarten Löffler, and Frank Staals. Segmentation of trajectories on nonmonotone criteria. *ACM Trans. Algorithms*, 12(2), December 2015. doi:10.1145/2660772. 3, 7

[6] Milutin Brankovic, Kevin Buchin, Koen Klaren, André Nusser, Aleksandr Popov, and Sampson Wong. (k, l)-medians clustering of trajectories using continuous dynamic time warping. In *Proc. 28th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 99–110. ACM, 2020. doi:10.1145/3397536.3422245. 2

[7] Kevin Buchin, Maike Buchin, David Duran, Brittany Terese Fasy, Roel Jacobs, Vera Sacristan, Rodrigo I. Silveira, Frank Staals, and Carola Wenk. Clustering trajectories for map construction. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '17, pages 14:1–14:10, New York, NY, USA, 2017. Association for Computing Machinery (ACM). doi:10.1145/3139958.3139964. 2

[8] Kevin Buchin, Maike Buchin, and Joachim Gudmundsson. Constrained free space diagrams: A tool for trajectory analysis. *International Journal of Geographical Information Science*, 24(7):1101–1125, 2010. doi:10.1080/13658810903569598. 2

[9] Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Jorren Hendriks, Erfan Hosseini Sereshgi, Vera Sacristán, Rodrigo I. Silveira, Jorrick Sleijster, Frank Staals, and Carola Wenk. Improved map construction using subtrajectory clustering. In *Proceedings of the 4th ACM SIGSPATIAL Workshop on Location-Based Recommendations, Geosocial Networks,*

and Geoadvertising, LocalRec'20, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3423334.3431451. 2

[10] Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Michael Horton, and Stef Sijben. Compact flow diagrams for state sequences. *Journal on Experimental Algorithmics*, 22(1), 2017. doi:10.1145/3150525. 2, 3, 4, 6

[11] Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Maarten Löffler, and Jun Luo. Detecting commuting patterns by clustering subtrajectories. In Seok-Hee Hong, Hiroshi Nagamochi, and Takuro Fukunaga, editors, *Algorithms and Computation*, pages 644–655, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. 2

[12] Kevin Buchin, Maike Buchin, Marc van Kreveld, Bettina Speckmann, and Frank Staals. Trajectory grouping structure. *Journal of Computational Geometry*, 6(1):75–98, 2015. doi:10.20382/jocg.v6i1a3. 2, 4, 7, 8, 19

[13] Kevin Buchin, Anne Driemel, Joachim Gudmundsson, Michael Horton, Irina Kostitsyna, Maarten Löffler, and Martijn Struijs. Approximating (k,l)-center clustering for curves. In *Proc. 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2922–2938. Society for Industrial and Applied Mathematics (SIAM), 2019. doi:10.1137/1.9781611975482.181. 2

[14] Kevin Buchin, Anne Driemel, Natasja van de L'Isle, and André Nusser. klcluster: Center-based clustering of trajectories. In *Proc. 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL*, pages 496–499. ACM, 2019. doi:10.1145/3347146.3359111. 2

[15] Maike Buchin, Anne Driemel, Marc van Kreveld, and Vera Sacristán. Segmenting trajectories: a framework and algorithms using spatiotemporal criteria. *Journal of Spatial Information Science*, 3:33–63, 2011. doi:10.5311/JOSIS.2011.3.66. 3, 7

[16] Maike Buchin, Bernhard Kilgus, and Andrea Kölzsch. Group diagrams for representing trajectories. *International Journal of Geographical Information Science*, 34(12):2401–2433, 2020. doi:10.1080/13658816.2019.1684498. 2, 4, 6

[17] MC Crofoot, RW Kays, and M Wikelski. Data from: Shared decision-making drives collective movement in wild baboons, 2015. URL: http://dx.doi.org/10.5441/001/1.kn0816jn, doi:doi:10.5441/001/1.kn0816jn. 35

[18] Thomas Devogele, Laurent Etienne, Maxence Esnault, and Florian Lardy. Optimized discrete fréchet distance between trajectories. In *Proceedings of the 6th ACM SIGSPATIAL Workshop on Analytics for Big Geospatial Data*, BigSpatial'17, page 11–19, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3150919.3150924. 2

[19] Anne Driemel, Amer Krivosija, and Christian Sohler. Clustering time series under the fréchet distance. In *Proc. 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 766–785. SIAM, 2016. doi:10.1137/1.9781611974331.ch55. 2

[20] 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, 09 2016. doi:10.1186/s40462-016-0086-5. 4

[21] 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. doi:10.1111/j.1461-0248.2009.01293.x. 3, 11, 14, 15

[22] Eliezer Gurarie, Chloe Bracis, Maria Delgado, Trevor D. Meckley, Ilpo Kojola, and C. Michael Wagner. What is the animal doing? tools for exploring behavioural structure in animal movements. *Journal of Animal Ecology*, 85(1):69–84, 2016. doi:https://doi.org/10.1111/1365-2656.12379. 4

[23] Jens Haase and Ulf Brefeld. Finding similar movements in positional data streams. In Davis Jesse, Jan Van Haaren, and Albrecht Zimmermann, editors, *Machine Learning and Data Mining for Sports Analytics - MLSA 2013*, number 1969 in CEUR Workshop Proceedings, pages 49–57, Germany, 2013. Sun Site Central Europe (RWTH Aachen University). URL: http://www.ecmlpkdd2013.org/. 2

[24] Jon S. Horne, Edward O. Garton, Stephen M. Krone, and Jesse S. Lewis. Analyzing animal movements using Brownian bridges. *Ecology*, 88(9):2354–2363, 2007. doi:10.1890/06-0957.1. 3, 12, 13

[25] A Kölzsch, GJDM Müskens, P Glazov, H Kruckenberg, and M Wikelski. Data from: Goose parents lead migration v, 2020. URL: http://dx.doi.org/10.5441/001/1.ms87s2m6, doi:doi:10.5441/001/1.ms87s2m6. 35

[26] Christine Parent, Stefano Spaccapietra, Chiara Renso, Gennady Andrienko, Natalia Andrienko, Vania Bogorny, Maria Luisa Damiani, Aris Gkoulalas-Divanis, Jose Macedo, Nikos Pelekis, Yannis Theodoridis, and Zhixian Yan. Semantic trajectories modeling and analysis. *ACM Comput. Surv.*, 45(4), August 2013. doi:10.1145/2501654.2501656. 3

[27] Sam Rycken, Kristin S. Warren, Lian Yeap, Bethany Jackson, Karen Riley, Manda Page, Rick Dawson, Karen Smith, Peter R. Mawson, and Jill M. Shephard. Assessing integration of black cockatoos using behavioral change point analysis. *The Journal of Wildlife Management*, 83(2):334–342, 2019. doi:https://doi.org/10.1002/jwmg.21609. 3

[28] Ariana Strandburg-Peshkin, Damien R. Farine, Iain D. Couzin, and Margaret C. Crofoot. Shared decision-making drives collective movement in wild baboons. *Science*, 348(6241):1358–1361, 2015. doi:10.1126/science.aaa5099. 29, 35

[29] Toru Tamaki, Daisuke Ogawa, Bisser Raytchev, and Kazufumi Kaneda. Semantic segmentation of trajectories with improved agent models for pedestrian behavior analysis. *Adv. Robotics*, 33(3-4):153–168, 2019. doi:10.1080/01691864.2018.1554508. 3

[30] Georgios Technitis, Walied Othman, Kamran Safi, and Robert Weibel. From a to b, randomly: a point-to-point random trajectory generator for animal movement. *International Journal of Geographical Information Science*, 29(6):912–934, 2015. doi:10.1080/13658816.2014.999682. 29

[31] Maryam Teimouri, Ulf Geir Indahl, Hanne Sickel, and Håvard Tveite. Deriving animal movement behaviors using movement parameters extracted from location data. *ISPRS International Journal of Geo-Information*, 7(2), 2018. doi:10.3390/ijgi7020078. 3

[32] Kevin Toohey and Matt Duckham. Trajectory similarity measures. *SIGSPATIAL Special*, 7(1):43–50, May 2015. doi:10.1145/2782759.2782767. 2

[33] Michalis Vlachos, George Kollios, and Dimitrios Gunopulos. Discovering similar multidimensional trajectories. In *Proc. 18th International conference on data engineering*, pages 673–684, 02 2002. doi:10.1109/ICDE.2002.994784. 2