UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

GERSON EDUARDO GROTH

Attribute Field K-Means: Clustering Trajectories with Attributes by Fitting Multiple Fields

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof. Dr. João Luiz Dihl Comba
Coadvisor: Prof. Dr. Carlos Scheidegger

Porto Alegre
August 2016

CIP — CATALOGING-IN-PUBLICATION

Groth, Gerson Eduardo

Attribute Field K-Means: Clustering Trajectories with Attributes by Fitting Multiple Fields / Gerson Eduardo Groth. – Porto Alegre: PPGC da UFRGS, 2016.

77 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2016. Advisor: João Luiz Dihl Comba; Coadvisor: Carlos Scheidegger.

1. Visualization. 2. Time Series. 3. Clustering. 4. Trajectory Attributes. 5. Interative Exploration. I. Comba, João Luiz Dihl. II. Scheidegger, Carlos. III. Título.

“ The greatest value of a picture is when it forces us to notice what we never expected to see. ”

— JOHN TUKEY

ABSTRACT

The amount of high-dimensional trajectory data and its increasing complexity imposes a challenge when visualizing and analysing this information. Trajectory Visualization must deal with changes both in space and time dimensions, but the attributes of each trajectory may provide insights about its behavior and important aspects. Thus, they should not be neglected. In this work, we tackle this problem by interpreting multivariate time series as attribute-rich trajectories in a configuration space that encodes an explicit relationship among the time series variables. We propose a novel trajectory-clustering technique called Attribute Field k-means (AFKM). It uses a dynamic configuration space to generate clusters based on attributes and parameters set by the user. Furthermore, by incorporating a sketching-based interface, our approach is capable of finding clusters that approximates the input sketches. In addiction, we developed a prototype to explore the trajectories and clusters generated by AFKM in an interactive manner. Our results on synthetic and real time series datasets prove the efficiency and visualization power of our approach.

Keywords: Visualization. Time Series. Clustering. Trajectory Attributes. Interative Exploration.

Attribute Field K-Means: Clusterizando Trajetórias com Attributos Através de Ajustes em Múltiplos Campos

RESUMO

A enorme quantidade de trajetórias contendo múltiplas dimensões, e o aumento de complexidade que esses dados ocasionam, impõem desafios ao visualizar e analisar estas informações. Visualização de Trajetórias deve lidar com alterações tanto na dimensão de espaço quanto na dimensão de tempo. Porém, os atributos de cada trajetória podem ocasionar uma melhor compreensão sobre seus comportamentos e características. Dessa forma, eles não deveriam ser neglicenciados. Neste trabalho, nós abordamos este problema interpretando séries temporais multivariadas com foco nos atributos das trajetórias, em um espaço de configuração que codifica um explícito relacionamento entre as variáveis das séries temporais. Nós propomos uma técnica original de clusterização de trajetórias, chamada *Attribute Field k-means* (AFKM). Ela usa um espaço de configuração dinâmica para gerar *clusters* baseados nos atributos e parâmetros definidos pelo usuário. Além disso, incorporando uma interface de *sketching*, nosso método é capaz de encontrar *clusters* que aproximam os exemplos de trajetórias desenhados pelo usuário. Nós também desenvolvemos um protótipo para explorar as trajetórias e *clusters* gerados pelo AFKM, de um modo interativo. Nossos resultados, em sintéticos e reais conjuntos de dados de séries temporais, provam a eficiência e o poder de visualização do nosso método.

Palavras-chave: Visualização. Séries Temporais. Clusterização. Attributos de Trajetórias. Exploração Interativa.

LIST OF ABBREVIATIONS AND ACRONYMS

AFKM	Attribute Field K-Means
GPS	Global Positioning System
HR	Heart Rate
MHR	Maximum Heart Rate
PAA	Piecewise Aggregate Reduction
SQL	Structured Query Language
TCX	Training Center XML
VFKM	Vector Field K-Means
XML	Extensible Markup Language
WebGL	Web Graphics Library

LIST OF FIGURES

CONTENTS

1 INTRODUCTION

As computers have become cheap, powerful, and small, so have the sensors they include. Many people now carry (in their phones) or wear (in their watches) full-fledged GPS units, and a variety of other sensors with them: barometers, heart-rate monitors, stride counters, etc. This provides lots of information, data that can be used for self-understanding is now commonplace and, as a result, we have easy access to large amounts of data on our location in space and time, together with health measurements. How do we enable users to make sense of all of this?

Time series are ubiquitous in data analysis, and so there is extensive literature on them, which includes summarization (PETITJEAN; GANÇARSKI, 2012), clustering (LIAO, 2005), and event detection (JANETZKO et al., 2014). The analysis becomes more exciting and challenging when the time series are multivariate, i.e. when each variable captures distinctive attributes of the same phenomena. The simplest approach for multidimensional data is to evaluate each attribute in the time series independently, although additional insight can be obtained by the analysis of the complex interplay of the different variables.

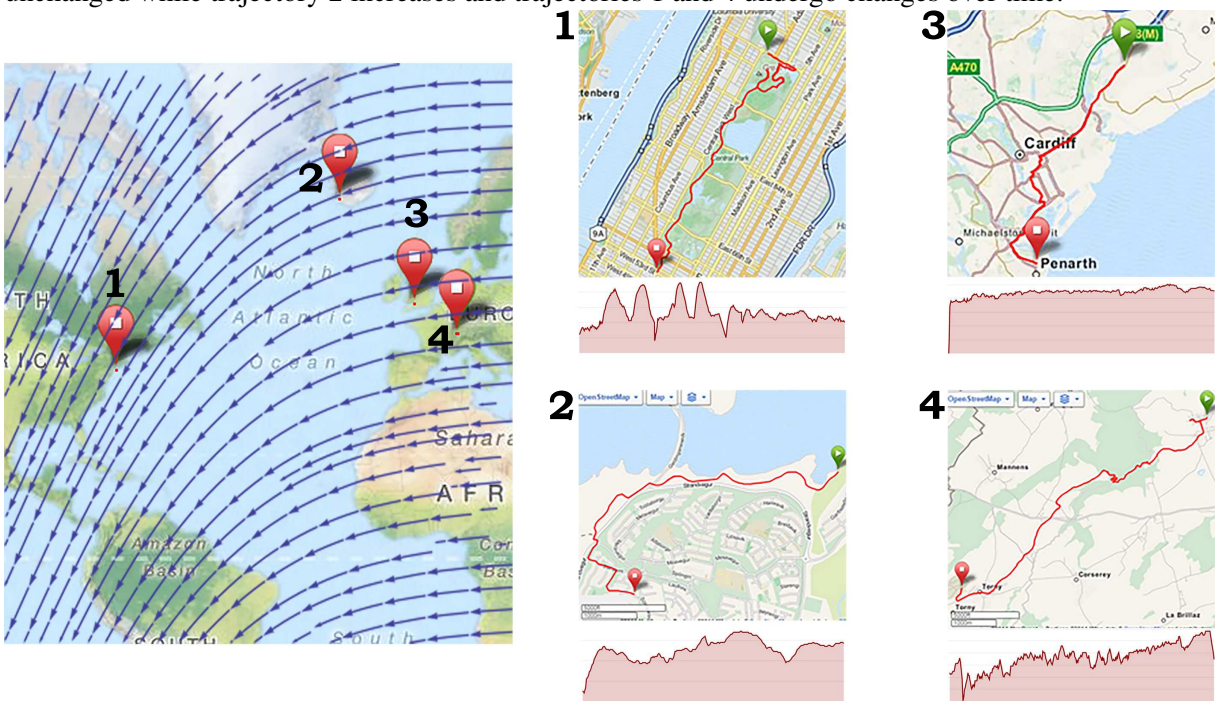
For example, Figure 1.1 presents four multivariate time series from an exercise dataset (more details in Section 6.3). Those trajectories remain in the same cluster when considering their trajectory vector field. However, analyzing other features such as pace, elevation, heart rate, time, and even the geographic coordinates, the trajectories differ meaningfully from each other. Those attributes are important and should be taken into account by the trajectory clustering.

The focus of this thesis is on spatiotemporal data, i.e., a type of time series data that contains, at least, a time stamp (temporal data), a spatial location (often a point described by latitude and longitude), and one object (describing what the data is). Nevertheless, most of the spatiotemporal data contains various objects existing and occurring in space, such as multidimensional attributes changing over time (ANDRIENKO et al., 2011). We will use the word *trajectory* to mean a time series of attributes that includes, among other attributes, location.

The complexity of such data makes trajectory visualization and analysis a challenging problem. But although complicated, it enables a variety of possible analysis tasks. For instance, we could be interested on monitoring behaviors of objects changing over time, or trying to infer some correlation between some of the recorded data. Several research fields have been making efforts in analyzing and developing techniques and tools for trajectory data: visualization, visual analytics, data mining, geographic information science, remote sensing.

In this thesis, we present a technique for clustering trajectories, specifically those tra-

Figure 1.1 – Figure illustrating the importance of using attributes for querying. 1, 2, 3, and 4 are all similar trajectories and belong to the same trajectory cluster (using a trajectory clustering that considers the path). However, their attributes pace, elevation, heart rate, time, and the geographic positions differ meaningfully from each other. Considering only heart rate (red), trajectory 3 remains practically unchanged while trajectory 2 increases and trajectories 1 and 4 undergo changes over time.



Source: (GARMIN, 2015).

jectories containing additional attributes of interest. Our technique is a novel interpretation and generalization of Ferreira et al.'s vector-field *k*-means (VFKM), a recently published algorithm for pure trajectory clustering (FERREIRA et al., 2013a). We show how to recast VFKM in terms of partial observations of any attribute in any field, providing a fast, robust algorithm for clustering attribute-rich trajectories. Analogous to VFKM, we call our technique attribute-field *k*-means (AFKM).

The analysis of multiple time series is fundamentally complicated because of alignment. Generally, time series overlap only partially, and any meaningful comparisons must account for this problem. When dealing with spatial series, this problem is compounded: instead of aligning segments, trajectory analysis must align *entire curves*. As we show in detail in Chapter 4, AFKM handles the alignment problem elegantly by letting two trajectories belong to the same cluster whenever they both could plausibly be *partial, noisy, observations of the same attribute field*. We then assume that the entire trajectory dataset can be described by a small number of attribute fields, and find which trajectories belong to which field. This process gives both a clustering and a simplified description of their behavior.

Even though the attribute field created by AFKM handles all attributes of the data, visual

exploration of trajectories is much more intuitive when done in a 2D interface. We designed such interface that displays each attribute separately over time, suggesting interesting choices of attributes for clustering. Besides, we allow the user defines additional filtering queries, enabling the data to be pruned as desired. The analysis of the data can proceed in two different ways: applied separately or in sequence. In a first way, the AFKM algorithm can be utilized to generate clusters based on parameters informed by the user. In a second approach, a sketching-based interface can be used to sketch a given trajectory, and the AFKM algorithm automatically finds clusters that better approximate the input sketch.

AFKM is quite general. Nevertheless, in this thesis, our experimental focus is on the analysis and visualization of exercise data gathered from a public website of a large company. In Chapter 6, we show how AFKM can be used to interactively cluster and visualize datasets containing each hundreds of workouts (and tens of thousands of attribute samples each) in major world cities such as New York, London, Boston, San Francisco and Salt Lake City. These experiments highlight both strengths and weaknesses of our technique, which are discussed in Chapter 7 and Section 7.6.

The **contributions** of this thesis include:

- AFKM, a novel trajectory clustering algorithm that handles attribute-rich trajectories,
- a theoretical and experimental evaluation of AFKM,
- a visual analytics tool for performing an analysis of trajectory data,
- a sketching-based interface for a guided clustering, and
- a case study involving clustering of trajectories with attributes, including a crowd-sourced dataset of exercise data.

1.1 Dissertation Structure

The dissertation is organized as follows. In Chapter 2, we provide a background for trajectories and k-means clustering. Chapter 3 introduces related works, presenting papers in Visualization, Clustering, Interactive Exploration of Trajectory Data, and Sketching-Based Interfaces. Next, in Chapter 4, we formulate AFKM and describe it. Chapter 5 describes the prototype implementation. Then, we validate our proposal with synthetic and real datasets in Chapter 6. Finally, in Chapter 7 we discuss strengths and weaknesses of our technique before concluding in Chapter 8.

2 BACKGROUND

In this Chapter, we summarize essential concepts to the understanding of our technique. We split this Chapter into two Sections: K-means method, and spatiotemporal data (more specifically, trajectories).

2.1 K-means Technique

K-means clustering (MACQUEEN et al., 1967) is a well-known clustering algorithm. It is a simple and easy method to classify the data into k clusters defined a priori. The central idea is the minimization of an objective function, which is usually the total distance (Euclidean) between all patterns and their respective clusters centers:

$$E = \sum_{j=1}^{k} \sum_{i=1} \| x_i^{(j)} - c_j \|^2, \tag{2.1}$$

where $x_i^{(j)}$ is the i^{th} pattern belonging to the j^{th} cluster and c_j is the centroid of the j^{th} cluster.

The algorithm relies on an iterative scheme, which can be summarized by the following steps (also represented in Figure 2.1):

1) Pick k elements representing the clusters centroids. Usually, the k elements are randomly chosen.

2) Assign each of the remaining objects to the cluster with the closest centroid, determined by the use of a distance function.

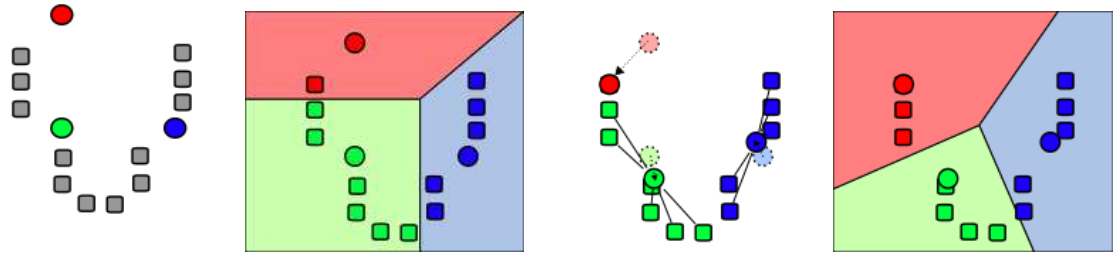
3) After assigning all objects, recalculate the position of the k centroids. The new centroid is the mean of all objects belonging the cluster.

4) Repeat Steps 2 and 3 until convergence, i.e., the centroids no longer move. Often is defined a number maximum of iteration, in cases where a precision error may interfere.

The result clustering depends on the initial centroids, which normally are ramdonly chosen, making this a non-deterministic method. Besides, there are no guarantees that it will converge to the global optimum. Often, *k*-means terminates at a local optimum (HAN; KAMBER; PEI, 2011). In practice, it is common to run *k*-means multiple times to obtain good results.

Picking an appropriate number of clusters remains an open problem. Methods like the proposed by Fang and Wang try to attack this issue (MELNYKOV; MELNYKOV, 2012). However, there is still no definitive algorithm for choosing the best number of clusters for all appli-

Figure 2.1 – The steps for k-means clustering. (a) Pick k random elements (in this case $k = 3$). (b) Assign the remaning objects to the nearest centroid, creating k clusters. (c) Recalculating the centroid from the elements belonging to each cluster. (d) Repeat Steps 2 and 3 until reach convergence.



(a) (b) (c) (d)

Source: (WIKIPEDIA, 2015).

cation domain optimally.

Another disadvantage of k-means is the choice of the initial clusters. This decision impacts the results achieved. Several techniques have been proposed to choose good initial centroids. Arthur et al. introduce the most famous technique, called k-means++ (ARTHUR; VASSILVITSKII, 2007). K-means++ precompute the data trying to find better seeds.

Concluding, although the disadvantages of the number of clusters and the initial centroids, K-means is broadly used by several techniques, mainly for its simplicity and fast convergence, besides the possibility of being expanded to a N-dimensional domain.

2.2 Trajectories

"To occur is to take place. In other words, to exist is to have being within both space and time. This entanglement of thing, space and time adds to the difficulty of analyzing these concepts" (PEUQUET, 2002). This entanglement is the reason why visualizing trajectory data is so important, and challenging. But, what are trajectories?

Trajectories are objects or agents (such as people, vehicles, or animals) moving through space over time. The paths of the moving objects are usually continuous, that is, they create a time-ordered sequence of positions. Unfortunately, movement data are discrete, meaning that we can only have this information at some time moments, and the positions between the timestamps are uncertain (ANDRIENKO et al., 2013). Figure 2.2 presents a single trajectory of a runner at the Central Park, New York. We show 100 positions of the runner recorded from a GPS device. Can we extract meaningful information from these records? What if we were dealing with groups of objects?

To extract information and obtain insights from trajectory data, we need to understand

Figure 2.2 – Single trajectory of a runner at the Central Park displayed over a map. The trajectory is color-mapped from yellow (start) to red (end).

Source: Author.

it before. In the rest of this section, we introduce the data we are concerned with, as also the questions that analysts might ask about such data.

2.2.1 Data

Peuquet (PEUQUET, 1994) distinguishes three major components in spatiotemporal data: space (*where*), time (*when*), and objects (*what*). Each one of these components may

have some characteristics (attributes). The primary goal of trajectory analysis is to discover the relations among these three elements and among components and attributes. By finding such relations we can answer questions to analysis, such as describing the location (*where*), the details at a given time (*when*) and objects (*what*) (ANDRIENKO; ANDRIENKO; GATALSKY, 2003).

Our primary concern is to explore the attributes of a group of objects along trajectories in space and time. More and more, people carry GPS devices (in their phones ou watches), and a variety of other sensors, such as barometers, heart-rate monitors, etc. The sensors provide a range of unexplored information. Moreover, we can directly derive attributes from raw trajectory data (speed, direction, acceleration, turn, among others).

2.2.2 Tasks

The questions about trajectory data are directly related to the three components, i.e., space, time, and objects. Most of the problems can be described by these three elements and the relations between them. Peuquet enumerates three type of questions (PEUQUET, 1994):

- *when* + *where* $\rightarrow$ *what*: Describe the objects or set of objects that are present at a given location or set of locations at a given time or set of times.
- *when* + *what* $\rightarrow$ *where*: Describe the location or set of locations occupied by a given object or set of objects at a given time or set of times.
- *where* + *what* $\rightarrow$ *when*: Describe the times or set of times that a given object or set of objects occupied a given location or set of locations.

Visualization, analysis, and clustering of trajectory data aim at understanding the interrelations between spatial (S), temporal (T), and attribute (A) components of the moving objects. The main goal in trajectory data is to understand the *behavior* of the attributes with respect to space and time, that is, understand the functional dependency $S \times T \rightarrow A$ (TOMINSKI et al., 2012).

Three behavior-related objectives can be derived from that functional dependency: (1) behavior characterization of A over the whole or parts of S and T, like identifying the behavior of the heart rate of a runner on a rise over a time interval, (2) search occurrences of a particular behavior in S and T, such as detect abnormal variations of the heart rate in a running, and (3) comparison of the behavior of A in different regions of S or different time intervals in T, for example, comparing the behavior of different runners in a particular path or even from the same

runner in distinct days.

Since the analysis of the overall behavior $S \times T \rightarrow A$ is a difficult task, we can decompose it into subtasks. By fixing $s \in S$ at a defined place, we can examine the local behavior with respect to space of A over T: $T \rightarrow A$. By focusing on selected times $t \in T$, we can analyze the local behavior with respect to time of A over S: $S \rightarrow A$. After analyzing local behavior, the goal remains in understanding the overall behavior $S \times T \rightarrow A$. Therefore, it is indispensable that a visualization tool supports the three behavior-related objectives for both local and overall behavior.

3 RELATED WORK

The three components space, time, and attributes of trajectory data lead to a particular interest in discovering relationships, similarities, and patterns among this data. Because of that complexity and recent popularity, trajectory datasets have received a significant amount of attention. Andrienko and Andrienko provide an overview of visual analysis for movement data (ANDRIENKO; ANDRIENKO, 2012), while Kehrer and Hauser give a more general survey of multiple-faceted data, from multiple simulation runs or heterogeneous data (KEHRER; HAUSER, 2013).

Many fields have been given attention to trajectories, including visual analytics, GI-Science, and data mining. Existing works focus mainly on analyzing spatial and temporal aspects of trajectories, detecting stops, interactions between trajectories, among other events, and aggregating and clustering trajectories in space and time. However, most of the works disregard the extra attributes of the trajectories that may be as important as spatial and temporal data in identifying the behavior of the moving objects.

In view to handling trajectory data, visualize, clustering, and analysis are vital operations in the successful understanding of the data. This chapter reviews the state-of-the-art techniques leading with trajectory data. It presents visualization approaches (Section 3.1) for obtaining insights from the data, clustering techniques (Section 3.2) trying to group the trajectories in some described manner, and interactive methods (Section 3.3) exploring the trajectory data. This Chapter shows most of the state-of-the-art techniques.

3.1 Visualization of Trajectory data

The field of information visualization is quite young. Nevertheless, it has a crucial importance in several areas, once visual displays provide the highest bandwidth channel from computer to human (WARE, 2012). This importance is due to us acquire more information through vision. Therefore, if a visualization is presented well, we can quickly interpret amounts of information that we could not if the data were presented in a simple table.

Ware (WARE, 2012) enumerates a few advantages of visualization:

- Provides an ability to comprehend vast amounts of data
- Allows the perception of emergent properties that were not anticipated
- Often enables problems with the data itself to become immediately apparent

- Facilitates understanding of both large-scale and small-scale features of the data
- Facilitates hypothesis formation

Because of these advantages and due to increasing growing in data related both to time and space, visualization techniques allowing useful knowledge discovery are indispensable to understanding the data. As the spatiotemporal data has spatial coordinates, the most widely applied method for representing the data are maps, just because spatial characteristics, such as cluster patterns and associated rules, are often more easily identified, as opposed to data in a table (ZHONG et al., 2012).

The most obvious problem in the visualization of trajectory datasets is the sheer amount of overplotting that is to be expected: trajectories overlap and cross one another, and features of interest might get buried in the graphical noise. This amount of overplotting in trajectories can become too high, making the resulting visualization complex to understand. Thereby, visualizations for trajectories data should provide an overview of the aspects of the trajectories, helping to validate what is known, and revealing what is unknown (WANG; YUAN, 2014).

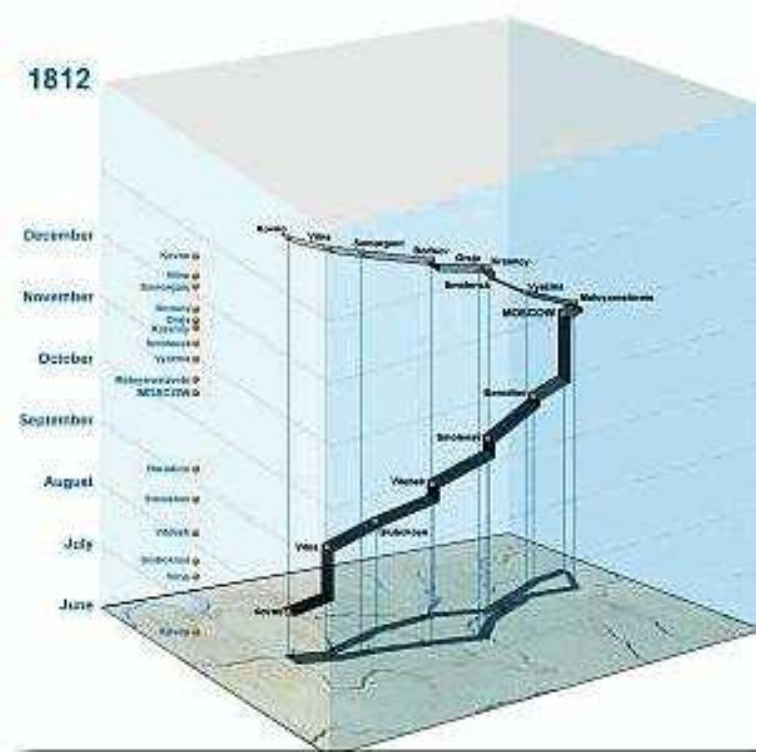
One of the most frequent attempts to reduce overplotting is to draw the trajectories in a three-dimensional view, as the well-known space-time cube (KRAAK; ORMELING, 2003). Generally, axes x and z represent the map (longitude and latitude, respectively), and the y-axis corresponds to time (Figure 3.1).

Using three-dimensional visualization allows individual trajectories to be picked out, but three-dimensional views of two-dimensional data come with problems of their own, like occlusion and clutter, and can be harder to obtain insights and interact with, typically resulting in cluttered and illegible visualization (TORY et al., 2007). Furthermore, three-dimensional visualization can be applied only to groups of trajectories behaving as a single unit. Otherwise, the data becomes hard to understanding.

Andrienko et al. proposed methods to help understanding movement behaviors and mobility patterns in a space-time cube (ANDRIENKO; ANDRIENKO; WROBEL, 2007). They developed an interactive visual display framework, applying pre-processing for data enrichment, and extraction of significant places and trips. The framework allows analyzing the data in different aspects, like clustering and summarization of trips and visited places. However, it is suitable for a small set of simple routes without low intersections.

Andrienko et al. use a space-time cube to analyze group movements (ANDRIENKO et al., 2013). They transform physical space into relative positions related to group center and direction. It creates an abstract group space, allowing investigate how the members behave within the group, but not how the whole group behaves.

Figure 3.1 – Representation of object movement in a "space-time cube". Axes x and z correspond to spatial coordinates over a map (longitude and latitude, respectively). Y-axis represents the time dimension.



Source: (KRAAK; ORMELING, 2003).

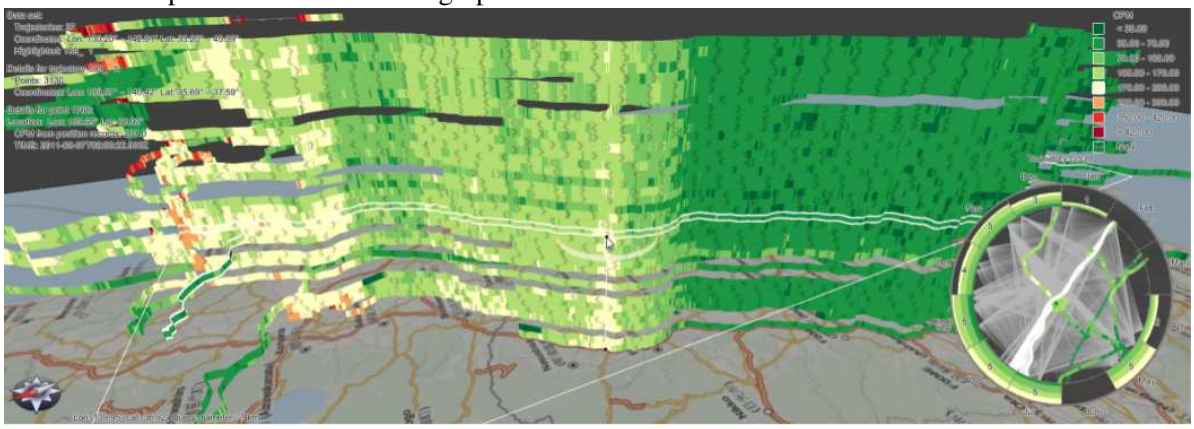
Some techniques use aggregation to reduce and cluster the trajectories. Andrienko and Andrienko proposed a method for spatial generalization and aggregation of movement data, combining trajectory flows among areas (ADRIENKO; ADRIENKO, 2011). They partition groups spatially from characteristic points extracted from the trajectories, resulting in an abstraction with essential characteristics of the movement. Janetzko et al. designed a visual abstraction to reduce clutter in spatiotemporal data, partitioning the data into dense regions linked by low-density transition zones between them (JANETZKO et al., 2013). Theses techniques based on aggregation can reduce clutter, but are difficult to obtain insights with no previous knowledge of the data.

Those previous works concern only about movement patterns. However, trajectory data is much richer in information and those works disregard the additional characteristics of the data. In the past few years, some works have been focusing on extending the visualizations of trajectory datasets to the attribute-rich case.

Tominski et al. stacked 3D trajectory bands along a 2D map, mapping attribute values to color (TOMINSKI et al., 2012). They also display a 2D time graph showing temporal information in full detail (Figure 3.2). Tominski and Schulz extended those ideas, developing a space-time continuum visualization called The Great Wall of Space-Time, referring to 2D

geographical space and 1D linear time (TOMINSKI; SCHULZ, 2012). This wall can display different visual representations for analyzing spatiotemporal variations, like color-coding and parallel coordinates. Andrienko et al. connect the wall with space-time cube, allowing visualization of trajectory attributes and facilitating interactive pattern detection (ANDRIENKO et al., 2014). However, this visualization techniques are well suited for only a few dozens of trajectories following similar paths, and the same occlusion and spatialization problems remain.

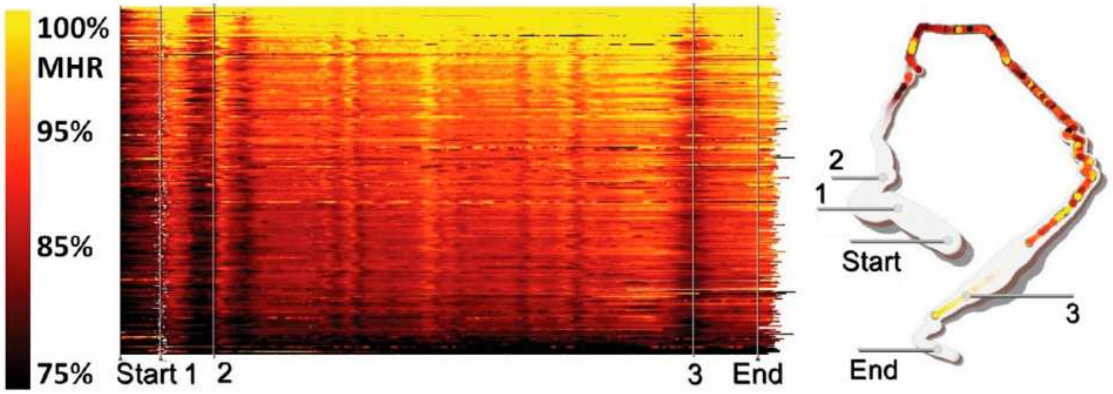
Figure 3.2 – Staking-based visualization of trajectory attribute data. A map is over the x and z axes. Each trajectory is stacked in the y dimension, where an attribute (speed, in this case) is mapped to color. The visualization enables analyzing the behavior of a few trajectories sharing the same path. The temporal information is presented in a 2D time graph.



Source: (TOMINSKI et al., 2012).

Oliveira et al. propose a technique to visualize attribute time series over a single trajectory(Figure 3.3) (OLIVEIRA et al., 2013). This visualization mitigates the 3D occlusion problems but is not directly applicable to unrestricted movement data. AFKM is potentially helpful in these cases with its unified treatment of trajectory position and attribute values.

Figure 3.3 – Example of a visual design for analyzes running races. Linear heatmap is presented on the left. Each time series for the heartbeat is normalized and plotted on a single line, where the x-axis represents the distance, and the color encoded the heartbeat frequency. On the right, an augmented track view is displayed, with the same distance markers to allow correlate patterns.



Source: (OLIVEIRA et al., 2013).

Our technique concerns about the various attributes associated with trajectories. We offer no contribution to solving intersections and overlapping. However, our method naturally encodes trajectory's features, and the works presented in this section could benefit from AFKM clusters. AFKM could be used in conjunction with any trajectory visualization method as a way to reduce harmful overplotting since its clusters will tend to separate unrelated trajectories.

3.2 Trajectory Modeling and Clustering Analysis

When previous knowledge of the dataset suggests that a small number of different behaviors account for a substantial amount of the variation in the data, clustering analysis is an attractive option. Clustering is one of the general approaches to explore large datasets since it allows specialists to analyze the data in a higher-level representation rather than individual objects (KISILEVICH et al., 2010). The methods for clustering trajectories aim to detect and group similar instances having similar paths, i.e., catching interesting patterns in the shape and the spatiotemporal characteristics of trajectories.

In the context of trajectory data, often the data has a small number of behaviors, and many clustering techniques have been published. Liao gives a survey of time series clustering (LIAO, 2005). It divides the clustering approaches intro three categories, depending if they work directly with raw data, or indirectly extracting features or building models from the raw data. Methods working with raw data are generally sampled at the same interval and needs to handle with high dimensional space and highly noisy. An alternative is the use of methods based on features to address these problems. However, the extracted features are usually application dependent, and may work well only for a particular application. The model-based approaches consider some model guiding the data, i.e., similar trajectories will have similar models.

Kisilevich et al. give a survey on spatiotemporal clustering methods in (KISILEVICH et al., 2010), providing a classification of spatiotemporal data. The descriptive and generative model-based clustering has as the main objective derive a global model that describes the whole dataset. Distance-based clustering methods define distance functions that encapsulate the concept of similarity among the data items. Moreover, the density-based methods use a density threshold for each object. It allows the technique to distinguish the relevant data from noise.

Some works use a progressive approach applied to clustering. It allows the clustering to be performed in an interactive manner. However, could lead to a trial and error process. Rinzivillo et al. developed a framework allowing visualization and interaction techniques in a progressive clustering approach (RINZIVILLO et al., 2008). Likewise, Andrienko et al. also

provides an progressive technique, allowing simple distance functions to be applied at each step, facilitating interpretable outcomes and allowing the user controls which distance function will be applied and in which step. The interactive nature of AFKM allows it to be used in a progressive sense (WILLIAMS; MUNZNER, 2004; FISHER et al., 2012) to achieve interactive even for extremely large datasets.

In some situations, it is preferable working with subsets of the entire dataset, due to its size. Thereby, Andrienko et al. applied a density-based clustering algorithm into a small subset of instances, building a classifier to attach the remaining objects (ANDRIENKO et al., 2009). Similarly, our approach could select a subset of elements, generating the correspondents attributes fields and attaching the remaining time series. Also, the user could define distinct attributes on each level of clustering. It facilitates user exploration and understanding of generated clusters.

Lee et al.'s TraClus is a popular method for trajectory clustering based on first partitioning large trajectory into smaller, "atomic" segments that are then grouped (LEE; HAN; WHANG, 2007). They observe that previous methods fail to detect common portions of the trajectories. To address this, they partition each trajectory into a set of line segments and then group similar sub-trajectories into clusters. While this is valuable where the problem is the detection of relatively small overlapping segments, TraCLUS does not offer the potential of uncovering large-scale patterns in the data.

Ferreira et al.'s VFKM is the clustering method that's closest to ours (FERREIRA et al., 2013a). It is a model-based clustering approach, built on vector-field fitting. They treat the trajectories as a whole and use an iterative model similar to K-means clustering. Their technique groups trajectories that can be approximated well by streamlines of a single vector field, and has the ability to capture global patterns. One of our contributions is in the observation that a generalization of VFKM maintains its favorable performance characteristics while vastly generalizing its applicability to multiple attribute fields. Also, if desired, our analysis yields an algorithm that can behave exactly as classic k-means (see Chapter 4 and Chapter 6).

Direct aggregation of trajectory attributes is also a popular method: instead of visualizing the raw data, aggregation seeks to find a small number of representative attributes from the entire dataset, assuming that the data attributes that become hidden are essentially noise (ANDRIENKO; ANDRIENKO, 2011; ANDRIENKO; ANDRIENKO, 2008). Janetzko et al. propose clustering trajectory positions themselves to generate a transition graph (JANETZKO et al., 2013); this is an interesting approach, but it is unclear how to extend it in the case of attribute-rich trajectories. The attribute fields derived by AFKM can be directly interpreted as

aggregations since they are the minimum-error fields over the particular clustering assignment.

As we briefly presented here, the trajectories clustering techniques often attempt to detect only spatial similarities (at most, they utilize the speed attribute). Our key observation is that clustering trajectories with a high number of attributes could not detect significant patterns in the data. Thereby, we bring an attribute-trajectory clustering encoding an explicit relationship between the trajectory attributes.

3.3 Interactive Exploration of Trajectory Data

Interactive exploration tools are another popular approach to the visual analysis of trajectory data. Andrienko et al. argue that the right visual analytic tools allow humans to interpret and understand movement behaviors and mobility patterns (ANDRIENKO; ANDRIENKO; WROBEL, 2007). Likewise, Liu and Heer discovered that interactive latency impacts the outcomes of exploratory visual analysis, reducing interaction, observations and overall performance due to the delay (LIU; HEER, 2014). Therefore, providing adequate support for interactive exploration of trajectory data is indispensable while challenging.

Meghdadi and Irani proposed a visual analytics system for interactive and analytic exploration of video data, summarizing and visualizing both the video content and trajectories of individuals (MEGHDADI; IRANI, 2013). Ferreira et al. explored big spatiotemporal taxi cab trips (FERREIRA et al., 2013b). The system supports origin-destination queries, different aggregations and visual representations for exploration and comparison of query results, enabling the study of mobility across the city.

Wang et al. explored sparse traffic trajectory data (WANG et al., 2014). The system allows three level of detail; global, cell, and correlation explorations, providing an overview of the data, as well as to analyze and correlate the trajectories. Krueger et al. developed a system for interactive visual analysis of semantically enriched movement data, providing a geographic and temporal exploration (KRUEGER; THOM; ERTL, 2014). As we can see, urban data is widely used in several works, due to easy data collection and, in several cases, the data is publicly available. Likewise, Wang et al. worked with urban traffic data, building a visual interface to explore traffic jams and their propagations (WANG et al., 2013). Wang et al. make some variations of the timeline visualization of the 2D trajectory for comparison of urban trajectories (WANG; YUAN, 2014). The framework allows visualizing changes in spatial features, such as the curvature, straightness, turns, and stops.

AFKM is efficient enough to be used in an interactive setting (as we discuss in Chap-

ter 6), and so could be seen as a way to naturally incorporate attribute-rich trajectory analysis in these systems. Liu et al. have shown that latency in interactive tools can have a significant impact on the overall data exploration experience by users (LIU; HEER, 2014). While the latency introduced by AFKM places the technique in the "bad" region of latency as identified by Liu et al., it appears to be the fastest algorithm for trajectory clustering.

Most of the trajectory techniques focus on the spatial information of trajectories, but there are still a few on temporal exploring and navigation along the trajectory (KONDO; COLLINS, 2014) or visualizing the attribute signatures of multivariate trajectory data (TURKAY et al., 2014). In the current state of the art, it appears that if any attribute-rich trajectory clustering algorithm can be used interactively, it is AFKM.

Figure 3.4 – A sketch-based clustering using template query. The user sketches an ellipse pattern (top left), and the matching streamlines are identified and displayed.

Source: (WEI et al., 2010).

3.4 Sketching

A more natural and intuitive way of interaction is the use of sketch-based interfaces. Although sketching in 3D space is possible, we analyze 2D sketching-based interfaces, once it is more intuitive and convenient for the user. One of the main benefits of the use of sketch-based interfaces is that they allow users to draw directly on top of the data. It makes easier for the user to explore what kind of patterns one is expecting to discover in the data. Shen et al. (SHEN et al., 2014) provide an overview of sketch-based interface and applications. For this reason, in this section, we present the two works most related to our sketch-based approach.

Schroeder et al. presented a sketch-based system called Drawing with the Flow for illustrating 2D vector fields (SCHROEDER; COFFEY; KEEFE, 2010). The interface is designed to artist and illustrators, who typically do not have a background in programming, but have a high sense of visual design. They draw directly on vector fields, making accurate hand-drawn illustrations. Wei et al. also developed a 2D sketch-based interface (WEI et al., 2010). Their method tries matching similar streamlines from the 2D pattern drawn by the user. It works like a clustering technique retrieving the closest streamlines to the input pattern. Figure 3.4 displays a user input (top left), and the matching streamlines are retrieved.

In AFKM, the idea of the pipeline is similar to the work of Wei et al. (WEI et al., 2010); a user sketches a pattern, and the system matches similar trajectories. However, we just draw sketches over each one of the desired available attributes, creating a new trajectory. This new trajectory serves as the model for one cluster in the clustering process (more details in Section 6.4).

4 ATTRIBUTE FIELD K-MEANS (AFKM)

AFKM is the main contribution of this thesis. In this chapter, we define basic concepts and fix the terminology used in the rest of the thesis, as well present our AFKM technique.

4.1 Terminology

In general, a multivariate time series is a function $\gamma(t) : [t_0^\gamma, t_1^\gamma] \to \mathbb{R}^D$ from a time interval to a high-dimensional space representing the different attributes. Using as motivation the special case of geographical trajectories, we are interested in multivariate time series from which we can single out spatial components. As described later, this makes it possible for us to exploit the spatial attribute behavior to model these time series. For this reason, in this work we define a trajectory with M-attributes, or just *M-trajectory* for simplicity, as functions of the form $\overline{\gamma} : [t_0^\gamma, t_1^\gamma] \to \mathbb{R}^2 \times \mathbb{R}^M$, such that $\overline{\gamma}(t) = (\gamma_S(t), \gamma_A(t))$, where $\gamma_S(t) \in \mathbb{R}^2$ denotes the spatial component of the trajectory and $\gamma_A(t) \in \mathbb{R}^M$ denotes the attributes of the time series. An M-attribute field, or simply M-field, is defined as a function $f : \mathbb{R}^2 \to \mathbb{R}^M$. This shows the complexity of the problem we face. The trajectory data encompass spatial and temporal data, as well as numerical and/or categorical features.

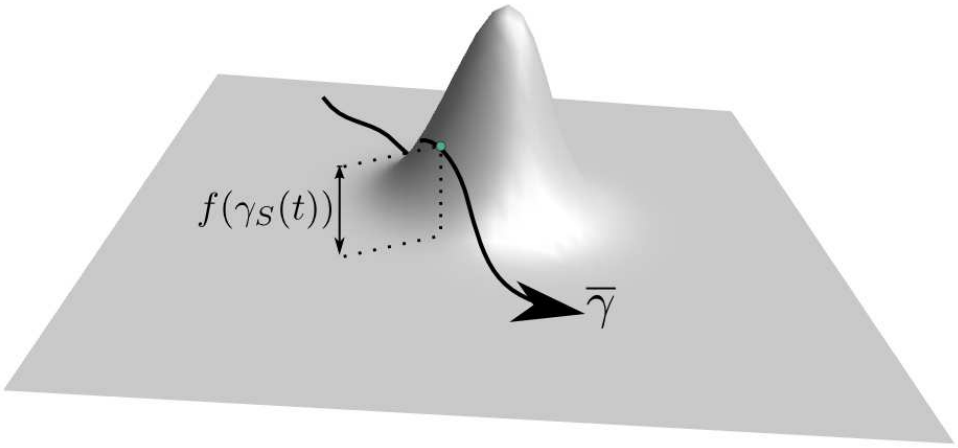
We say that a trajectory $\overline{\gamma}$ is *consistent* with an attribute field f, if $f(\gamma_S(t)) = \gamma_A(t)$ for all $t \in [t_0^\gamma, t_1^\gamma]$. These concepts are illustrated in Fig. 4.1 for the case where $M = 1$. Similarly, given a set of trajectories $T = \{\overline{\gamma}_1, \overline{\gamma}_2, ..., \overline{\gamma}_n\}$ we say that an attribute field f is consistent with the set T, if each γ_i is consistent with f for $i = 1, 2, ..., n$.

Given a enupla $x = (x_1, , ..., x_M)$ we denote the d^{th} entry of x by $x^{(d)}$. Thus, given an M-trajectory $\overline{\gamma}$ its d^{th} attribute is denoted by $\gamma_A^{(d)}$. Similarly, the d^{th} attribute of an M-field, f, if denoted by $f^{(d)}$.

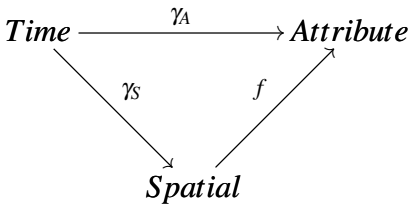
4.1.1 General Overview

As briefly discussed before, the goal of our method is to capture patterns in trajectory attribute values. The main idea behind it is to do so by exploring the attributes's spatial distribution via attribute fields, as illustrated in the following diagram

Figure 4.1 – Illustration of basic concepts. An 1-attribute f is represented as a height field. A trajectory $\overline{\gamma}$ consistent with f is also shown.



Source: Author.



There are two main advantages of this approach. First, attribute fields are intuitive and relatively easy to visualize (individual attributes are basically heatmaps). Second, as discussed in the following it allows us to define the *fitness* of a set of trajectories in terms of consistency. Intuitively, we say that a set of trajectories T is comprised of similar trajectories if there exists an attribute field f which is consistent with T. This notion of fitness is used in our method to group trajectories with similar attribute distributions.

In general, however, for a non-trivial set of trajectories there is no attribute field f consistent with it. Therefore, in order to handle real world scenarios we need to define consistency in an approximate sense. We do this by defining a measure of how consistent an attribute field f is to a set of M-trajectories $T = \{\overline{\gamma}_1, \overline{\gamma}_2, ..., \overline{\gamma}_n\}$ as follows

$$E(f,T) = \sum_{i=1}^{n} \int_{t_0^{\gamma_i}}^{t_1^{\gamma_i}} ||f(\gamma_S(t)) - \gamma_A(t)||^2 \, dt, \tag{4.1}$$

It is also possible to extend the previous measure to include domain knowledge by the use of weights for the different attributes shown below:

$$E_W(f,T) = \sum_{i=1}^{n} \sum_{d=1}^{M} w_d \int_{t_0^{\gamma_i}}^{t_1^{\gamma_i}} \left|\left| f^{(d)}(\gamma_S(t)) - \gamma_A^{(d)}(t) \right|\right|^2 dt, \tag{4.2}$$

where $W = (w_1, ..., w_M)$ denotes a weight vector for the different attributes. By changing these weights, one can express how the consistency of one attribute is preferred compared to the others. Since this last definition includes the previous one we are always going to assume that weights are defined and for notation simplicity we omit them from formulas.

4.2 Technique

In this section we describe the main contribution of this thesis, namely the *Attribute Field K-Means* technique, that uses the notion of approximate consistency to perform clustering on a set of M-trajectories $T = \{\overline{\gamma}_1, \overline{\gamma}_2, ..., \overline{\gamma}_n\}$. Assuming we are given the number of clusters K to be found, the goal of this technique is to to find $M-$attribute fields $f_1, ..., f_K$ and assignment $\Theta : \{1, ..., n\} \to \{1, ..., K\}$ that minimize the following total consistency measure:

$$E(f_1, ..., f_K, \Theta) = \sum_{j=1}^{K} \lambda \left|\left| \Delta f_j \right|\right|^2 + \sum_{\overline{\gamma}_i \in \Theta^{-1}(j)} E(f_j, \overline{\gamma}_i). \tag{4.3}$$

In this formula the use of the Laplacian of the attribute field was inspired by the method by Ferreira et al. (FERREIRA et al., 2013a) in which this term is used as a regularization term for the optimization (in fact, as described later, our technique is a strict generalization of (FERREIRA et al., 2013a)). Furthermore, $\Theta^{-1}(j)$ denotes the set of trajectories assigned to the j^{th} cluster and λ is a weight for the optimization that can be tuned to control the *smoothness* of the derived attribute fields.

In order to minimize the energy in Eq. 4.3, we use an alternating optimization approach (JAIN; NETRAPALLI; SANGHAVI, 2013). This is an optimization strategy which consists in an iterative process that alternates between a number of subproblems (two in our case) each of which considers only a subset of the optimization variables of the initial problem and therefore are more tractable than the original problem. In each iteration, a current solution is kept and via the solution of the subproblems this solution is improved until a stop condition is met. In our particular scenario, the current solution are the current estimates of the attribute fields $f_1, ..., f_K$ as well as Θ which represents the best assignment of trajectories to clusters. In our technique,

the subproblems are called *Attribute Field Fitting* and *Cluster Assignment* (see pseudo-code in Alg. 1) and are described in more details in the following.

Algorithm 1 Attribute Field K-Means Outline

Require: k: # of clusters, $\mathscr{T} = \{\overline{\gamma_1}, \ldots, \overline{\gamma_n}\}$: Trajectory data
Ensure: $V = \{f_1, \ldots, f_k\}$, $\Theta : \mathscr{T} \to \{1, \ldots, k\}$
  $\Theta \leftarrow$ Initialize($\mathscr{T}$,k)
  repeat
    for $i = 1$ to k **do**
      $f_i \leftarrow fitAttributeField(\Phi^{-1}(i))$
    end for
    for $i = 1$ to n **do**
      $j_0 \leftarrow \underset{j \in \{1,2,\ldots,k\}}{\operatorname{argmin}} E(f_j, \overline{\gamma_i})$
      $\Phi(\alpha_i) \leftarrow j_0$
    end for
  until converge

Attribute Field Fitting This step consists in fitting an attribute field for each of the current clusters (which assumed to be fixed at this step) and it is denoted by the function *fitVectorField* in Alg. 1. In more details, for each $j = 1, ..., k$ we fit an attribute field by solving the following problem

$$f_j = \min_f \lambda \, ||\Delta f||^2 + \sum_{\overline{\gamma_i} \in \Theta^{-1}(j)} E(f, \overline{\gamma_i}). \tag{4.4}$$

Cluster Assignment In this step we optimize the cluster assignment Θ, while we are given attribute fields $f_1, ..., f_K$, which are assumed to be fixed at this step. In order to do so, for each trajectory $\overline{\gamma_i}$ we solve the following problem

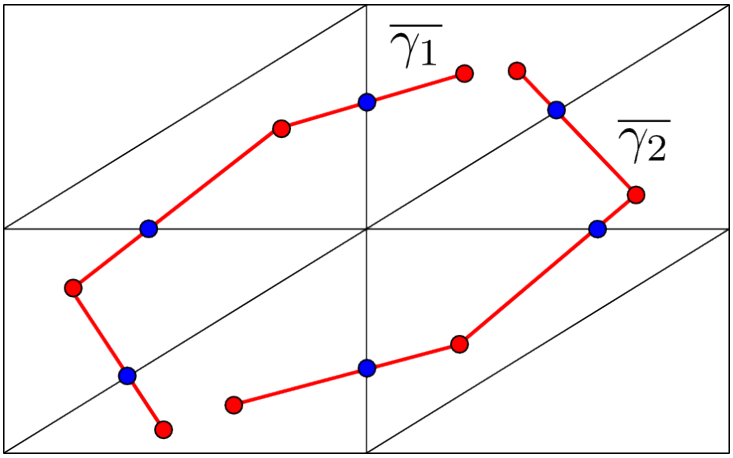
$$\Theta(\overline{\gamma_i}) = \min_{j=1,\ldots,k} E(f_j, \overline{\gamma_i}). \tag{4.5}$$

In the rest of this section we describe how we can discretize the problems in Eqs. 4.4 and 4.5 as well as different initialization strategies for our method.

4.2.1 Discretization

Trajectory data is generally represented as a set of samples, i.e., each $\overline{\gamma} = (\gamma_S, \gamma_A)$ is given as a discrete set of samples $\{(\gamma_S(t_1), \gamma_A(t_1)), (\gamma_S(t_2), \gamma_A(t_2)), \ldots, (\gamma_S(t_n), \gamma_A(t_n))\}$ and it is

Figure 4.2 – Illustration of the triangular grid use for attribute field fitting. Trajectories are tessellated (in the figure, blue vertices are added) so that each trajectory segment is contained in a grid face. In this case $R = 3$.



Source: Author.

commonly defined, we assume that both the spatial and attribute components can be modeled by linear interpolation between two samples.

We use a discretization strategy similar to the one used by Ferreira et al. (FERREIRA et al., 2013a). First, all the attribute fields are defined on a triangular grid that covers the spatial domain where the trajectory dataset T is defined (see Fig. 4.2). For notation simplicity, we consider grids with R^2 vertices, where R denotes the grid resolution. Attribute fields are defined on the vertices of this grid and the values inside each triangular face is obtained by linear interpolation.

The first step in our algorithm is to tesselate the spatial component for each trajectory in T so that each trajectory segment (portion between two consecutive samples) lies completely inside a face of the grid (see Fig. 4.2). With this in mind, we can compute $E(f, \overline{(\gamma)})$ for a single trajectory by computing this for each of its segments, s, which is given by

$$
\begin{aligned}
E(f,\overline{\gamma}) = \sum_{s\in\overline{\gamma}} \int_{t_0^s}^{t_1^s} ||f(\gamma_S(t)) - \gamma_A(t)||^2 \, dt = \\
\sum_{s\in\overline{\gamma}} \sum_{d=1}^{M} \int_{t_0^s}^{t_1^s} \left|\left| f^{(d)}(\gamma_S(t)) - \gamma_A^{(d)}(t) \right|\right|^2 dt = \\
\sum_{s\in\overline{\gamma}} \sum_{d=1}^{M} (t_0^s - t_s^s) \int_0^1 ||(1-\sigma)(f(\gamma_S(t_0^s) - \gamma_A(t_0^s)) \\
+\sigma(f(\gamma_s(t_1^s) - \gamma_A(t_1^s)))||^2 d\sigma = \\
\sum_{s\in\overline{\gamma}} \sum_{d=1}^{M} (t_0^s - t_s^s)[\frac{1}{3}||f(\gamma_S(t_0^s) - \gamma_A(t_0^s)||^2 + \frac{1}{3}||f(\gamma_s(t_1^s) - \gamma_A(t_1^s)||^2 + \\
\frac{1}{3}\langle f(\gamma_S(t_0^s) - \gamma_A(t_0^s), f(\gamma_s(t_1^s) - \gamma_A(t_1^s)\rangle].
\end{aligned}
\tag{4.6}
$$

The term inside the summation on the right-hand side of the last equality, can be written as $||\Lambda(C\tilde{f} - b)||^2$, where C is a $2 \times R$ containing in each line the barycentric coordinates of the corresponding segment ends and b is a 2×1 vector containing the values of the corresponding attribute on the ends of the segment. Finally, the matrix Λ is used to provide the right mixture coefficients and it is given by

$$
\Lambda^T\Lambda = \begin{pmatrix} \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} \end{pmatrix}.
$$

4.2.2 Algorithm Initialization

As it is common to alternate optimization methods, initialization is critical to performance (MELNYKOV; MELNYKOV, 2012). In spite of being crucial it is difficult, in general, to have guarantees on the optimization performance based on the optimization. This was achieved for particular cases such as $K-$Means clustering (ARTHUR; VASSILVITSKII, 2007) and low-rank matrix completion (JAIN; NETRAPALLI; SANGHAVI, 2013). The usual solution is to use heuristics for initialization (MELNYKOV; MELNYKOV, 2012). In this work we consider three of these heuristics (described in the following).

In our technique, the initialization is performing by deriving an initial assignment Θ_0 of the trajectories to clusters. The simplest initialization method to achieve this is based on random selection, i.e., each trajectory is randomly and independently assigned to one of the K clusters. The remaining methods consist on choosing K trajectories and using those to gen-

erate K attribute fields (one for each cluster) and distributing the trajectories according to the *Cluster Assignment* step described in the previous section. We consider two ways of choosing such trajectories. The first one is to proceed in the following way: pick the first trajectory at random and the following ones are chosen to maximize the approximate consistency measure with the fields generated by the previously, i.e., for $j = 2, ..., n$ we choose trajectories $\gamma_j = \max_T \min_{j=1,...,j-1} E(f_f, \gamma)$. Finally, the second and last method considered is an probabilistic version of the second one which works in a similar manner as the K-mean++ initialization for regular K-Means (ARTHUR; VASSILVITSKII, 2007). That difference between this and the previous method is that instead of deterministically choosing the trajectory that maximizes the approximate consistency with the previously derived fields, trajectories are chosen with probability proportional to the approximate consistency measure with the fields generated by the previously chosen trajectories. In more details, if we define, for $j = 2, ..., n$, $D_j = \max_T \min_{j=1,...,j-1} E(f_f, \gamma)$, then each trajectory $\overline{\gamma_j}$ is chosen with probability $\frac{D_j}{\sum_i D_i}$.

4.2.3 Incorporating User Feedback

Most of the clustering techniques are used as black boxes. Many of these techniques depend on initialization and therefore it is common that they produce non-optimal separations. One solution to this problem is to include user feedback as part of the exploration process. In AFKM, we allow user input in the form of fixing $\tilde{k} \leq k$ of the attribute fields $f_1, ..., f_k$ that are manipulated in Alg. 1. We modify the algorithm so that the $\tilde{k}$ attribute fields pre-defined are not changed through the execution (only the remaining ones are evaluate at each iteration). As we show in Section 6.4, we provide a simple interaction mechanism through which it is possible to define attribute fields and therefore give feedback to AFKM.

5 PROTOTYPE IMPLEMENTATION

To perform the evaluation of AFKM, we developed an interative web application. At the server side, we used Java as the programming language and Hibernate (HAT, 2015) for mapping an object-oriented domain model to the local MySQL database (ORACLE, 2015b). Thus, Hibernate deals with the results of SQL queries and their conversion to model objects. The local application runs on a Glassfish Server (ORACLE, 2015a).

At the front-end, the base programming language was JavaScript. However, we use a series of JavaScript-based plugins to develop all the features: (1) a WebGL library called THREEjs (CABELLO, 2015) to render the trajectories; (2) Leaflet (AGAFONKIN, 2015) with OpenStreetMap (FOUNDATION, 2015) to display the maps; (3) D3 (BOSTOCK, 2015), a visualization library; (4) DC.js (WOODHULL, 2015), a multidimensional charting library that integrates D3 and Crossfilter (SQUARE, 2015) (library for exploring multivariate datasets in the browser); and (5) jQuery (JQUERY, 2015), a feature-rich JavaScript library to easy manipulate the document, handle events, and make Ajax calls to the server.

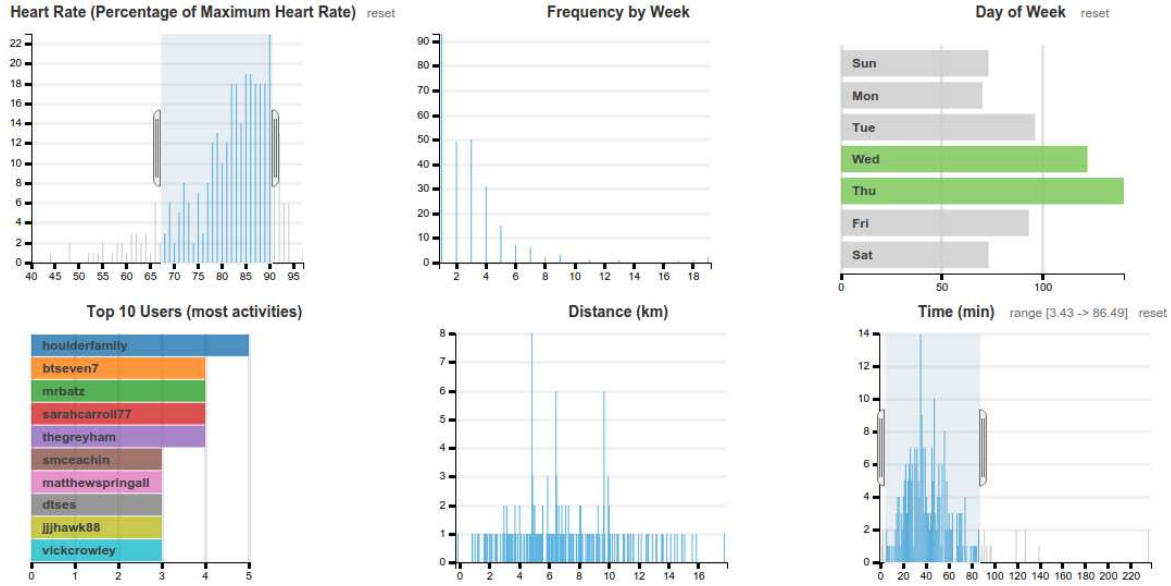
The prototype works in two phases: data filtering and interactive exploration. It displays a few datasets, previously loaded. After chosen a dataset, we present several charts with statistical information about the chosen dataset. Figure 5.1 displays six (6) of eleven (11) available charts for the central park exercise dataset. A user can use them to filter, subsequently selecting a subset with the desired range for each attribute, instead using the whole data to analyze. There are two types of charts: bar chart and row chart. We use row charts to display ordinary data, as the top users (in number of activities), or the number of activities per day of the week. In row charts, only individual (but not limited by one) elements can be selected. In contrast, in bar charts, we can select an interval of values. For example, selecting the activities in which the runner runs with maximum heart rate between 70% and 90%. In Figure 5.1, the user is filtering by Heart Rate, Time, and Day of Week attributes.

After filtering the attributes of the chosen dataset, the prototype loads the selected trajectories and presents the interactive exploration phase. This stage contains three components: Trajectories, Overview, and Control.

5.1 Trajectories

An essential aspect of clustering trajectories is to visualize them. We present the trajectories in two different forms: a map and by attributes. Figure 5.2 (A) displays the activities

Figure 5.1 – In the data filtering phase, a user can previously filter the dataset, allowing the analysis of a particular group of interesting trajectories instead of interact with the whole dataset.



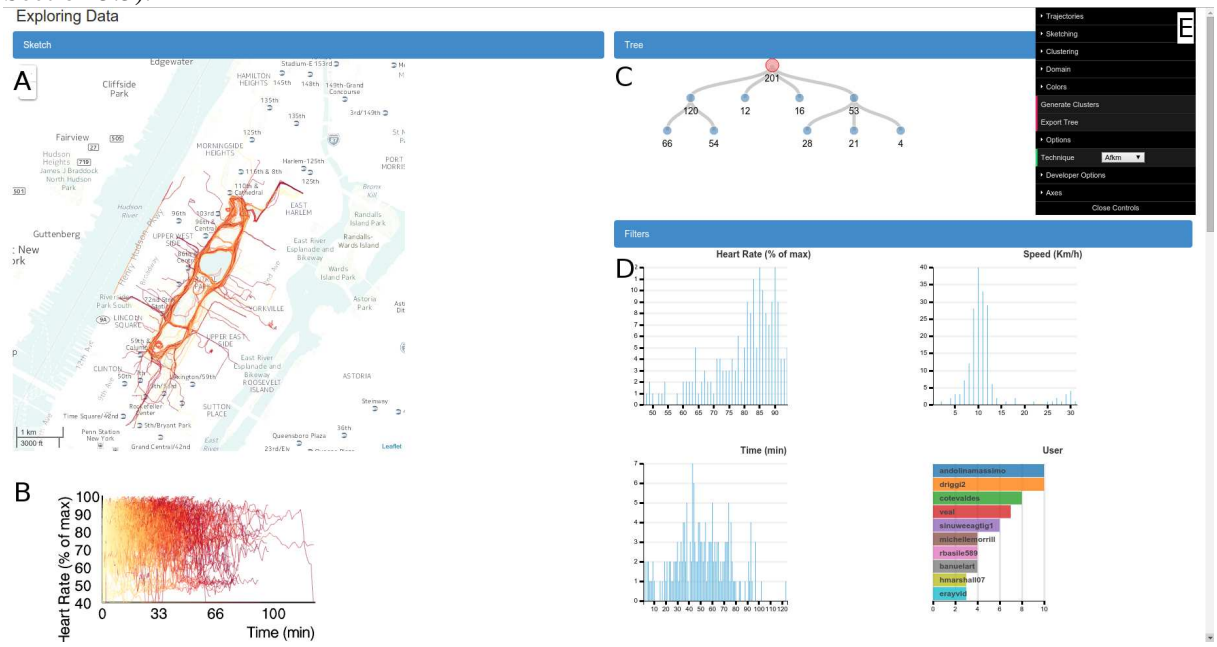
Source: Author.

in the Central Park. All trajectories are color-mapped from yellow to red, relative to time. At the map, the user is able to pan and zoom. Also, the user can analyze the trajectories by their attributes. Figure 5.2 (B) displays one attribute-time chart (there is one chart for each one of the attributes). The trajectories at the attribute-time charts are color-mapped in the same way as the trajectories in the map. A user can draw the sketches over these charts, or choose a trajectory as a model (more details in Section 6.4).

5.2 Overview

The clustering process generates k clusters each time (the parameter k may vary from each execution). Each one of the clusters have some trajectories belonging to them, and a user can, interactively, generates new clusters from a previous cluster. A tree presents this entire process (Figure 5.2 (C)). When selecting a node in the three, the trajectories (Figure 5.2 (A) and (B)) are updated, being displayed only those belonging to the selected cluster. This procedure affects also the filters (Figure 5.2 (D)). The filters are an extension of those used previously to load the data. But now, they only display statistics about the rendered trajectories. When selecting a cluster in the tree, the filters are updated to display the statistics just about the trajectories belonging to the selected cluster.

Figure 5.2 – Exploring Phase. The trajectories are color-mapped from yellow to red (relative to time). (a) A map displaying all the loaded trajectories at the Central Park, New York. (b) A 2D time chart showing the attribute Heart Rate. The image displays only one of this charts, but the prototype contains one for each attribute. The sketching process is made over these attribute-time charts. (c) Clustering Tree presenting the order of clustering. Firstly, a user defines four (4) clusters. For the resulting first cluster, the user creates two new clusters. Moreover, for the fourth cluster of the first clustering process, it was executed a new clustering process, dividing the sub-cluster into three clusters. (d) Charts presenting statistic information about the selected cluster. (e) An interface to control the visualization of the trajectories (color and opacity, for example) and to define the parameters for clustering (more details in Section 5.3).



Source: Author.

5.3 Control

This phase (Figure 5.2 (E)) contains all the parameter and options for users control and cluster the trajectories. Related to the visualization of the trajectories, a user can change the opacity and width of the rendered trajectories, set the color scheme (from a pre-defined list of palettes or create a new one), and choose which attribute is mapped to color. Options for saving a sketch or picking a trajectory as the basis of the sketch are also present.

The central part is the parameters for sketching. From the domain choice (we always define latitude and longitude as domain in our experiments) to the weights for each parameter, as also the number of clusters, grid size, smoothness, initialization method and number maximum of trajectories. Chapter 7 explains the parameters in detail. Finally, a user can define the clustering algorithm (AFKM, VFKM, or k-means) to generate clusters. We allow the user to choose VFKM and k-means for comparison against AFKM.

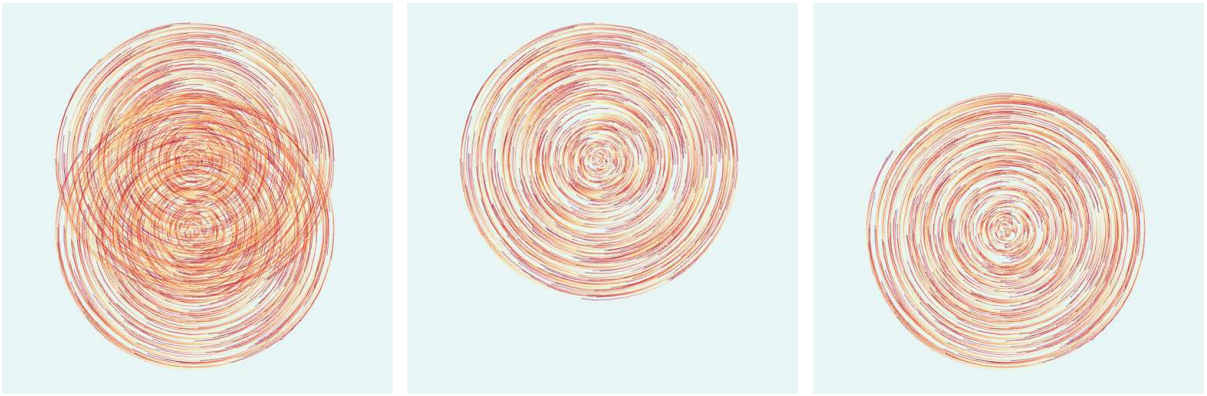
6 VISUALIZATION RESULTS AND ANALYSIS

In this Chapter, we evaluate the effectiveness of our attribute-rich trajectory clustering AFKM. To accomplish this, we perform clustering over three different datasets, showing how attribute fields created by AFKM can find better clusters for multi-attribute data. Firstly, we present the results using synthetic datasets in (Section 6.1), exploring specifics characteristics of AFKM. Then, in Section 6.2, we revisit the HURDAT2 dataset used in VFKM. Moreover, in Section 6.3 we explore an exercise dataset with activities from distinct places in the world, containing, among others attributes, heart rate and speed. We also present a sketching-based approach to help the clustering process in an intuitive and interactive manner (Section 6.4).

6.1 Synthetic Dataset

The synthetic dataset used in (FERREIRA et al., 2013a) consists of two overlapping circulatory movement patterns, each one representing 1000 trajectories. None of the trajectories forms a complete circle. Each one of them covers only a portion of the circle, at random section and distance from the center. Figure 6.1 (a) shows the two overlapping patterns. Since AFKM is a generalization of VFKM, by using latitude and longitude tangents, AFKM recovers the two patterns perfectly (Figure 6.1 (b) and (c)).

Figure 6.1 – (a) Synthetic dataset consisting of two overlapping circulatory movement patterns, each one composed of 1000 trajectories. In (b) and (c), by using the tangents of latitude and longitude, AFKM recovers the two overlapping patterns perfectly. The colors represent the relative time of the trajectories, i.e., the trajectories are mapped from start (yellow) to end (red).

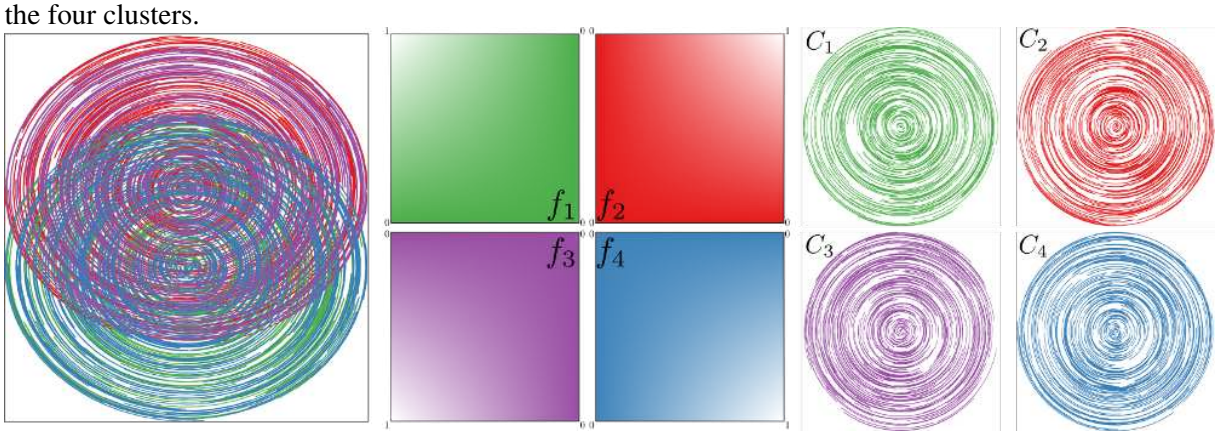


(a) 2000 trajectories (b) Cluster 1 (c) Cluster 2

Source: Author.

However, this dataset consists of spatial and temporal attributes only, and VFKM also accomplish the same result shown in Figure 6.1. Thus, we extend the synthetic dataset, includ-

Figure 6.2 – Two overlapping circulatory movement patterns, representing 2000 trajectories. Each trajectory is randomly assigned to one field (green, red, blue, purple), and a new attribute is calculated for each point based on its respective field. On the right, we observe that AFKM successfully separates the four clusters.

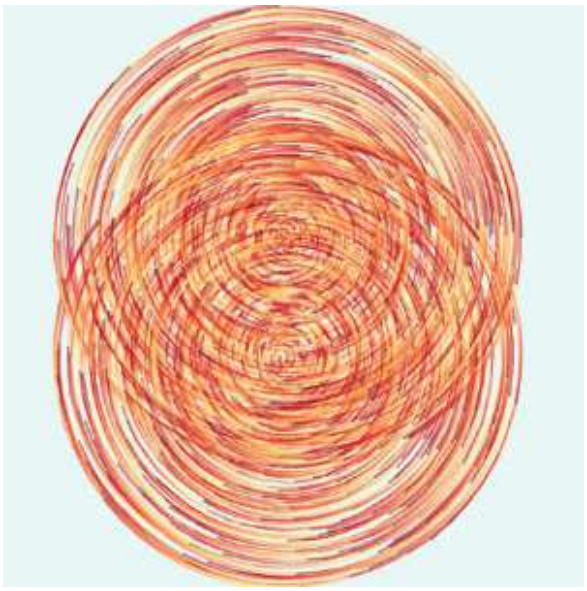


Source: Author.

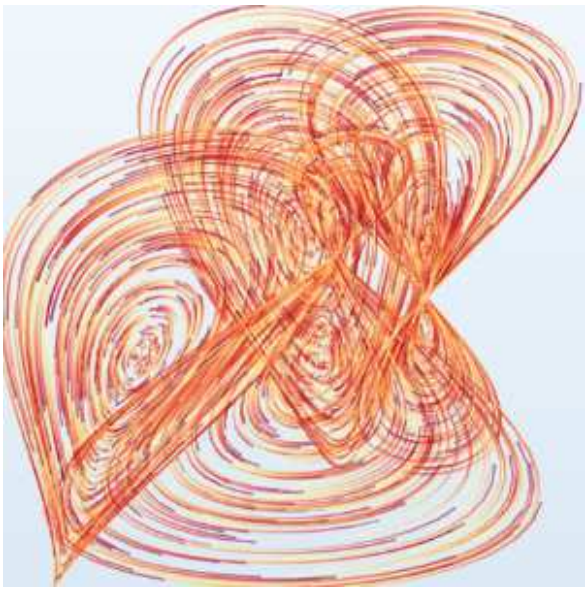
ing new attributes for each trajectory. As a first example, we divided each circular pattern in half, obtaining four groups (represented by colors green, red, purple, and blue in Figure 6.2). For each of these partitions, we created an attribute field (f_1, f_2, f_3 *and* f_4) by defining the values of this attribute field on the corners of the bounding box of the trajectories and using bilinear reconstruction. We then assign the attribute value for each trajectory, by retrieving the value of the corresponding attribute field. In other words, we created a new attribute to the trajectories using the attribute field designed for each partition. Therefore, for each point of each trajectory, we compute the Euclidean distance from the corner to the respective point and set this distance as the new attribute. As shown on the right of Figure 6.2, AFKM recovers the four clusters associated with the respective corners (top left, top right, bottom left, and bottom right).

We created a second dataset in which we start with two corners as references (Figure 6.3), instead of using only one as presented in the previous experiment. In addition to the standard longitude-latitude plot, a longitude-attribute plot reveals the four distinct patterns created. The resulting AFKM four clusters are shown using the longitude-attribute plot. AFKM perfectly recognizes clusters (c) and (d), but a few trajectories belonging to cluster (e) appear in the cluster (f). However, even though not all trajectories appear in their respective cluster, most of them do. We revisit both extended synthetic datasets in Section 7.6 to assess how other techniques managed to separate these patterns.

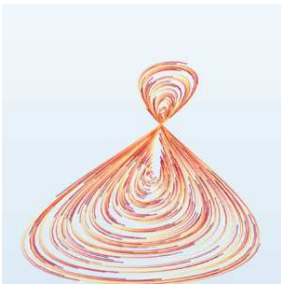
Figure 6.3 – Synthetic dataset with 2000 trajectories arranged in four patterns, illustrated by either the latitude-longitude (a) or latitude-attribute (b) plots. AFKM resulting clusters shown from (c) to (f). AFKM perfectly recognizes clusters (c) and (d). A few trajectories belonging to cluster (e) appear in cluster (f). The trajectories are color mapped with relative time (start to end).



(a) Latitude x Longitude

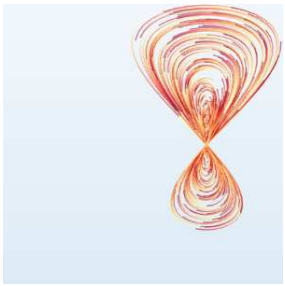


(b) Latitude x Attribute



(c) Cluster 1

(d) Cluster 2



(e) Cluster 3

(f) Cluster 4

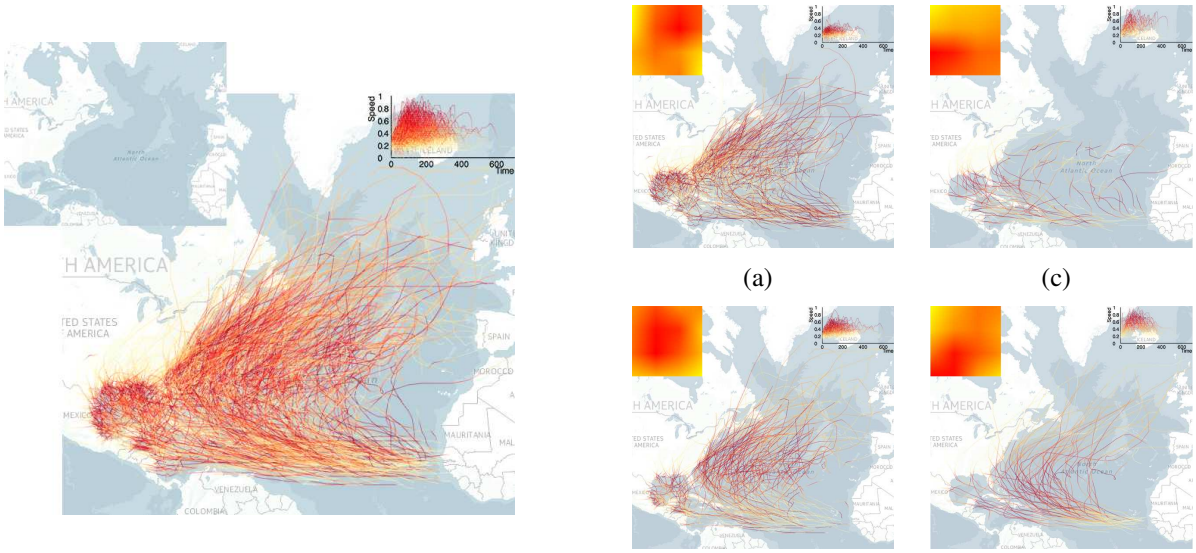
Source: Author.

6.2 Hurricane Dataset

The hurricane dataset HURDAT2 is maintained by the National Hurricane Center (NHC) (HURDAT, 2012). It is a tracking of hurricanes at 6-hourly intervals, containing geographic position (latitude and longitude), maximum sustained surface wind speeds and minimum sea-level pressure. We consider only a subset of 931 Atlantic tropical storms between 1950 and 2014 because the interval between 1861 and 1949 has none or incorrect information about wind speeds.

The VFKM approach (FERREIRA et al., 2013a) successfully used this dataset to illustrate how it can separate different subsets of hurricanes based on their trajectories. In this work, we show how attributes associated with the hurricanes trajectories, such as speed, can further subdivide classes of similar trajectories. In Figure 6.4, AFKM separates the hurricanes by their

Figure 6.4 – Left: 931 Atlantic tropical storms from the HURDAT2 dataset, with trajectories color-coded by their relative speed. Right: AFKM four clusters using speed as input, showing their respective attribute field and speed chart. Cluster (a) has the slowest trajectories and an attribute field with the highest intensity at upper-right. The higher values in the attribute field of the cluster (b) appear in the center of the region. Cluster (c) contains chaotic high-speed trajectories while (d) contains high-speed Cape Verde-type cyclones turning back to the Atlantic Ocean.



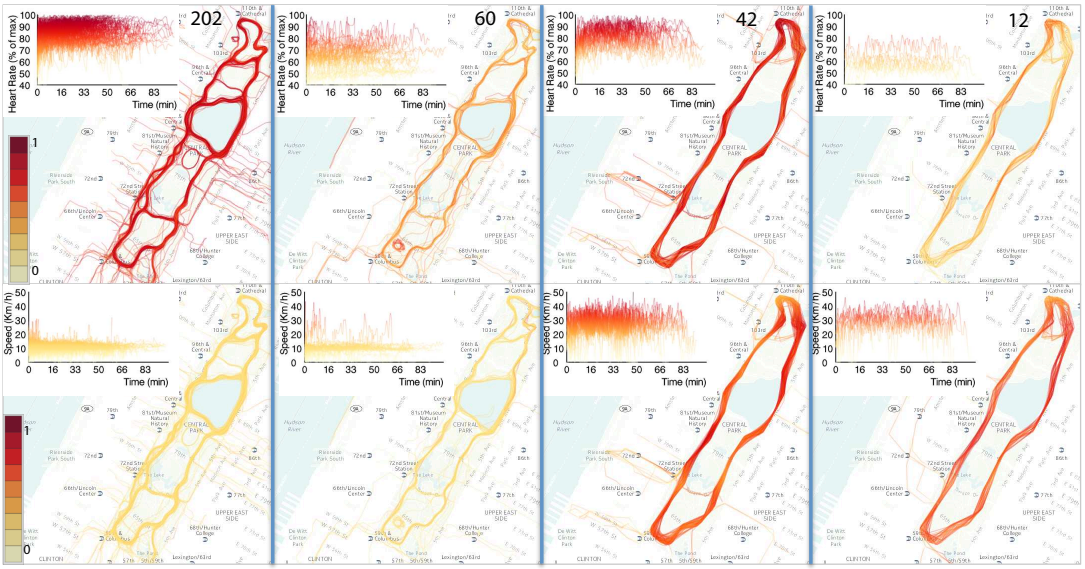
Source: Author.

speed distribution in space. The two clusters with the smallest speed contain most of the storms developed in the Gulf of Mexico, and the highest intensity in the middle and top of the field. One of the clusters with high speed contains storms dissipating either in the coast or the Atlantic Ocean. The last cluster contains the storms developed in Cape Verde turning back to the Atlantic Oceans and losing speed.

6.3 Exercise Dataset

Heart-rate monitors offer an easy way to collect data during exercise activities. The exercise dataset comprises of activities collected using Garmin's monitors, which are publicly available through Garmin Connect web site(GARMIN, 2015). Each activity is described using an XML format called TCX, describing the following time series of data: latitude, longitude, heart rate, speed, altitude, elevation gain, among others. We collected several activities and organized them internally into a MySql database, in which we can query according to our needs. The data originally collected is noisy, and simple data cleaning was performed to remove obvious incorrect sensor information, as described in Section 6.3.2.

Figure 6.5 – Clustering 316 trajectories corresponding to workouts in New York's Central Park using AFKM. The trajectories are color-mapped linearly from yellow to red, according to the color legend at left. In this illustration, the colors are mapped between the minimal and maximum heart rate values and speed, respectively in the first and second row. We choose four clusters using heart rate (MHR) and speed as attributes of the trajectories. The resulting clusters are shown from left to right in decreasing order of trajectory count (202, 60, 42 and 12), and trajectories are color mapped with respect to heart rate (first row) and speed (second row). The predominant cluster corresponds to people performing exercises at 75-95% of MHR, and speed between 8-14km/h, which often corresponds to running. The second cluster has similar speed intervals, but lower MHR (between 65-80%), which corresponds to people running with higher fitness levels or walking. The third cluster has persons with high MHR but also high speed (above 20Km/h), which often corresponds to bicyclists. Here, the trajectories are restricted to the outer loop of Central Park. The final cluster is also associated to bikers, with high speeds, but lower MHR, often related to recreational bikers.



Source: Author.

6.3.1 Data Acquisition

Garmin Connect (GARMIN, 2015) offers a vast amount of activities, uploaded by users, in Training Center XML files (TCX), a format similar to XML. TCX files provide, besides GPS tracks, standards for transferring heart rate, cadence, calories in the track. However, Garmin Connect provides only information about a single activity at a time. Due to this, we developed a web crawler, i. e., an Internet bot that systematically browses the Garmin Connect page to gather the information.

We developed the crawler using Selenium (SELENIUM, 2015), a Java-based programming language for automating web applications. Our crawler login into Garmin Connect and, after, travels across it downloading the TCX files and extra statistics available, parsing those activities and saving in a local MySQL database.

6.3.2 Preprocessing

Due to the inconsistencies of the activities, we made a simple preprocessing phase on the entire dataset, removing obvious incorrect information. We use a set of filters to achieve this. Firstly, we cleaned up the GPS data, removing errors caused by GPS. In step 2, a linear interpolation on missing trackpoints was applied. Next, we compute, from each time series, useful information used by the tool. Finally, we sampled each trajectory.

6.3.2.1 GPS Data Cleaning

GPS data is, essentially, noisy. Filters F1, F2, and F3 remove erroneous data with respect to time and distance. The problems manifest as two records with identical time stamps or a long distance/time between two records.

F1: Duplicated Time Stamp One activity may contain trackpoints with the same time stamp. If it occurs, we only keep the first occurrence and remove the duplicated trackpoints.

F2: Long Distance If the distance between two consecutive trackpoints is bigger than 1 Km, we split the trajectory into two parts. This error can be caused by a delay at GPS to synchronize, and the first few trackpoints contain old GPS coordinates. If it was the case, the first part is removed in filter F6.

F3: Long Time We remove time series containing time intervals over 1 min. As we want to keep only time series that contain detailed information, intervals over that period could lose valuable information about the exercise.

6.3.2.2 Completing Missing values

Some attributes can be missing or empty due to problems with the device. In such cases, F4 recomputes the missing information.

F4: Missing Values If an attribute is missing, we apply a linear interpolation and update the trackpoint. As the time intervals usually are just a few seconds distant from each other and, taking into account that variations in attributes for short periods are smooth, we made a simple linear interpolation.

6.3.2.3 Calculating Essential Informations

We also compute some statistics from the trajectories to be used by our prototype.

F5: Additional Information Some simple statistics, as the mean of each attribute, are

used by our prototype to enable filtering. Another significant extra value is the number of sessions in the week; For each exercise, we calculate the number of sessions that the user made in the last seven days. This information is useful because we can associate the number of sessions with the heart rate or speed, and try to extract insights about theses associations.

6.3.2.4 Tiny Trajectories

We noted that a few exercises contain only a few samples, or the duration and distance are very short. In such cases, we apply filter F6.

F6: Tiny Trajectories If the trajectory contains less than 30 trackpoints, the distance is shorter than 1 km, or the time total is less than 1 min, we remove the trajectory.

6.3.2.5 Sampling trajectories

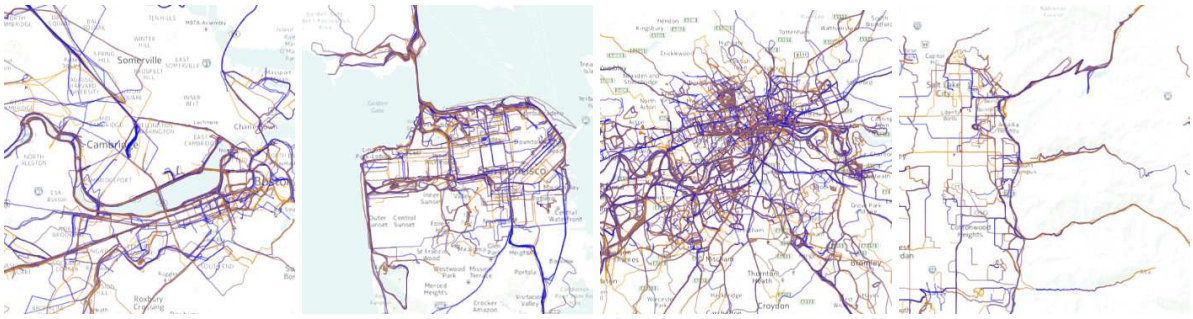
To speedup performance during the clustering analysis of this dataset, we experimented with smaller versions of each trajectory using piecewise aggregate reduction (PAA) (KEOGH et al., 2001). Since we observe not much difference in the clustering results, in the exercises datasets, we opt to reduce the resolution of each trajectory to up to 100 points. The choice was made by dividing each activity in a fixed number of segments, allowing us to compare trajectories easily with different lengths. Therefore, we can compare races by segments and may aggregate users by similarities in the beginning, middle or end of races. Using PAA, we reduce the number of trackpoints from 500M to 45M.

6.3.3 Analysis

We focused our analysis into separate datasets comprising different cities in the world, such as: Boston, San Francisco, Chicago, Salt Lake City, and London, shown in Figure 6.6, as well as the New York Central Park shown in Figure 6.5.

We performed several clustering analysis over the exercise dataset. Our implementation can compute a hierarchical clustering of the dataset, informing at each node of the hierarchy the type of clustering algorithm to use (AFKM, VFKM or K-means), as well as the number of clusters and varying clustering parameters. For the case of AFKM and K-means, it is possible to enumerate the attributes to be used in the clustering, while VFKM uses fixed latitude and longitude. The resulting trajectories can be color-coded about any of the attributes involved, using several color schemes. In this work, we display results using a sequential color scheme

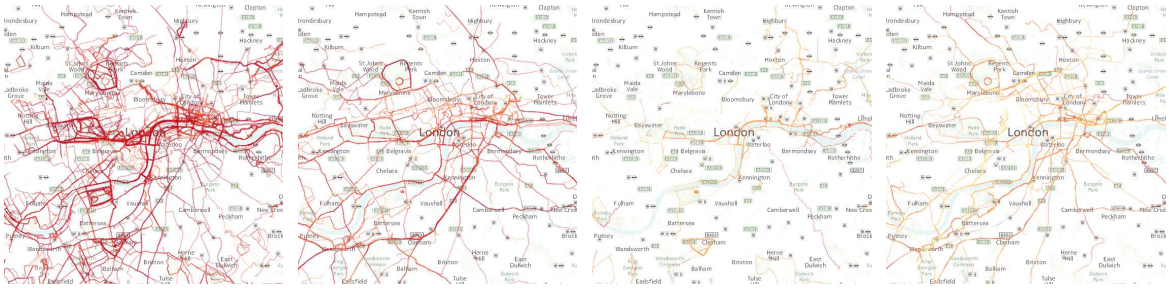
Figure 6.6 – Exercise dataset. Distinct cities in the world and their exercise trajectories, color-coded from start (blue) to end time (yellow).



(a) Boston (b) San Francisco (c) London (d) Salt Lake City

Source: Author.

Figure 6.7 – Four clusters obtained from the London dataset. Trajectories are color-coded using the heart-rate series. The clusters associated with low speed reveal traditional running spots, such as the Thames river banks and parks. High-speed clusters reveal preferred biking trajectories.



(a) High Effort, Low Speed (b) High Effort, High Speed (c) Low Effort, Low Speed (d) Low Effort, High Speed

Source: Author.

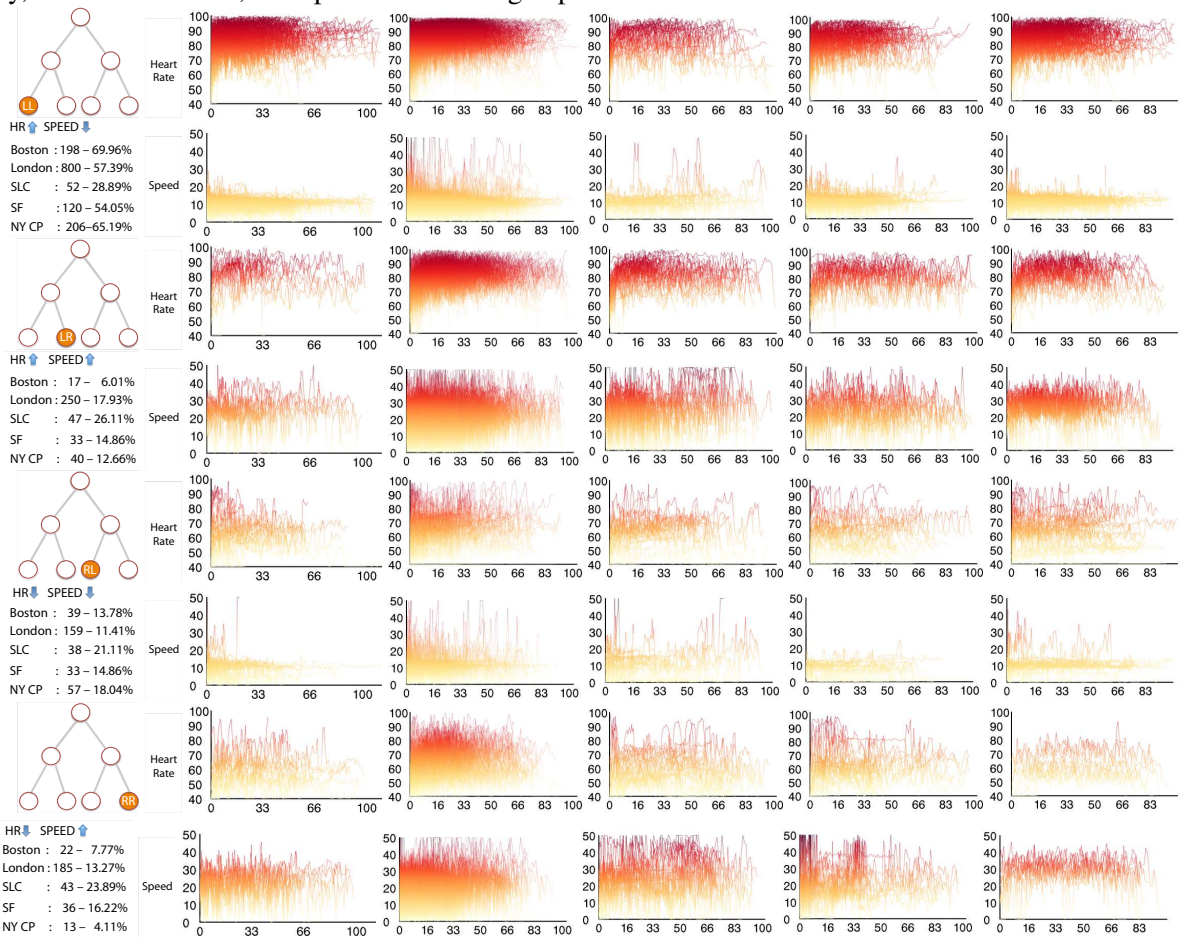
from yellow to red to display attributes, and from blue to yellow to show time.

In Figure 6.7 we used the full resolution dataset for the city of London, comprising activities of up to 100 minutes. Clustering was performed using both heart rate and speed, with the number of clusters set to 4. The four resulting clusters are displayed, using the heart-rate to color code the trajectories. In Figure 6.8 we present the results of computing a 2-level hierarchy using AFKM for activities of up to 100 minutes using the lower-resolution datasets. In the first level, we created two clusters using the heart rate time series. Each of the clusters was further partitioned using speed. We display the plots of heart rate and speed with respect to time for each cluster created.

6.4 Sketching

We provide a sketch-based interface to allow an intuitive way for discovering time series patterns. In our interface, the user can sketch a trajectory over the spatial latitude-longitude plot

Figure 6.8 – AFKM two-step clustering for activities up to 100 minutes in 5 cities. In the first step, 2 clusters were requested using the heart rate time series, which partitions the exercises in high and low effort levels. Each group was further partitioned into two clusters using speed, which often separates people exercising at low speeds (e.g. running or walking) from people at high speeds (e.g. biking). As a result, 4 clusters were generated on the second level (LL, LR, RL, RR). For each of these 4 clusters, we display the heart rate and speed time-series, along the cluster size counts and percentage to the total number of trajectories. Its code identifies each cluster (LL, LR, LL, RR) and position in the tree of cluster generation. We observe that NY Central Park and Boston have similar patterns, favoring lower speeds and high-intensity heart rate, often related to running. San Francisco and London are still predominantly associated with running, but have more people at more top speed, often associated to biking. Salt Lake City, on the other hand, have preference for high-speed exercises.



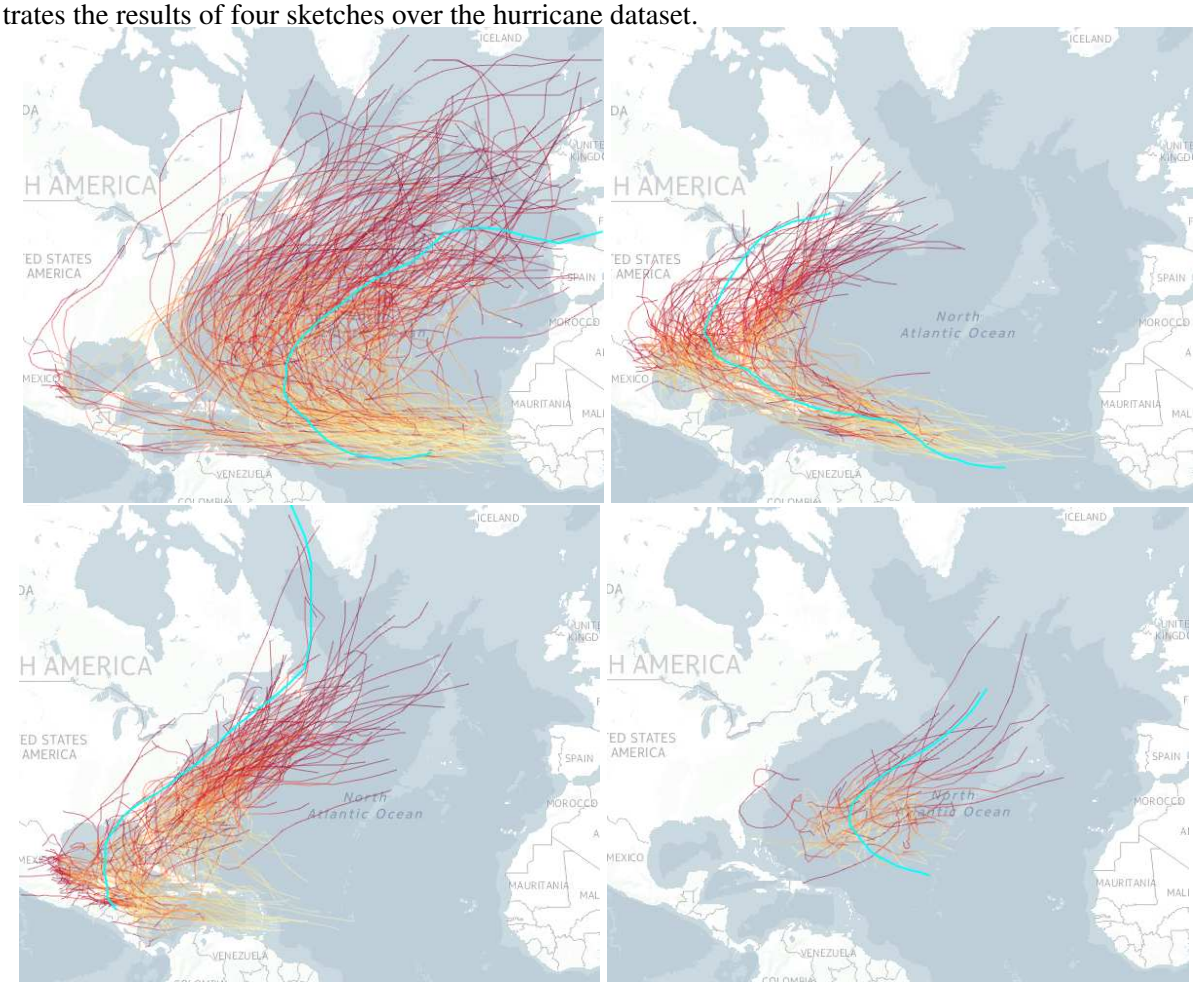
(a) Boston (b) London (c) Salt Lake City (d) San Francisco (e) NY Central Park

Source: Author.

or the attribute-time plot, for any given attributes. AFKM receives the sketch as a parameter and tries finding a cluster that best matches its trajectory. Additionally, one can pick a given trajectory and use as input sketch.

The process works as follows. First, the user draws the desired pattern over the corresponding spatial or attribute plot. For each sketch, a time series is defined by re-sampling and linearly interpolating the sketch. AFKM receives this time series as input, and it serves as the basis for one cluster. This process can be repeated until all desired patterns are defined. Once

Figure 6.9 – Sketching examples. The user draws a sketch trajectory over the latitude-longitude or attribute-time plots, and AFKM computes the best cluster that fits the given sketch. This example illustrates the results of four sketches over the hurricane dataset.



Source: Author.

sketching finishes, each re-sampled time series associated with each sketch is given to AFKM, and the respective attribute field is computed. If the number of requested clusters is bigger than the number of sketches, AFKM performs its default initialization. After, AFKM returns the clusters, including the trajectories associated with each sketch.

In other words, we pass only the sketch (as a trajectory) to AFKM, defining the number of clusters. The clustering returns the corresponding attribute fields. Therefore, we fix each cluster with the attribute fields of a sketch. Finally, AFKM does not recompute the attributes fields, only fits the trajectories with the closest cluster for each one of trajectories.

One thing to notice is that, at the interpolation phase, start and end points from each attribute sketch may differ. As estimate the values missing is a problem, we define all attribute sketches (from one cluster) containing the same start and end point, i.e., points out of the common interval are discarded.

Figure 6.9 illustrates results for the HURDAT2 dataset. Lines in blue represent the user

sketch. We observe that the resulting trajectories in this dataset are similar to the informed sketches. We believe that user feedback seems more intuitive, especially when the user knows what kind of patterns one wants to discover. We defer to Section 7.6 to show more results on sketching over attribute-time plots.

7 DISCUSSION AND COMPARISON

In this chapter, we discuss the alternatives for parameter selection and parameter space, limitations, and advantages of AFKM. Then, we compare our technique with VFKM , *k*-means, and TraClus.

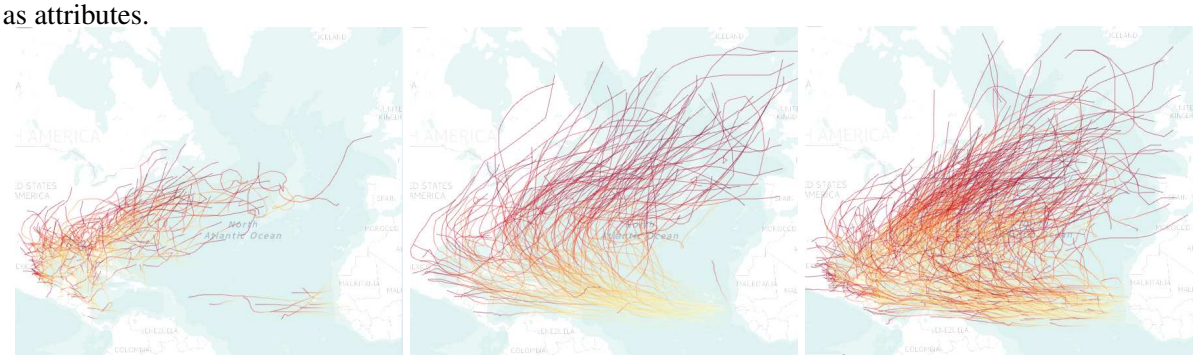
7.1 Parameter Selection

The grid resolution *R* controls the level of detail in the attribute fields. By using smaller values of *R*, larger weights are passed to the eigenvectors of the Laplacian. Thus, fewer details can be detected, and minor features will hold no influence in clustering, capturing global trends especially. On the other hand, by increasing *R*, more complex behavior can be modeled. Selecting the domain for clustering depends on the correlation among attributes. For example, in the synthetic dataset (Figure 6.2), the additional attribute is associated with latitude and longitude attributes. In that situation, the relationship is clear.

Due to the distinct range of values for each attribute, we apply a normalization of the trajectory values before clustering. Thereby, even though the original values may be different, their weights will be the same. However, in certain cases we might want an attribute to have a higher impact in clustering. In such cases, the user can define weights for the desired attributes. We can observe that situation in Figure 7.2. By applying AFKM on the heart rate and speed attributes, we create two clusters (Figure 7.2 (a)). However, there are clearly two groups of trajectories in the speed plot. Thereby, we use a weight 2 for the speed attribute. As a result, AFKM splits the trajectories into higher and lower speeds (Figure 7.2 (b)).

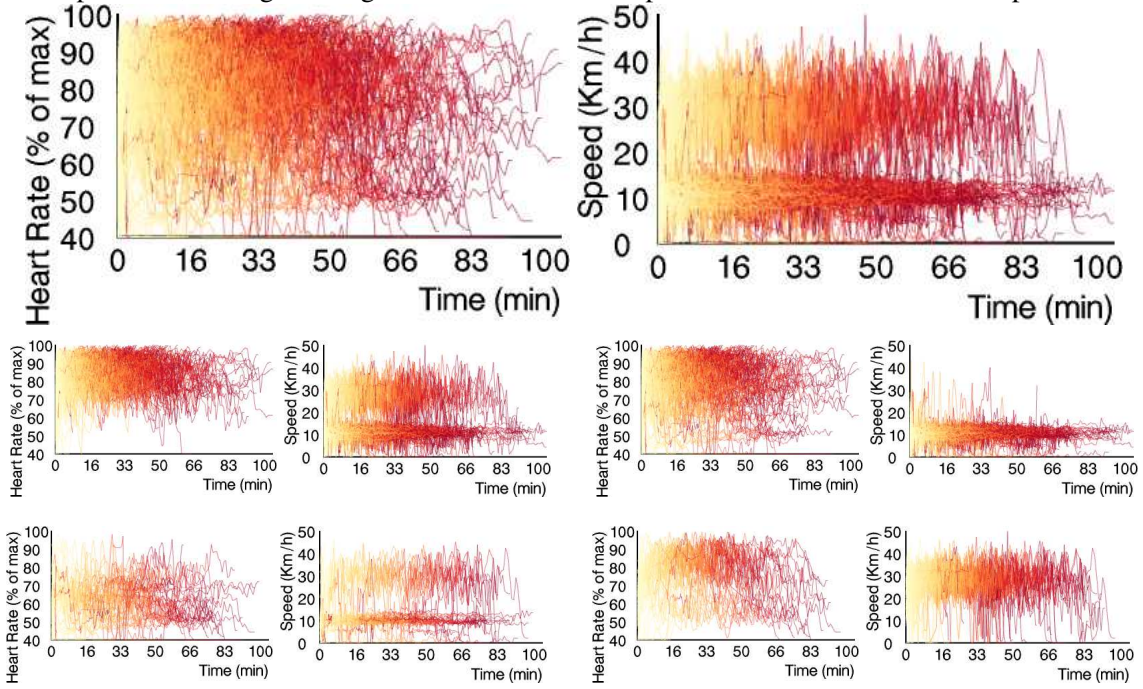
Choosing the right number of clusters for each experiment remains an open problem. Although tedious and a try-and-error process, we believe human analysts can use AFKM as an interactive procedure, even obtaining insights from a distinct number of clusters in the same data. In most of our experiments, we group the datasets into two or four clusters. This choice is essentially due to simplicity. For example, in Figure 7.1, we pick 3 clusters for hurricane dataset. The first cluster (left) contains lower speeds while cluster 1 and 2 are divided mainly by the shape of the trajectory.

Figure 7.1 – We cluster hurricane dataset with $k = 3$, using tangents for latitude and longitude and speed as attributes.



Source: Author.

Figure 7.2 – Attributes with different weights. Top: Heart rate and speed plots for Central Park dataset. We can observe two distinct patterns at the speed attribute (lower and higher). (a) Default clustering with weight 1 in the heart rate and speed attributes, generating two clusters without separation of the detected pattern. (b) By changing the speed weight to 2, AFKM separates into lower and higher speed attributes. In this experiment, the higher weight associated with the speed attribute allows a better separation.



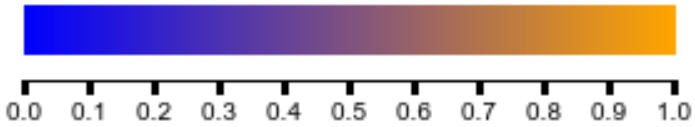
(a) Uniform weights = 1 (b) Speed weight = 2

Source: Author.

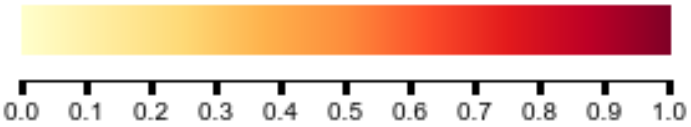
7.2 Hierarchical Clustering

An important aspect is the specification of the number of clusters. As we are not concerned with finding the optimal number of clusters, we suggest an interactive procedure that allows clustering to be applied hierarchically. The user can reuse previous clusters, and apply further clustering against that subset of trajectories, even using different attributes and number of attributes.

Figure 7.3 – Figure displaying the color palettes used in this entire thesis. (a) Linear interpolation from blue to orange. (b) Linear interpolation from yellow to red.



(a) Blue to Orange



(b) Yellow to Red

Source: Author.

For example, in Figure 6.8, we first generated 2 clusters using the heart rate attribute. Then, we selected the resulting cluster and created 2 clusters using speed. Also, the user can specify a maximum number of trajectories for clustering. Imagine a scenario where a dataset has hundreds of thousands or even millions of trajectories. The clustering of the entire dataset can be computationally expensive and time-consuming. In such cases, instead of selecting all data, we can randomly pick the maximum number trajectories, and compute their clustering. Finally, scalar fields of each generated cluster are utilized to fit the remaining trajectories to the clusters obtained.

7.3 Mapping Attributes to Visual Properties

Trajectories are composed of data items containing at least three attributes: latitude, longitude, and time. In practice, there are additional attributes, and even derived attributes can be computed, such as speed. Although AFKM can handle any number of attributes, for clarity and simplicity, we choose to present individual attribute-time plots. When analyzing latitude and longitude, we use an interactive map as background.

The user can map attributes into color using pre-defined color mappings, and use this mapping to alter the color of each point in a given trajectory. Figure 7.3 displays the two color palettes applied to render trajectories in the images of this thesis. Color mapping can be relative to the trajectory or absolute for the entire dataset, i.e., we can analyze the behavior of an attribute in a trajectory relative to itself, or considering the boundaries of the whole dataset. Using color mapping, we can visually inspect how attributes vary in different trajectory parts, as presented in Figure 7.4.

Figure 7.4 – Using color mapping, we can visually discern spatial features in our clusters, as areas with high speed (b) inversely correlated to areas with high elevation (c).

(a) Heart Rate (b) Speed (c) Elevation

Source: Author.

7.4 Limitations

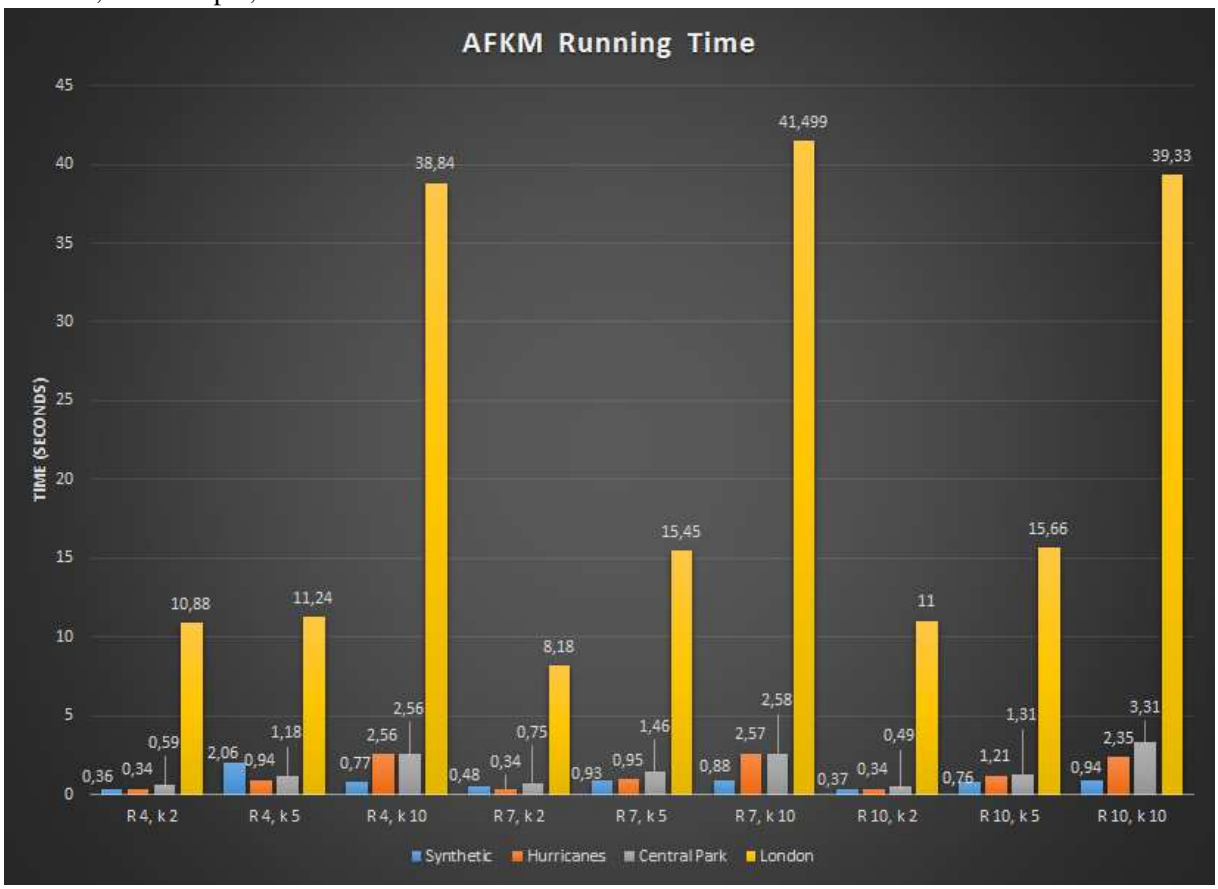
Even though AFKM can handle any number of attributes, our framework shows attributes always in 2D attribute-time plots, or longitude/latitude plots with underlying maps. Another issue is the choice of initial clusters. Depending on the first choice of AFKM, we may obtain some entirely different clusters. In addition, initial attributes fields for subsequent clusters are created by picking the time series with the worst error compared with previous clusters, or fully random. Thus, we offer no guarantees that our result is close to the global optimum.

As the result may be affected by the first choice of AFKM, obtain insights from the dataset can lead to an iterative try-and-error process. As the user needs to select both the number of clusters and the attributes for clustering, this approach can be time-consuming and error-prone, may leading to missing interesting features, supposing the user neglect some combination of attributes.

7.5 Running Times

We now report the results of running AFKM in distinct datasets with distinct parameter configuration. We start with small grid resolution ($R = 4$) and number of clusters ($k = 2$), and increasing them (until both reach 10). This experiment is used to demonstrate how AFKM behave by changing the parameter choice for distinct datasets. As we can note, normally, by increasing R and k, the time spent to clustering always increases.

Figure 7.5 – Image presents the time spent to clustering in four distinct datasets: Synthetic, Hurricanes, Central Park, and London. We ran AFKM for each one of those datasets changing the number of clusters (k) and the grid resolution (R) parameters. By increasing the grid resolution, AFKM tends to spend more time to generate clusters, due to the more details being captured. It is not always true, as we can note in London, for example, between $R = 7$ and $R = 10$ with 10 clusters.

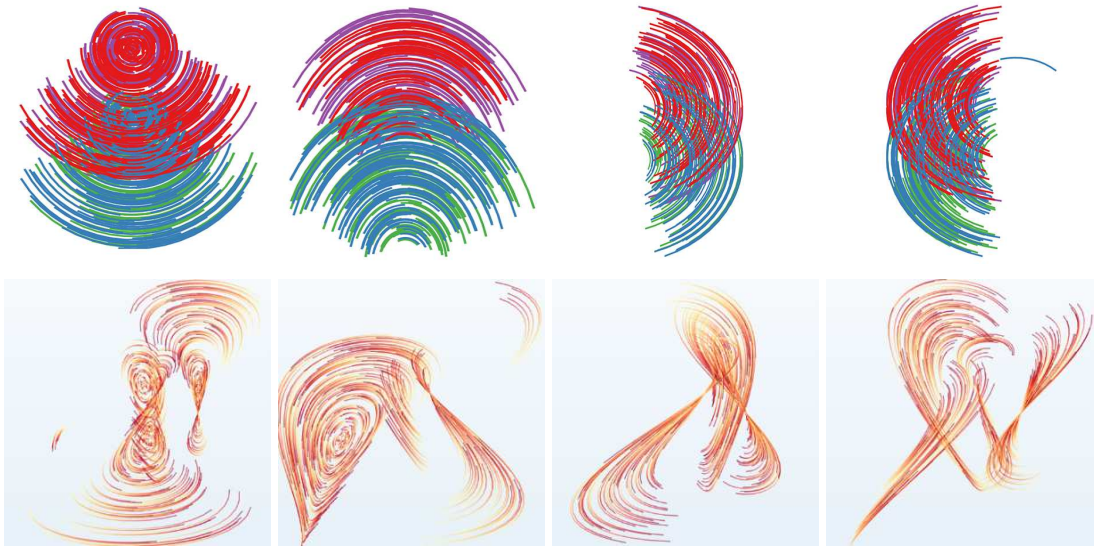


Source: Author.

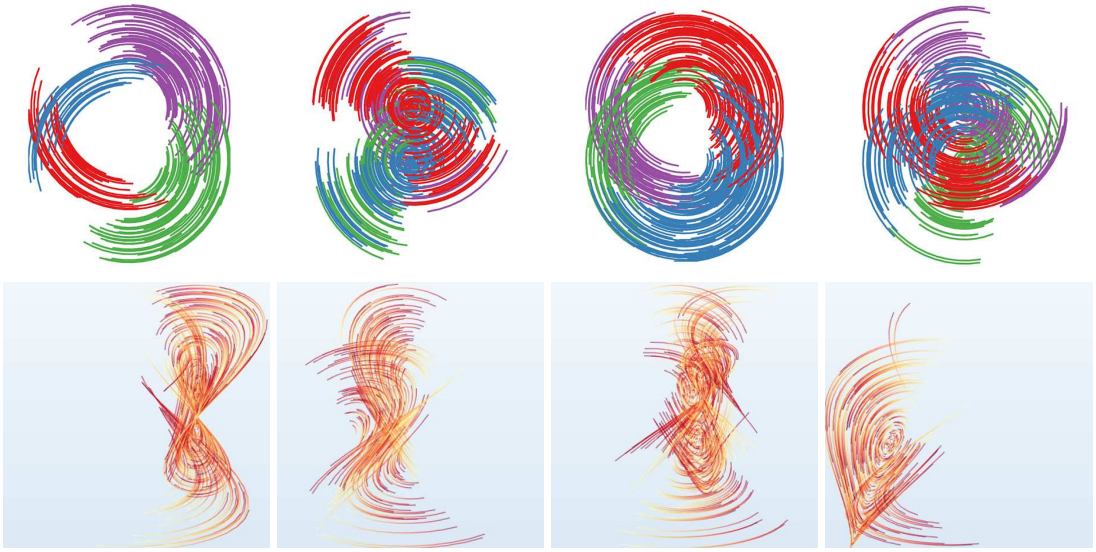
Figure 7.5 presents the running time for the datasets. For the synthetic dataset, composed by 2000 trajectories, AFKM spent less than 1 second to clustering (only exception was using $R = 4$, and $k = 5$). HURDAT2 (931 trajectories) and Central Park (316), take no more than 3 and 6 seconds, respectively. In contrast, London, with 1550 trajectories, takes between 8 and 42 seconds to generate clusters.

Accordingly with those experiments, we can notice (as expected) that, by increasing the number of trajectories, the number of clusters and the level of detail (grid resolution), the time spent by AFKM can increase considerably. Thus, limiting the number of trajectories, as mentioned in Section 7.2, can be a way out to workaround this problem.

Figure 7.6 – VFKM and K-means clustering for synthetic datasets described in Figure 6.2 and Figure 6.3. (a) VFKM handles only trajectory attributes (latitude and longitude), ignoring other attributes and failing to detect those patterns. (b) K-means use the mean of the generated attribute, and also does not detect distinguishable patterns.



(a) VFKM clusters



(b) K-means clusters

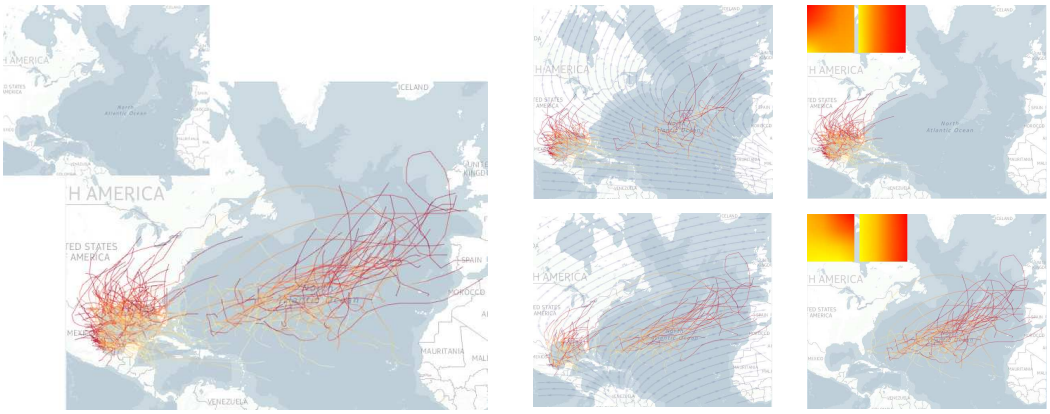
Source: Author.

7.6 Comparison

7.6.1 Comparison against *k*-means and VFKM

In this section, we compare the clustering results of AFKM against K-means and VFKM (FERREIRA et al., 2013a). The comparison against K-means investigates how the handling of trajectory attributes introduced by AFKM improves on the clustering using the average of the attributes used by K-means. Similarly, the comparison against VFKM establishes how attributes

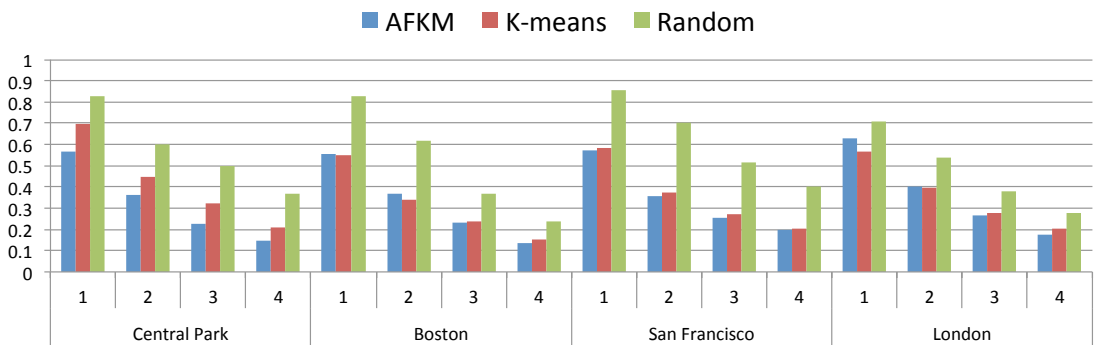
Figure 7.7 – Comparing AFKM with VFKM. Trajectories are color-coded from start (yellow) to end time (red). Left: subset of 264 trajectories from HURDAT2 dataset. We can visually discern two spatial patterns; one in the Gulf of Mexico and another in the middle of the Atlantic Ocean. Right: Resulting clusters using VFKM and AFKM, respectively. (a) Using latitude and longitude attributes, VFKM tries fitting trajectories only as movements, being unable to separate both spatial groups. (b) Otherwise, AFKM recognizes those patterns, as we can see by the two generated clusters and their corresponding scalar fields relative to longitude and latitude.



(a) VFKM (b) AFKM

Source: Author.

Figure 7.8 – Evaluation of the spatial distribution variance of attributes using the implicit clutter measure. For each dataset, we perform a hierarchical clustering using a branching factor of two, and report the implicit clutter at levels 1, 2, 3 and 4 of the tree. We observe that AFKM produces smaller errors at the higher levels of the tree. However, in general, the errors of K-means and AFKM are comparable.



Source: Author.

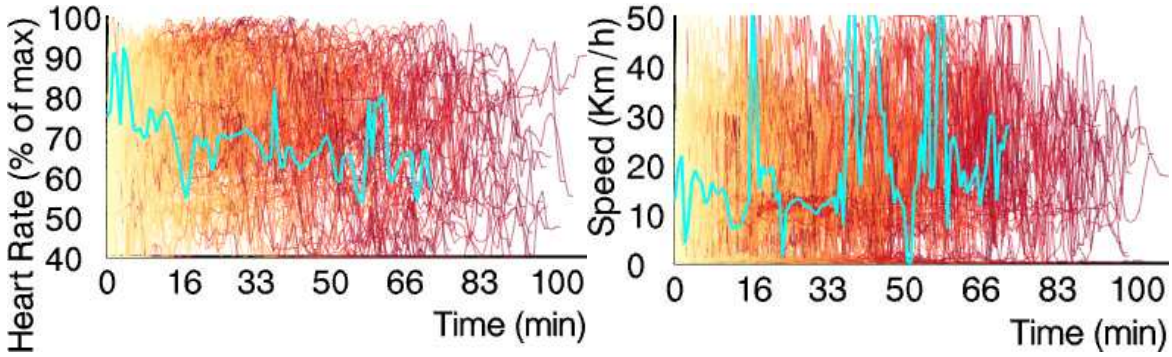
improve cluster separation. We used the C++ implementation of VFKM[1], and implemented a basic K-means algorithm using the average of attribute values.

The first comparison uses the synthetic datasets presented in Section 6.1. As can be seen in Figure 7.6, both K-means and VFKM are not able to perform the cluster separation given by AFKM in Figures 6.2 and 6.3. In both datasets, there is a clear separation provided by the attributes, which can not be leveraged by the average computation of K-means or lack of attribute analysis of VFKM.

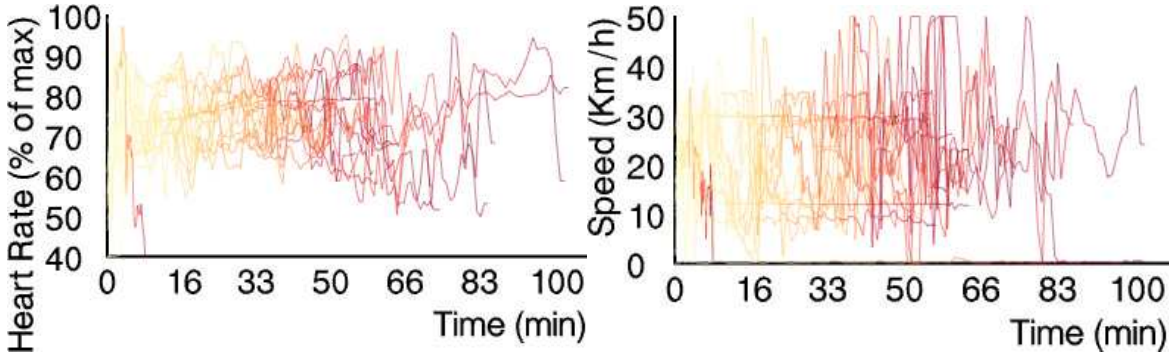
The second comparison is specific to VFKM. We used a subset of 264 trajectories from

[1]https://github.com/nivan/kmeans

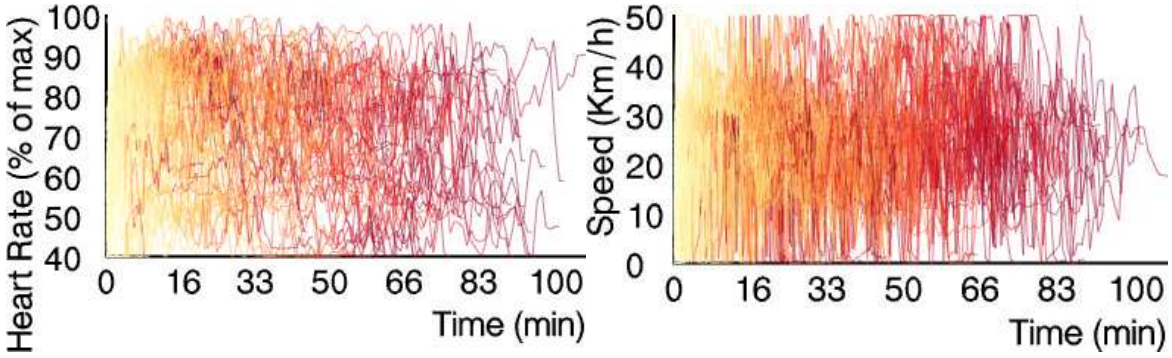
Figure 7.9 – AFKM and K-means sketching comparison. (a) subset of 184 exercises at Salt Lake City, displaying Heart Rate and Speed attributes. One trajectory is chosen as the basis for the sketch process (sketch line in blue). (b) We notice the cluster generated by AFKM approximates well the noisy sketch. (c) Otherwise, due to K-means simplification of trajectory attributes as mean, K-means groups a large subset of trajectories.



(a) Heart Rate and Speed attributes of 184 exercises with the respective sketches (lines in blue).



(b) AFKM cluster obtained from sketch.



(c) K-means cluster obtained from sketch.

Source: Author.

HURDAT2(Figure 7.7). We observe two distinct spatial clusters, showing storms being developed in the Gulf of Mexico (Figure 7.7 (b) top) and middle of the Atlantic Ocean (Figure 7.7 (b) bottom). While AFKM separates both spatial groups (Figure 7.7 (b)), VFKM only considers trajectories, resulting in clusters with similar paths, but spatially distinct (Figure 7.7 (a)).

The third comparison, and more intriguing one, uses the exercise dataset. During the analysis of this dataset, we noticed that AFKM appeared to separate the attributes initially by

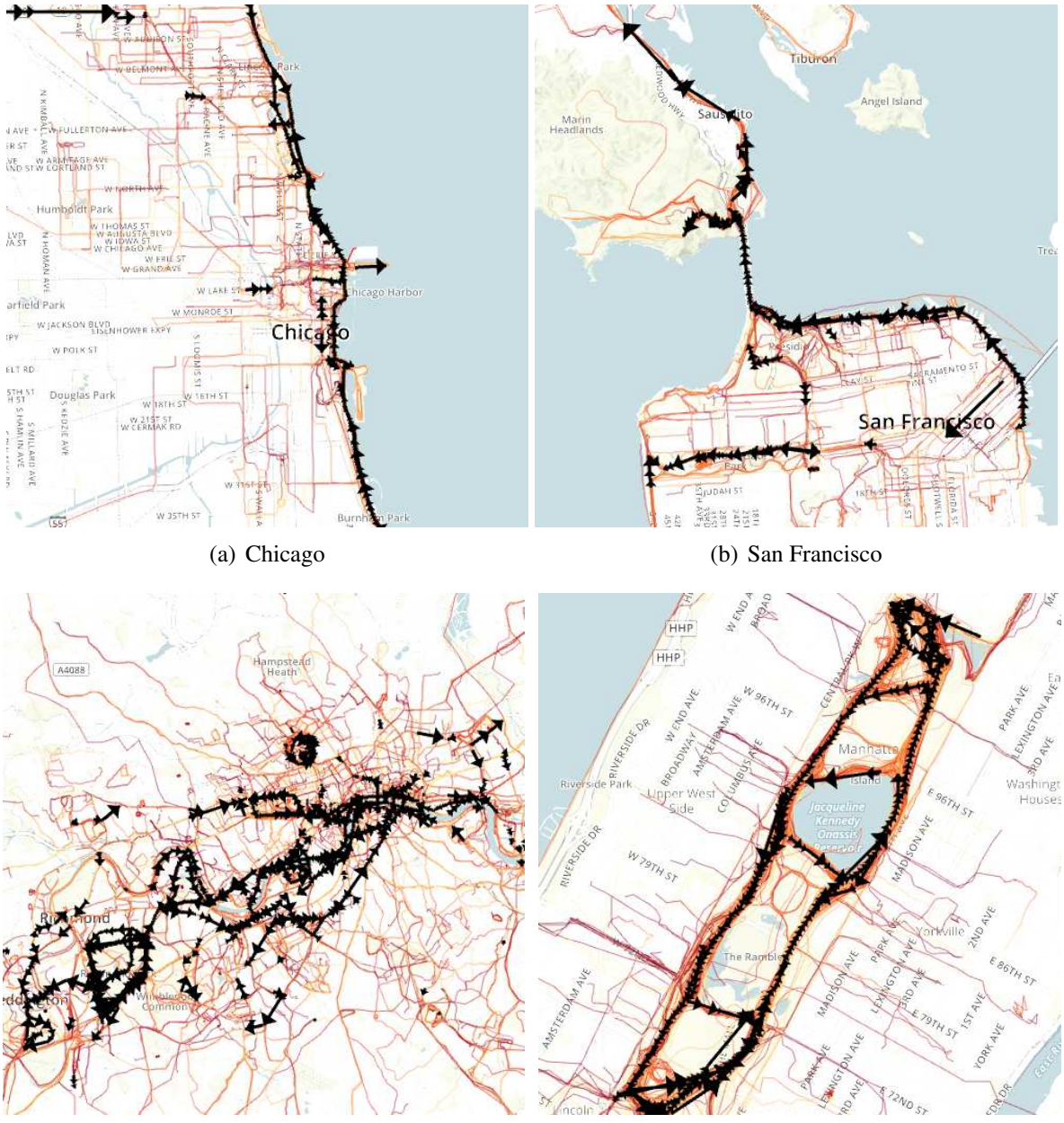
their means. By analyzing the attributes of the exercise dataset, it is clear that those attributes (especially heart rate) have minor variations over time during exercise. Therefore, the time-series have similar behavior to the mean, and makes sense to compare against the K-means algorithm.

One limitation of this analysis is that the problem of clustering evaluation is not well defined. Different approaches are designed to capture different aspects of the data. Therefore, one needs to fix one evaluation measure to compare results of different techniques. In this work, we propose a simple measure based on the creation of a spatial 2D-histogram (or heatmap) of the attributes distribution. In more details, given a trajectory dataset it is common to try to visualize its attributes through heatmaps (WILLEMS; WETERING; WIJK, 2009). While intuitive, this strategy has a major shortcoming due to the overlap of trajectories with different attribute values. We quantify this notion of attribute value overlap by taking discrete (equally spaced) points over the trajectory and mapping those points to the cells of a 2D histogram. For each cell, we compute the variance of the attribute values and call the sum of variances for all the cells *implicit clutter*. We use this notion to evaluate trajectory cluster algorithms. Algorithms that can produce smaller implicit clutter, can produce better attribute maps, i.e., reduces clutter.

In Figure 7.8, we report the results we obtained by performing a hierarchical clustering with branching factor of 2, until 4 levels in the tree. We computed the *implicit clutter* measure for all nodes in the tree. To evaluate how the measure decreases at different levels of the tree, we accumulated the metric values for each level (1-4). Errors are normalized across different datasets with respect to their highest error to allow comparison. The results are given for AFKM, K-means and a random algorithm that simply selects a random attribute value. We observe that AFKM always have the smallest implicit clutter measure at the deepest level of the tree, although the difference to K-means is not significative, due probably to the nature of the data.

Another way to compare AFKM and K-means is through the sketching procedure. In Figure 7.9 we illustrate the ability to perform the sketching over the attributte-time plot. In this case, we selected one given trajectory of the input dataset to serve as the sketch, shown in blue, for both heart rate and speed trajectories. As expected, AFKM produces a cluster that more faithfully adapts to the input sketch, while K-means includes additional trajectories that have similar average values to the input sketches. This result shows the generality of AFKM and ability to adjust to the variation in the trajectories attributes.

Figure 7.10 – The image presents TraClus clusters for the exercise dataset in four distinct cities: Chicago, San Francisco, London, and New York (Central Park). Cluster representative trajectories are in black. The number of discovered clusters for each city is high. As the nature of TraClus, it detect local features, grouping the clusters by similar sub-paths. (a) 97 clusters are found in Chicago with $\varepsilon = 40$ and *MinLns*$= 4$ in 16 seconds. (b) 152 clusters in San Francisco with $\varepsilon = 40$ and *MinLns*$= 3$ discovered in 5 seconds. (c) In London, using $\varepsilon = 30$ and *MinLns*$= 3$, TraClus discovers 782 clusters in 150 seconds. (d) For the Central Park, in New York, 37 clusters are found with $\varepsilon = 29$ and *MinLns*$= 8$ in 42 seconds.

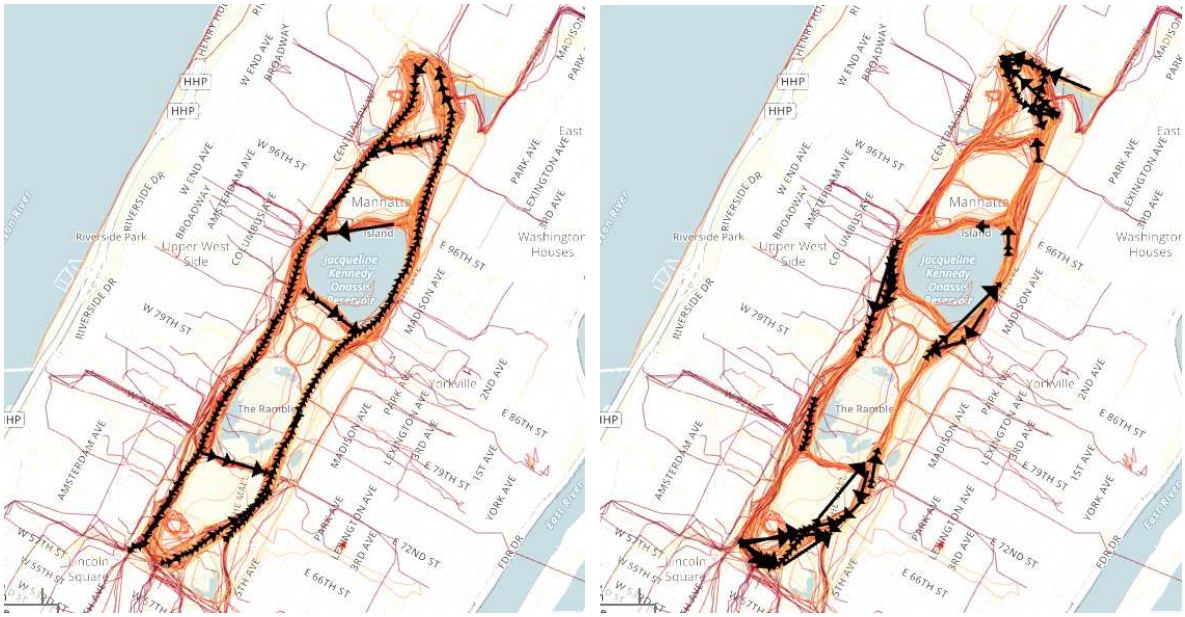


(a) Chicago (b) San Francisco

(c) London (d) New York Central Park

Source: Author.

Figure 7.11 – Figure shows de clusters found by TraClus in the central park dataset. We split the clusters in two images. On the left (a), we display the 7 most relevant clusters discovered by TraClus, composed by trajectories in both sides of the outer loop, and by trajectories crossing the park. On the rigth (b), 30 small clusters discovered by TraClus, located mainly at the north and south side of the park.



(a) (b)

Source: Author.

7.6.2 Comparison against TraClus

We also perform a comparison with TraClus, by Lee et al. (LEE; HAN; WHANG, 2007), using their C++ implementation[2]. TraClus is one of the main references in trajectory clustering. It consists of a simplification of trajectories into line segments, and the clustering of those segments. The algorithm contains two main parameters: ε as a distance threshold defining the neighborhoods for each segment, and *MinLns* used to find neighborhoods that define clusters.

In the following experiments, we use their heuristic to select the parameter values for clustering. As the patterns mined by AFKM and TraClus are distinct, it is difficult to compare both techniques, saying that one method is better than the other. However, we investigate the differences in their recovered patterns, showing what kind of pattern TraClus detects and AFKM does not, as the opposite.

Unlike AFKM, TraClus does not incorporate attribute information, and all the attributes in trajectories are lost. For this reason, results for the synthetic and hurricane datasets will not differ from those demonstrated by Ferreira et al. (FERREIRA et al., 2013a). So, TraClus is unable to detect both circular patterns (Figure 6.1), or the generated peaks (Figure 6.2 and

[2]http://dm.kaist.ac.kr/jaegil/#Publications

Figure 6.3) for the synthetic datasets. Traclus also fails to identify the Cape Verde-type cyclones turning back from Atlantic Ocean (Figure 6.2 (d)).

TraClus discovers a large number of clusters for each exercise dataset. Figure 7.10 displays the TraClus clusters for four exercises datasets: Chicago, San Francisco, London, and Central Park (New York), taken, respectively, 16, 5, 150, and 42 seconds to execute . As we can observe, the clusters are mainly located in areas with large number of activities, as the coasts in Chicago and San Francisco, as well as close to the River Thames and parks in London.

In New York, our dataset contains only trajectories at the Central Park. TraClus detects cluster on both sides of the outer loop of the Central Park, as well small clusters composed by trajectories crossing the park (represented by the street traverses), as demonstrade by Figure 7.11 (a). AFKM handles trajectories as a whole, detecting the inner loops of the Central Park, but not only the crossing as TraClus does. TraClus also detects a large number of small clusters, as presented by Figure 7.11 (b). They are located specially at both sides of the central Park (north and south), and close to the inner loop.

8 CONCLUSIONS AND FUTURE WORK

We have introduced a novel trajectory cluster technique that generalizes VFKM. Our method, called AFKM, allows clustering of any type of time series and number of attributes. AFKM preserves the advantages of VFKM, such as the catching of global patterns that are not apparent at the local level in noisy trajectories, and its highly parallelization, due to each attribute field is computed independently. We propose an interactive framework that enables heterogeneous clusters, i.e., each level of clustering could be generated by different attributes, besides allowing guided clustering (by users input). The algorithm opens many possibilities of exploratory analysis and visualization tools.

One fascinating feature for user feedback is returning the trajectory that best approximates the attributes fields from a cluster generated by AFKM. This is a procedure similarly inverse of sketching. Instead of the user draws a sketch to obtain one cluster, from the generated cluster, the user gets in return a trajectory that can generates that cluster. It could help meaningly in understanding the results from each attribute field, and an interesting avenue for future work. In addition, the sketches on two dimensions (like on the map) could be done not just by drawing a trajectory, but also drawing the field directly. For example, a user could draw the field of speed or heart rate directly on the map. It could provides a more intuitive way of sketching.

Future works on the prototype refer to optimize the webgl scripts to render and manipulate the trajectories. These improvements can contribute to a better user experience using the prototype. We also want to allow users to upload their own trajectories, without being limited by the trajectories presented in this thesis. Therefore, they can manipulate, create clusters, and obtain insights from their own data using AFKM.

Once each attribute is computed independently, we could easily take advantage of this characteristic of AFKM and make it parallelizable. We did not measure the times for all experiments, but in most of them, it takes just a few seconds do generate and return the clusters. Working with a parallel version of AFKM may bring the response of most clustering to less than a second. Of course, considering the user configuration, as also explained in Section 7.1

REFERENCES

ADRIENKO, N.; ADRIENKO, G. Spatial generalization and aggregation of massive movement data. **Visualization and Computer Graphics, IEEE Transactions on**, IEEE, v. 17, n. 2, p. 205–219, 2011.

AGAFONKIN, V. Leaflet. 2015. Available from Internet: <http://leafletjs.com/>. Accessed in: 2016-03-01.

ANDRIENKO, G.; ANDRIENKO, N. Spatio-temporal aggregation for visual analysis of movements. In: IEEE SYMPOSIUM ON VISUAL ANALYTICS SCIENCE AND TECHNOLOGY, 2008. VAST'08. **Proceedings...** Columbus, Ohio, USA, 2008. p. 51–58.

ANDRIENKO, G. et al. A conceptual framework and taxonomy of techniques for analyzing movement. **J. Vis. Lang. Comput.**, Academic Press, Inc., Orlando, FL, USA, v. 22, n. 3, p. 213–232, jun. 2011. ISSN 1045-926X. Available from Internet: <http://dx.doi.org/10.1016/j.jvlc.2011.02.003>.

ANDRIENKO, G. et al. **Visual analytics of movement**. New York, NY, USA: Springer Science & Business Media, 2013.

ANDRIENKO, G. et al. Interactive visual clustering of large collections of trajectories. In: IEEE SYMPOSIUM ON VISUAL ANALYTICS SCIENCE AND TECHNOLOGY. **Proceedings...** Atlantic City, New Jersey, USA, 2009. p. 3–10.

ANDRIENKO, G. et al. Visualization of trajectory attributes in space-time cube and trajectory wall. In: CARTOGRAPHY FROM POLE TO POLE. **Proceedings...** New York, NY, USA: Springer, 2014. p. 157–163.

ANDRIENKO, G.; ANDRIENKO, N.; WROBEL, S. Visual analytics tools for analysis of movement data. **SIGKDD Explorations**, v. 9, n. 2, p. 38–46, 2007.

ANDRIENKO, N.; ANDRIENKO, G. Spatial generalization and aggregation of massive movement data. **IEEE Trans. Visualization and Computer Graphics**, v. 17, n. 2, p. 205–219, Feb 2011. ISSN 1077-2626.

ANDRIENKO, N.; ANDRIENKO, G. Visual analytics of movement: An overview of methods, tools and procedures. **Information Visualization**, Sage Publications, 2012.

ANDRIENKO, N. et al. Space transformation for understanding group movement. **IEEE Trans. Visualization and Computer Graphics**, v. 19, n. 12, p. 2169–2178, Dec 2013. ISSN 1077-2626.

ANDRIENKO, N.; ANDRIENKO, G.; GATALSKY, P. Exploratory spatio-temporal visualization: an analytical review. **Journal of Visual Languages & Computing**, Elsevier, New York, NY, USA, v. 14, n. 6, p. 503–541, 2003.

ARTHUR, D.; VASSILVITSKII, S. K-means++: The advantages of careful seeding. In: ACM-SIAM SYMPOSIUM ON DISCRETE ALGORITHMS, SOCIETY FOR INDUSTRIAL AND APPLIED MATHEMATICS. **Proceedings...** Philadelphia, PA, USA, 2007. p. 1027–1035. Available from Internet: <http://dl.acm.org/citation.cfm?id=1283383.1283494>.

BOSTOCK, M. D3 data-driven documents. 2015. Available from Internet: <http://d3js.org/>. Accessed in: 2016-03-01.

CABELLO, R. three.js. 2015. Available from Internet: <http://threejs.org/>. Accessed in: 2016-03-01.

FERREIRA, N. et al. Vector field k-means: Clustering trajectories by fitting multiple vector fields. In: COMPUTER GRAPHICS FORUM, WILEY ONLINE LIBRARY. **Proceedings...** Swansea, South Wales, UK, 2013. v. 32, p. 201–210.

FERREIRA, N. et al. Visual exploration of big spatio-temporal urban data: A study of new york city taxi trips. **IEEE Trans. Visualization and Computer Graphics**, v. 19, n. 12, p. 2149–2158, Dec 2013. ISSN 1077-2626.

FISHER, D. et al. Trust me, i'm partially right: incremental visualization lets analysts explore large datasets faster. In: SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, ACM. **Proceedings...** Austin, Texas, USA, 2012. p. 1673–1682.

FOUNDATION, O. Openstreetmap. 2015. Available from Internet: <http://www.openstreetmap.org/>. Accessed in: 2016-03-01.

GARMIN. Garmin connect. 2015. Available from Internet: <http://connect.garmin.com/>. Accessed in: 2016-03-01.

HAN, J.; KAMBER, M.; PEI, J. **Data mining: concepts and techniques: concepts and techniques**. New York, NY, USA: Elsevier, 2011.

HAT, R. Hibernate. 2015. Available from Internet: <http://hibernate.org/>. Accessed in: 2016-03-01.

HURDAT. HURDAT: The national hurricane center's north atlantic hurricane database. Feb 2012. Available from Internet: <http://www.aoml.noaa.gov/hrd/data_sub/re_anal.html>. Accessed in: 2016-03-01.

JAIN, P.; NETRAPALLI, P.; SANGHAVI, S. Low-rank matrix completion using alternating minimization. In: FORTY-FIFTH ANNUAL ACM SYMPOSIUM ON THEORY OF COMPUTING. **Proceedings...** New York, NY, USA, 2013. (STOC '13), p. 665–674. ISBN 978-1-4503-2029-0. Available from Internet: <http://doi.acm.org/10.1145/2488608.2488693>.

JANETZKO, H. et al. Visual abstraction of complex motion patterns. In: IS&T/SPIE ELECTRONIC IMAGING, INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. **Proceedings...** San Diego, California, USA, 2013. p. 90170J–90170J.

JANETZKO, H. et al. Anomaly detection for visual analytics of power consumption data. **Computers & Graphics**, Elsevier, New York, NY, USA, v. 38, p. 27–37, 2014.

JQUERY, j. F. jquery - write less, do more. 2015. Available from Internet: <https://jquery.com/>. Accessed in: 2016-03-01.

KEHRER, J.; HAUSER, H. Visualization and visual analysis of multifaceted scientific data: A survey. **IEEE Trans. Visualization and Computer Graphics**, v. 19, n. 3, p. 495–513, March 2013. ISSN 1077-2626.

KEOGH, E. et al. Dimensionality reduction for fast similarity search in large time series databases. **Knowledge and information Systems**, Springer, New York, NY, USA, v. 3, n. 3, p. 263–286, 2001.

KISILEVICH, S. et al. Spatio-temporal clustering. In: **Data Mining and Knowledge Discovery Handbook**. New York, NY, USA: Springer US, 2010. p. 855–874. ISBN 978-0-387-09822-7. Available from Internet: <http://dx.doi.org/10.1007/978-0-387-09823-4_44>.

KONDO, B.; COLLINS, C. Dimpvis: Exploring time-varying information visualizations by direct manipulation. **IEEE Trans. Visualization and Computer Graphics**, v. 20, n. 12, p. 2003–2012, Dec 2014. ISSN 1077-2626.

KRAAK, M.-J.; ORMELING, F. J. **Cartography: visualization of geospatial data**. New York, NY, USA: Pearson Education, 2003.

KRUEGER, R.; THOM, D.; ERTL, T. Visual analysis of movement behavior using web data for context enrichment. In: PACIFIC VISUALIZATION SYMPOSIUM, 2014 IEEE. **Proceedings...** Raiosha, Yokohama, Japan, 2014. p. 193–200.

LEE, J.-G.; HAN, J.; WHANG, K.-Y. Trajectory clustering: a partition-and-group framework. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA. **Proceedings...** Beijing, China, 2007. p. 593–604.

LIAO, T. W. Clustering of time series data - a survey. **Pattern Recognition**, Elsevier, New York, NY, USA, v. 38, n. 11, p. 1857–1874, 2005.

LIU, Z.; HEER, J. The effects of interactive latency on exploratory visual analysis. **IEEE Trans. Visualization and Computer Graphics**, v. 20, n. 12, p. 2122–2131, Dec 2014. ISSN 1077-2626.

MACQUEEN, J. et al. Some methods for classification and analysis of multivariate observations. In: FIFTH BERKELEY SYMPOSIUM ON MATHEMATICAL STATISTICS AND PROBABILITY. **Proceedings...** Oakland, CA, USA., 1967. v. 1, n. 14, p. 281–297.

MEGHDADI, A.; IRANI, P. Interactive exploration of surveillance video through action shot summarization and trajectory visualization. **IEEE Trans. Visualization and Computer Graphics**, v. 19, n. 12, p. 2119–2128, Dec 2013. ISSN 1077-2626.

MELNYKOV, V.; MELNYKOV, I. Initializing the {EM} algorithm in gaussian mixture models with an unknown number of components. **Computational Statistics & Data Analysis**, v. 56, n. 6, p. 1381 – 1395, 2012. ISSN 0167-9473. Accessed in: 2016-03-01. Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0167947311003963>.

OLIVEIRA, G. et al. Visualizing running races through the multivariate time-series of multiple runners. In: SIBGRAPI-CONFERENCE ON GRAPHICS, PATTERNS AND IMAGES, IEEE. **Proceedings...** Arequipa, Peru, 2013. p. 99–106.

ORACLE. Glassfish server. 2015. Available from Internet: <https://glassfish.java.net/>. Accessed in: 2016-03-01.

ORACLE. Mysql. 2015. Available from Internet: <https://www.mysql.com/>. Accessed in: 2016-03-01.

PETITJEAN, F.; GANÇARSKI, P. Summarizing a set of time series by averaging: From steiner sequence to compact multiple alignment. **Theoretical Computer Science**, Elsevier, New York, NY, USA, v. 414, n. 1, p. 76–91, 2012.

PEUQUET, D. J. It's about time: A conceptual framework for the representation of temporal dynamics in geographic information systems. **Annals of the Association of american Geographers**, Wiley Online Library, v. 84, n. 3, p. 441–461, 1994.

PEUQUET, D. J. **Representations of space and time**. New York, NY, USA: Guilford Press, 2002.

RINZIVILLO, S. et al. Visually driven analysis of movement data by progressive clustering. **Information Visualization**, SAGE Publications, v. 7, n. 3-4, p. 225, 2008.

SCHROEDER, D.; COFFEY, D.; KEEFE, D. Drawing with the flow: A sketch-based interface for illustrative visualization of 2d vector fields. In: SEVENTH SKETCH-BASED INTERFACES AND MODELING SYMPOSIUM. **Proceedings...** Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2010. (SBIM '10), p. 49–56. ISBN 978-3-905674-25-5. Available from Internet: <http://dl.acm.org/citation.cfm?id=1923363.1923372>.

SELENIUM. Selenium browser automation. 2015. Available from Internet: <http://www.seleniumhq.org/>. Accessed in: 2016-03-01.

SHEN, E. et al. Sketch-based interactive visualization: a survey. **Journal of Visualization**, Springer Berlin Heidelberg, v. 17, n. 4, p. 275–294, 2014. ISSN 1343-8875. Available from Internet: <http://dx.doi.org/10.1007/s12650-014-0225-2>.

SQUARE, I. Crossfilter - fast multidimensional filtering for coordinated views. 2015. Available from Internet: <http://square.github.io/crossfilter/>. Accessed in: 2016-03-01.

TOMINSKI, C.; SCHULZ, H.-J. The great wall of space-time. In: VISION, MODELING & VISUALIZATION, THE EUROGRAPHICS ASSOCIATION. **Proceedings...** Strasbourg, France, 2012. p. 199–206.

TOMINSKI, C. et al. Stacking-based visualization of trajectory attribute data. **IEEE Trans. Visualization and Computer Graphics**, IEEE, v. 18, n. 12, p. 2565–2574, 2012.

TORY, M. et al. Spatialization design: Comparing points and landscapes. **IEEE Trans. Visualization and Computer Graphics**, IEEE, v. 13, n. 6, p. 1262–1269, 2007.

TURKAY, C. et al. Attribute signatures: Dynamic visual summaries for analyzing multivariate geographical data. **IEEE Trans. Visualization and Computer Graphics**, v. 20, n. 12, p. 2033–2042, Dec 2014. ISSN 1077-2626.

WANG, Z. et al. Visual traffic jam analysis based on trajectory data. **IEEE Trans. Visualization and Computer Graphics**, IEEE, v. 19, n. 12, p. 2159–2168, 2013.

WANG, Z. et al. Visual exploration of sparse traffic trajectory data. **IEEE Trans. Visualization and Computer Graphics**, v. 20, n. 12, p. 1813–1822, 2014. ISSN 1077-2626.

WANG, Z.; YUAN, X. Urban trajectory timeline visualization. In: BIGCOMP, INTERNATIONAL CONFERENCE ON BIG DATA AND SMART COMPUTING. **Proceedings...** Bangkok, Thailand, 2014. p. 13–18.

WARE, C. **Information visualization: perception for design**. New York, NY, USA: Elsevier, 2012.

WEI, J. et al. A sketch-based interface for classifying and visualizing vector fields. In: PACIFICVIS, IEEE PACIFIC VISUALIZATION SYMPOSIUM. **Proceedings...** Taipei, Taiwan, 2010. p. 129–136.

WIKIPEDIA. K-means clustering. 2015. <https://en.wikipedia.org/wiki/K-means_clustering>.

WILLEMS, N.; WETERING, H. van de; WIJK, J. J. van. Visualization of vessel movements. In: 11TH EUROGRAPHICS, IEEE - VGTC CONFERENCE ON VISUALIZATION. **Proceedings...** Chichester, UK: The Eurographs Association & John Wiley & Sons, Ltd., 2009. (EuroVis'09), p. 959–966.

WILLIAMS, M.; MUNZNER, T. Steerable, progressive multidimensional scaling. In: INFOVIS 2004, IEEE SYMPOSIUM ON INFORMATION VISUALIZATION. **Proceedings...** Paris, France, 2004. p. 57–64.

WOODHULL, G. Dc.js - dimensional charting javascript library. 2015. Available from Internet: <https://dc-js.github.io/dc.js/>. Accessed in: 2016-03-01.

ZHONG, C. et al. Spatiotemporal visualisation: A survey and outlook. In: **Digital Urban Modeling and Simulation**. New York, NY, USA: Springer, 2012. p. 299–317.