Of Motifs and Goals: Mining Trajectory Data*

Joachim Gudmundsson
School of IT
University of Sydney
NSW 2006, Australia
joachim.gudmundsson@sydney.edu.au

Andreas Thom
Dept. of Computer Science
University of Münster
Münster, Germany
andreas.thom@uni-muenster.de

Jan Vahrenhold
Dept. of Computer Science
University of Münster
Münster, Germany
jan.vahrenhold@uni-muenster.de

ABSTRACT

In response to the increasing volume of trajectory data obtained, e.g., from tracking athletes, animals, or meteorological phenomena, we present a new space-efficient algorithm for the analysis of trajectory data. The algorithm combines techniques from computational geometry, data mining, and string processing and offers a modular design that allows for a user-guided exploration of trajectory data incorporating domain-specific constraints and objectives.

Categories and Subject Descriptors:
H.2.8 [Database Management]: Database Applications—Data Mining

General Terms:
Algorithms.

Keywords:
Trajectory Clustering, Space-Efficiency.

1. INTRODUCTION

Four decades ago, Harry Messel started a ground-breaking study [26, 36] in Northern Australia using solar powered radio telemetry to track large Estuarine crocodiles, *crocodylus porosus*. Since then, technological advances in tracking and sensing devices have greatly facilitated the study of wildlife movement ecology [27, 30, 33]. Ecologists can now use miniaturized animal-borne tags to record high-resolution spatio-temporal movements of wild animals and use these recordings to analyze aspects of the animals' behavior and physiology, or properties of their environments.

Today, a new revolution is underway, not only in ecology but in many areas using tracking technologies. For instance, this change can be seen in team sports analysis: In the late 1990's, the first attempts were made to track football players to analyze performance and team strategies. Nowadays, there are commercial systems [3] that can track the ball, the players, and the referees in real time with very high accuracy and resolution (25 Hz). Sport scientists have gone from spending the majority of their time collecting data to being flooded with data sets that are so large that they can impossibly be analyzed manually.

There is an eclectic set of disciplines that are affected in the same way, including geography [11], surveillance analysis [34], transport analysis [21], and market research [22]. All these areas are in great need of automated tools to make maximal use of the data being collected.

While the technologies for capturing movement have been extensively researched and reached considerable sophistication, the analytical toolbox is much less developed [14]. To a large extent, the analysis on movement data is still done manually, or at most semi-automatically using visual tools. The lack of analytical tools is partly explained by the hardness of accurately defining the problems: Often only vague descriptions of what domain experts seek are given. Hence, the most successful tools up to this date are supervised algorithms where the domain expert interactively can adjust the parameters and thus guide the semi-automated analysis.

Probably the most important open problem in the area is the (sub)trajectory clustering problem. There are two versions of this problem, either cluster a given set of trajectories or cluster subtrajectories of one or many trajectories. We will study the latter (which is a generalization of the first problem), that is, given a set of trajectories find a set of clusters such that every cluster C contains at least k disjoint subtrajectories and the "distance" between each pair of trajectories in C is at most some given threshold α.

1.1 Related research

Since an efficient clustering algorithm for trajectories, or subtrajectories, is essential for various analysis tasks [14, 35], many approaches have been proposed in the past. Here, we briefly discuss the ones most relevant to our work.

A simple approach to cluster trajectories (not subtrajectories) is to map each trajectory into a point and then use a standard clustering algorithm to cluster the points, see for example [8, 29]. The main drawback using an approach like this is its sensitivity to noisy data or outliers. Gaffney *et al.* [12, 13] proposed a model-based clustering algorithm for trajectories, where a set of trajectories is represented using a regression mixture model. Specifically, their algorithm is used to determine cluster memberships and, as above, it only clusters whole trajectories.

*The first author was funded by the Australian Research Council FT100100755. Part of the work of the second and third author has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis" (`http://sfb876.tu-dortmund.de`), project A2.

ACM SIGSPATIAL GIS '12, November 6-9, 2012. Redondo Beach, CA, USA

Clustering trajectories as a whole does not detect similar portions of the trajectories. To overcome this problem, Lee *et al.* [23], and similarly Zhou and Huang [38], suggested an approach that partitions a trajectory into a set of line segments and then group similar segments. The obvious drawback is that only single segments are clustered.

Mamoulis *et al.* [25] used a different approach and assumed that the trajectories are given as a sequence of spatial regions, for example `ABC` would denote that the entity started at region `A` and then moved to region `C` via region `B`. Using this model data mining tools such as association rule mining can be used. A drawback is that two trajectories that might be very similar may visit different regions, thus they will not show up as being similar.

In an attempt to develop more robust measures, researchers considered approaches that had been successful in other areas. A typical example is Dynamic Time Warping (DTW), which was first used to match signals in speech recognition. The DTW is easy to compute. Given polygonal curves represented by the sequences of the vertices, of length m and n respectively, a simple dynamic programming algorithm yields the optimal solution in $\mathcal{O}(mn)$ time. There are fast approximate implementations and they usually work well in practice [20, 31].

A major problem with all the above approaches is that they match the vertices (not curves) along the trajectories which requires that the vertices along the trajectories are uniformly and densely sampled, which is rarely the case. If the trajectories are simplified (compressed) in a preprocessing step then this will definitely not be the case. One could conceivably re-sample the input curves, but that would again increase the complexity of the curves. Furthermore, curves that are almost identical to each other may appear quite different in the distance measure because of different sampling on each curve.

In an attempt to take the continuity of the trajectories into consideration, Buchin *et al.* [4] developed a clustering algorithm using the Fréchet distance, which is sometimes referred to as the "dog-man" distance. The Fréchet distance is a max-measure and the dependence on the maximum value can also lead to non-robust behavior, where small variations in the input can distort the distance function by a large amount [9]. However, algorithms using the Fréchet distance do find more involved clusters than the simpler ones discussed above. The main drawback is in the complexity requirements: Computing the Fréchet distance between two general curves requires quadratic time in the complexity of the curves [2]. It is currently an open question if it is possible to do better even when one tries to approximate this quantity within a constant multiplicative factor. In an effort to make this approach more useful, Gudmundsson and Valladares [15] recently presented a GPU implementation of the algorithm.

Thus every existing subtrajectory clustering algorithm has one of the following drawbacks:

1. High complexity (superlinear space or quadratic running time).

2. Restriction to dectect only simple clusters.

3. Inefficent handling of unsynchronized or sparse data (data from different sensors).

1.2 Motivation and approach

Often analysts are looking for general characteristics such as frequent runs or how a player occupies space. Such patterns can be described using rather coarse terms, e.g., "first move straight, then start a long curve to the right". We wish to facilitate clustering according to such terms that make sense from the analysts' perspectives but seem hard to capture in a domain-independent, formal way. The appealing feature of such terms, however, is that once they have been agreed upon inside a particular application context, they can be used to label the trajectory's subpaths. For the analysis, we then identify each term with a distinct characters of some alphabet; the collection of the labels of a trajectory's subpaths then is a word over this alphabet that can be analyzed using string processing techniques, e.g., for finding frequently occurring substrings.

In addition, our approach has these important advantages:

1. It uses linear space which is a crucial requirement with the recent improvement in tracking technologies.

2. The meaning of a cluster depends on the application and the parameter values; thus it is important that a domain expert easily can guide the program to find the type of clusters that makes sense for his/her analysis. Our algorithm offers a modular design that allows for a user-guided exploration of trajectory data incorporating domain-specific constraints or objectives. Specifically, it supports the functionality to restart a phase of the algorithm with changed parameters without the need to re-run the algorithm from scratch.

3. The pipelined design of our approach enables a semi-dynamic analysis where new trajectories, or extended trajectories, can be added to an already existing set of analyzed trajectories, e.g., the track of a new hurricane can be added to the already existing tracks. Due to the modular design the four first phases of the algorithm only need to be run on the new data, while only the last two phases are global operations that require the full set of trajectories to be considered.

Note that our algorithm makes no assumptions on the input data regarding density, regularity, or synchronization. In particular, the algorithm can also cope with multiple spatially non-aligned trajectories; such situations may arise in team sports analysis, where the general movement pattern of a player may be the same over a series of games but where the exact location of the pattern on the pitch usually depends on the opponents' behavior.

2. OUR ALGORITHM

Following the above discussion, the design of our algorithm is as follows (see Figure 1): In the first phase, we preprocess the input data such that we eliminate as much noise as possible without trading too much of the data's accuracy. The second phase of the algorithm then segments the input data into subtrajectories that are classified and labeled according to their main geometric property, e.g., "wide left turn" or "short straight segment". In the third phase, the algorithm computes frequently occurring substrings, so-called *motifs*, from the resulting string of characters. The algorithm then maps subtrajectories corresponding to motifs

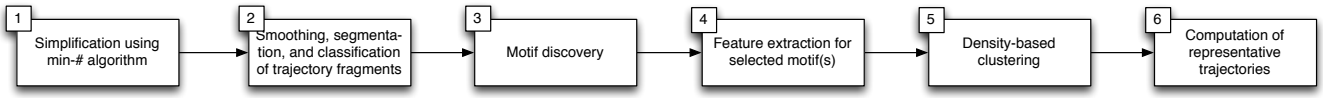

Figure 1: Workflow of the proposed algorithm.

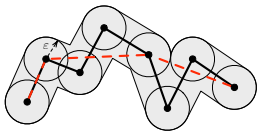

Figure 2: Simplification using a min-# algorithm. The input (black, solid) and output trajectory (red, dashed) have a Haussdorff distance of at most ε.

into some feature space (Phase 4) and performs a density-based clustering (Phase 5). In the final, sixth phase, the resulting clusters are postprocessed for easier visual representation. We next discuss each phase in more detail. The analysis of the algorithm can be found in Section 3.

Simplification.

To decrease the influence of noise in the data and to facilitate recognition of movement patterns, we first run a line-simplification algorithm on the input trajectory P with n vertices. The main objective is not to reduce the number of points on the trajectory but to reduce noise; thus, we use a so-called "min-# algorithm", that, given a tolerance ε, computes a simplified trajectory $P_\varepsilon = \langle p_0, \ldots, p_{n'-1} \rangle$, where $n' \leq n$, with a minimal number of vertices such that the Haussdorff distance between P and P_ε is at most ε, see Fig. 2. P_ε connects a subset of the input vertices in the same order as the original trajectory, intersects each ε-ball centered at an original vertex, and always stays within a tube of radius ε along the original segments.

While there is a variety of min-# algorithms, our implementation is based on the min-# algorithm of Chen and Daescu [7] which only uses linear space. The algorithm constructs edges in a shortest-path graph only as they are needed by the algorithm and thus does not need to explicitly store the quadratic-size all-pairs-shortest-path graph.

Note that using a min-# algorithm (instead of a min-ε algorithm that, given a number $n' \leq n$ reduces the input n-vertex trajectory to an n'-vertex trajectory with the smallest possible Haussdorff distance ε) seems somewhat orthogonal to our initial motivation that the space requirement of the algorithm should be minimal. In contrast to a min-ε algorithm where we can fix a target size for the output, the size of P_ε may still be of the same order of magnitude as P since we only guarantee an upper bound on the distance between P and P_ε. Nonetheless, the data reduction may still be significant. Figure 3 shows an example where a simplification with a very small threshold ($\varepsilon = 0.3$ cm) generates a trajectory that is almost identical to the original trajectory and has less than 50% of the input vertices. In this figure, the player is moving with an average speed of approximately 11 km/h; which also explains why a coarser simplification with a tolerance of $\varepsilon = 50$ cm still yields a visually acceptable approximation as long as the focus is not on local features, such as the "dribbling" in the left-hand part of the figure that is captured by the finer simplification but not in the coarser simplification.

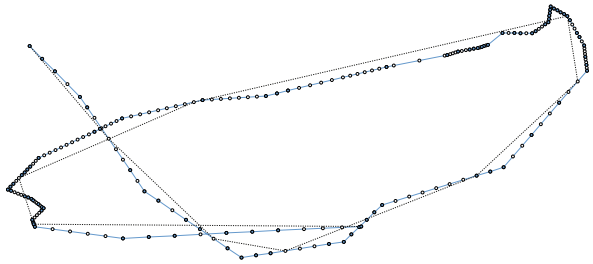

Figure 3: Simplifying a 20-second excerpt from a football player's trajectory inside an area of 20.02 m $\times$ 8.75 m. The input trajectory has 200 vertices, the result has 94 vertices (solid); the trajectories' Haussdorff distance is less than $\varepsilon = 0.3$ cm. The dotted line is an 11-vertex simplification ($\varepsilon = 50$ cm).

Smoothing, Segmentation, and Classification.

As can be seen from Fig. 3, P_ε may still contain very short segments whose directional angle may vary locally. Since we aim at identifying global motion patterns, the second phase of our algorithm first smoothes P_ε and then segments P_ε based upon characteristics derived from the smoothed trajectory P_ε^s.

There is a broad variety of algorithms for smoothing a trajectory, e.g., based on splines, Kalman filters [37], or generalized moving averages [6]. Since we are working with an approximation of the original trajectory and since we do not aim at, e.g., reconstructing a road network from multiple trajectories, we do not need to fully exploit the power of more advanced techniques. We restrict ourselves to the classical moving average technique; note, however, that our general approach also allows for using more advanced techniques as long as the space requirement is linear.

Given the simplified trajectory $P_\varepsilon = \langle p_0, \ldots, p_{n'-1} \rangle$ and a parameter $\kappa \geq 0$, we compute the moving-average trajectory $P_\varepsilon^s = \langle r_0, \ldots, r_{n'-1} \rangle$ of via $r_i := \frac{1}{2\kappa+1} \sum_{i=j-\kappa}^{j+\kappa} p_i$ (we define $p_j := p_0$ for $j < 0$ and $p_j := p_{n'-1}$ for $j > n'-1$). Using P_ε^s, the simplified trajectory P_ε^s is then partitioned into segments of roughly the same motion pattern. In the basic version of our algorithm, we consider the following geometric patterns:

Code	Geometry	Code	Geometry
`L`	wide left turn	`R`	wide right turn
`M`	short left turn	`Q`	short right turn
`X`	sharp left turn	`Z`	sharp right turn
`S`	long straight	`C`	circle
`T`	short straight	`A`	(none of the above)

We note that a set of similar patterns has been used by Harguess and Aggarwal [17] who use a combination of shape matching and combinatorial optimization to match curves to predefined shapes given by their geometric description.

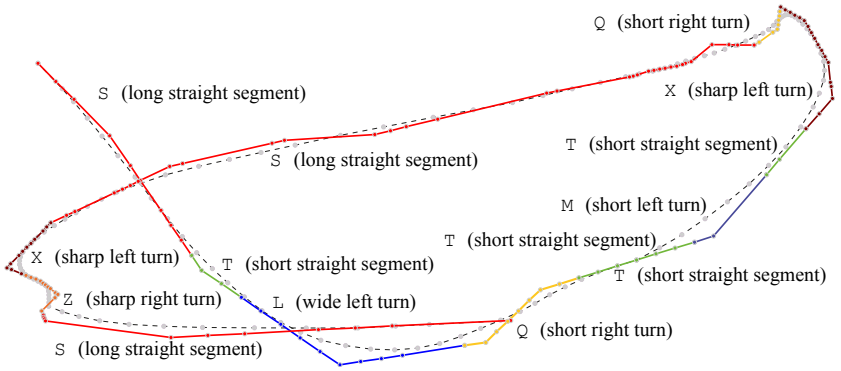

Figure 4: Segmentation of the trajectory from Figure 3 using a moving average window of nine points, i.e., for $\kappa = 4$; The moving average trajectory is shown in the background (dashed).

The segmentation algorithm iterates over all the points of the trajectory $\langle p_0, \ldots, p_{n-1} \rangle$ and, for each point p_i, considers the corresponding moving average point r_i and the κ "past" and "future" vertices in the smoothed trajectory. Starting from an (arbitrary) initial pattern, the algorithm tries to greedily extend the set of points inducing the same general pattern, i.e., "straight", "left curve", or "right curve". The decision of how to classify the current point p_i is based upon separately classifying the "past" subtrajectory $\langle r_{i-\kappa}, \ldots, r_{i-1} \rangle$ and the "future" subtrajectory $\langle r_i, \ldots, r_{i+\kappa} \rangle$. To distinguish between "straight", "left turn", or "right turn", we compute, for the "past" and "future" subtrajectory relative to r_i, the aggregate α_{left} of all left-turn angles, the aggregate α_{right} right-turn angles, and the average β of all angles. The "past" subtrajectory is classified as "straight", if $((\beta \leq \kappa + 1 \wedge \max\{\alpha_{\text{left}}, \alpha_{\text{right}}\} \leq \kappa^2) \vee (|\alpha_{\text{right}} - \alpha_{\text{left}}| \leq k + 1))$, otherwise it is classified as a "left turn" or a "right turn" depending on whether or not $\alpha_{\text{left}} > \alpha_{\text{right}}$. If both subtrajectories are classified in the same way, e.g., as a "left turn", the point p_i is included in the current segment, otherwise a new segment corresponding to the "future" pattern is started. Since this scheme uses monotone criteria, this simple, greedy approach yields an optimal segmentation [5, Theorem 3].

The final classification (e.g., "wide", "short", or "sharp" turn) is done by evaluating each segmented subtrajectory of the trajectory P_ε based on its curvature and the perpendicular distance of its points w.r.t. the segment connecting the first and last point of the subtrajectory. Figure 4 shows an example derived using this algorithm.

As can be seen from Fig. 4, the algorithm may produce two or more consecutive subtrajectories with the same classification—see the two "short straight" subtrajectories in the bottom-right part of the trajectory consisting of three segments each. Such configurations may occur if the moving-average window, i.e., the set of $2\kappa + 1$ vertices around the point currently considered, is larger than the "real" straight subtrajectory; in such a situation, the point inside the window but outside the subtrajectory may cause the smoothed representation to indicate a turning angle that is to large to allow for a "straight" classification. In addition, two consecutive subtrajectories may be classified as the same type of curve, e.g., as a wide right turn, if they join at a point where an isolated opposing turn occurs.

The influence of the parameter κ is at least twofold: first, the larger the value of κ, the more noise is reduced; this is an advantage when looking at coarse trajectory patterns such as "moving forward" but a disadvantage when analyzing more detailed patterns such as "dribbling" patterns. Secondly, the min-# algorithm removes many points from long subtrajectories that are smooth curves, see Fig. 3 (top).

In the case when the samples come from uniformly spaced time steps, many removed samples result in a trajectory where samples are more spread out in time. This in turn implies that the moving average filter may look ahead and back with different (time) intervals, and thus, the smoothing, at least to some extent, is time-dependent.

Motif Discovery.

As described in the previous subsection, the second phase of our algorithm results in a collection of subtrajectories each of which is labeled according to its shape and size. Thus, the input trajectory can be characterized by the sequence of labels assigned to its subtrajectories.

Using the above characterization, the task of identifying common subtrajectories can be stated as the so-called *motif discovery* problem which is a fundamental problem in bioinformatics: given a sequence of characters over a finite alphabet, the motif discovery problem asks for frequently occurring substrings (*motifs*).[1] In the most basic version considered here, given a string s and two positive integers f and ℓ, the task is to report all substrings of s that are of length at least ℓ and occur at least f times in s. Alternative, domain-dependent variations of motif discovery include extracting motifs of length at least ℓ or searching for patterns that match a certain pattern, possibly allowing wildcards or mismatches, see [28, 32] and the references therein.

A standard data structure for processing strings is the so-called *suffix array* in which the possible suffixes along with their respective start index in the input string are represented in lexicographic order—see Figure 5 (right). We point out that a suffix array can be constructed in linear time and linear space using radix sort [19].

To extract the motifs of length ℓ once the suffix array has been constructed, the linear-time algorithm of Abouelhoda *et al.* [1] scans through the suffix array and maintains the length *lcp* of the *longest common prefix* of the current suffix and its predecessor in the suffix array. As soon as the al-

[1]This connection between motifs and data analysis has also been observed by Lin *et al.* [24] who used characters to represent (ranges) of y-values in time series analysis.

Name	Definition	Geometric property / purpose
F1	Minimum axis aligned bounding box	Location and spread of the subtrajectory
F2	Direction vector of start and end point	Global direction
F3	Start and end point	Spatial coherence
F4	Centroid of the vertices	Spatial coherence
F5	Width and height of minimum enclosing rectangle	Geometric shape of the subtrajectory
F6	Motif (mapping from nominal to numeric)	(Strong partitioning; no strict order of the motifs)
F7	Average speed	Distinction between running and walking
F8	Length of subtrajectory	Similarity of curves or straights
F9	Difference of total right and total left turning	Overall turning direction
F10	Average of the curvature radii	Overall curvature of the subtrajectory

Table 1: Features that can extracted from the subtrajectories together with the geometric property the feature aims at characterizing or the purpose, e.g., strongly separating subtrajectories, the feature serves.

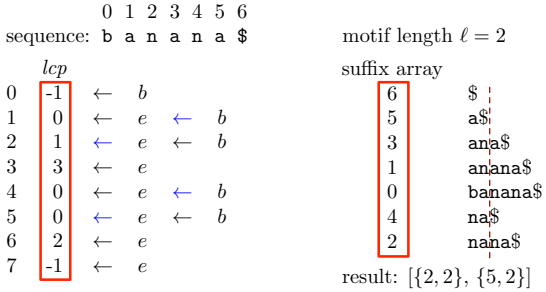

Figure 5: Extracting motifs from a suffix array.

gorithm processes a suffix such that $lcp \geq \ell$, we record the current index b in the suffix array. Then, we scan forward to find the first index e such that lcp drops below ℓ. If the difference $b-e$ is at least as large as the minimum frequency f, we extract the prefix of length m from the entry at position b and record the difference $e-b$ as the frequency of this prefix and continue to scan forward to find the next index b as described above. Figure 5 illustrates this process for finding all motifs of length $\ell = 2$ that occur at least twice.

In our algorithm, we apply the above motif discovery scheme to the string resulting from the classification of the input trajectory. The result of this step, a list of all motifs of given length and frequency can then be refined further by the analyst, e.g., by selecting only the top-k motifs.

Feature Selection.

The result of the third phase of our algorithm is a set of m subtrajectories extracted from the simplified input trajectory where each of the trajectories is additionally labeled with a motif.

In the fourth phase, we map the subtrajectories into a carefully defined feature space and try to cluster these feature points to obtain a clustering of the subtrajectories. A similar approach has been employed by Lee *et al.* [23] who define a distance measure between the (implicitly defined) feature points of two line segments on the basis of the two segments' perpendicular distance, parallel distance, and angle distance. In contrast, we consider multiple-segments subtrajectories and explicitly map these subtrajectories into a feature space. In the case of working with motifs of length $\ell \geq 2$, each feature point represents the (joined) subtrajectories corresponding to the motif.

Table 1 lists several features along with the geometric properties of the subtrajectories that they aim at describing. With the exception of Feature F5, all features can be extracted in linear time. To extract Feature F5 we need the convex hull of the feature points which we compute using the rotating-calipers algorithm, which requires $\mathcal{O}(n \log n)$ time.

We point out that Table 1 is non-exclusive in the sense that depending on the analysis task at hand, other properties may be more appropriate. Due to the modular design of our approach, such additional or alternative features can be easily incorporated. Another important observation is that dimensions in feature space can be scaled thus allowing for emphasizing or de-emphasizing their importance. The latter is important if we wish to ignore small spatial offsets in data space, something that would be penalized heavily when using the Fréchet distance.

Density-based Clustering.

In the fifth phase, the algorithm attempts to identify similar subtrajectories on the basis of the features extracted in the fourth phase. For each subtrajectory, we construct a single feature point in d-dimensional Euclidean space by concatenating the points corresponding to the selected features. For example, Features F1 (minimum axis-aligned bounding box), F3 (start and end point), and F7 (average speed) result in a point in 9-dimensional space, e.g., (*bb_lower*, *bb_left*, *bb_upper*, *bb_right*, *start_x*, *start_y*, *end_x*, *end_y*, *avg_speed*).

To identify clusters in feature space, we employ the well-known DBSCAN algorithm [10] which we briefly sketch for the sake of self-containment. In a nutshell, DBSCAN takes two parameters $\bar{\varepsilon}$ and *MinPts* in addition to a point set S and attempts to identify clusters in S of size at least *MinPts* based upon (local) nearest neighbor computations.

Initially, all points in S are marked as unsettled. DBSCAN then iterates over all unsettled points $\mathbf{x} \in S$ and computes the $\bar{\varepsilon}$-neighborhood $N_{S,\bar{\varepsilon}}(\mathbf{x})$, i.e., the set of unsettled points in S that lie inside the d-dimensional ball $B_{\bar{\varepsilon}}(\mathbf{x})$ of radius δ centered at $\mathbf{x}$. If $N_{S,\bar{\varepsilon}}(\mathbf{x})$ contains at least *MinPts* points, $\mathbf{x}$ is labeled as a core point identifying a new cluster ρ, otherwise it is labeled as noise. If $\mathbf{x}$ was identified as a core point, the algorithm greedily inspects the $\bar{\varepsilon}$-neighborhood $N_{S,\bar{\varepsilon}}(\mathbf{y})$ for each $\mathbf{y} \in N_{S,\bar{\varepsilon}}(\mathbf{x})$. If $N_{S,\bar{\varepsilon}}(\mathbf{y})$ contains at least *MinPts* points, $\mathbf{y}$ is also labeled as a core point of the current cluster ρ and the algorithm

continues the exploration. If, in the other hand, $N_{S,\bar{\varepsilon}}(\mathbf{y})$ contains less than *MinPts* points, $\mathbf{y}$ is labeled as a boundary point of the current cluster ρ. If all points in $N_{S,\bar{\varepsilon}}(\mathbf{x})$ have been inspected, the method terminates indicating that the cluster ρ has been identified completely. At the end of each (recursive) inspection of the neighborhood of a core point, the core point is marked as settled and thus excluded from future computations. Similarly, each boundary point is marked as settled as well.

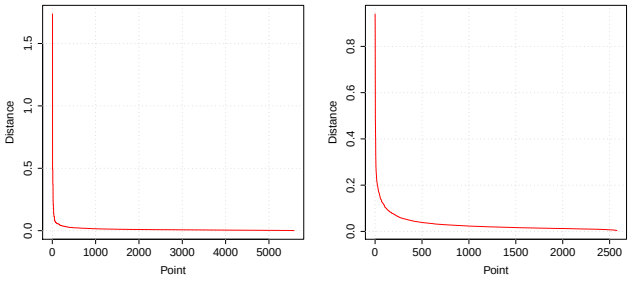

Figure 6: k-distance graph with $k = MinPts = 10$ for the Hurricane (left) and Player 1 (right) data set (see Sections 3.1 and 3.3 for a detailed description).

Our choice of the parameter $\bar{\varepsilon}$ is based upon the original heuristic described by Ester *et al.* [10]. The authors proposed to automatically infer $\bar{\varepsilon}$ from the so-called *k-distance graph*, in which each data point is plotted against the distance to its k-th nearest neighbor and the data points are arranged in descending order of their distance values, see Fig. 6. This graph reflects the density distribution of the data set. According to Ester *et al.* [10, Sec. 4.2], a good candidate for the size of the $\bar{\varepsilon}$-neighborhood is the distance-value of the point at the end of the first valley of the k-distance graph where $k := MinPts > 4$. All points to the right of this value identify a cluster and the others will be classified as noise or a boundary points.

Computation of Representative Trajectories.

In the final phase, we compute representative trajectories for each cluster; these trajectories are then used in the visual representation of our algorithm's output. A representative trajectory should reflect the general movement and the distribution of the segments of the trajectory cluster such that domain experts are enabled to quickly grasp the movement of the entities generating the subtrajectories of each cluster.

As mentioned above, Lee *et al.* [23] proposed an algorithm for clustering line segments. While our algorithm is more general as it allows for clustering not only single segments but also subtrajectories consisting of multiple segments, we can still adopt Lee *et al.*'s algorithm for computing a representative trajectory from a set of line segments—in our case, the segments are extracted from the subtrajectories.

Lee *et al.*'s [23] algorithm processes a set of segments in four steps (refer to Fig. 7). In the first step, the average direction vector $\vec{V}$ of all segments is computed. In the second step the coordinate system is rotated such that $\vec{V}$ points along the x-axis. In the third step, the algorithm performs a standard plane-sweep along the x-axis: the algorithm stops at each start and endpoint of a trajectory and computes the

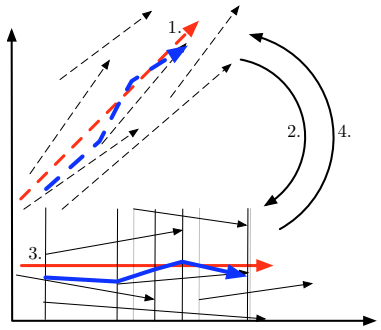

Figure 7: Computing a representative trajectory [23] (blue; red: average direction vector).

representative trajectory's corresponding vertex as the average of all y-coordinates of the segments currently crossed by the sweep-line. Finally, in the fourth step, the rotation performed in Step (2) is undone. The output of the algorithm is shown in Fig. 7 as the dashed blue trajectory.

To prune noise and to avoid excessive segmentation of the representative trajectory, Lee *et al.* propose to use two parameters *MinDist* and *MinLns*. The first parameter lower-bounds the x-distance between two sweep-line events at which a point may be added to the representative trajectory. In Figure 7, gray vertical lines indicate events where no points were added because the distance to the previous event was too small. The second parameter lower-bounds the number of segments that need to be active at a given event to allow the sweep-line to process this event at all; the pruning effect of this parameter can be seen at the beginning and the end of the sweep where no representative trajectory is computed.

Our algorithm automatically infers the values of *MinDist* and *MinLns* from the previous phase: the minimum x-distance *MinDist* is set to half the average length of the segments inside the current cluster and the minimum number of segments required is set to the square root of the minimum cluster size of DBSCAN, i.e., $MinLns := \lfloor\sqrt{MinPts}\rfloor$.

3. ANALYSIS

In this section, we present a theoretical analysis of all algorithmic phases and comparative experimental studies.

Before detailing our analysis, we point out an important conceptual advantage of the pipelined design: the algorithm can be restarted prior to any phase (cf. Table 2) while reusing the results of the previous phases. For example, once the trajectory has been simplified, segmented, classified, and analyzed w.r.t. motifs, the analyst can start multiple runs of the feature extraction and clustering phases without the need to re-run the initial steps of the algorithm. In particular, this facilitates data space exploration since the effects of changing feature extraction or selecting different features for clustering can be investigated with minimal overhead. Finally, the pipelined design of our approach enables a semi-dynamic analysis where new trajectories can be added to an already existing set of analyzed and classified trajectories, e.g., measurements from Champion's League games can be added to the already existing regular season's games. Due to the modular design, the simplification, segmentation, classification, and feature extraction algorithms only need to be run on

Phase		Time	Space	Reference
1	Simplification using min-♯ algorithm	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	[7]
2	Smoothing, segmentation, and classification of (sub-)trajectories	$\mathcal{O}(n)$	$\mathcal{O}(n)$	[this paper]
3	Motif discovery	$\mathcal{O}(n)$	$\mathcal{O}(n)$	[1, 19]
4	Feature extraction for selected motif(s)	$\mathcal{O}(m \log m)^{(*),(\star)}$	$\mathcal{O}(n)$	[this paper]
5	Density-based clustering in d-dimensional feature space	$\mathcal{O}(m \log^{d+1} m)^{(*)}$	$\mathcal{O}(n)$	[10]
6	Computation of representative trajectories	$\mathcal{O}(m \log m)^{(*)}$	$\mathcal{O}(n)$	[23]

Table 2: Analysis of the time and space requirements of the proposed algorithm.
$^{(*)}$: $m \leq n$ denotes the number of points in the subtrajectories selected in Phase (3); usually, $m \ll n$ holds.
$^{(\star)}$: Depending on the features extracted; all features we consider can be extracted in $\mathcal{O}(m \log m)$ time.

the new data, only motif discovery and clustering are global operations that require reconsidering all trajectories.

3.1 Data Sets

The data sets used in our experimental study are taken from two application domains: team sports analysis and weather analysis. We worked with two non-public recordings of football players that were recorded in discrete time steps of 1/10 of a second; each recording of a single player's movement covers a single game and consists of approximately 57,000 points. The resolution of these data sets is 10 mm. For the weather analysis scenario, and in line with the previous study by Lee *et al.* [23], we used the *Hurricane* data set available at `http://weather.unisys.com/hurricane/atlantic/`. This data set contains the hurricane's latitude, longitude, maximum sustained surface wind, and minimum sea-level pressure. Following Lee *et al.* [23], we extracted the latitude and longitude of hurricanes from the years 1950 to 2004. The resulting data set consists of 3,353 trajectories with a total of 106,090 points and thus is a proper superset of the (unspecified) subset used in the study of Lee *et al.* [23] who worked with 570 trajectories consisting of 17,736 points in total.

3.2 Theoretical Analysis

As mentioned in the introduction, one desirable feature of an algorithm for large-scale trajectory analysis is a low space requirement. A single player's collective trajectories sampled a 1/10 of a second over one season of Germany's Bundesliga (34 games/season, 90 minutes/game, no overtime) consist of roughly 1.8 million data points; assuming two bytes to represent a coordinate and assuming that the constant factor in the Big-Oh notation is (unrealistically low) 1, a back-of-the-envelope calculation reveals that a quadratic-space algorithm would require at least 50 GB of RAM.

Table 2 summarizes the time and space requirements of our algorithm and shows that our algorithm is guaranteed to have a linear space requirement. Monitoring the memory used for the whole program revealed that the algorithms' average space consumption when run on a football player's trajectory with approximately 57,000 points was around 15 MB.

As far as the runtime analysis is concerned, we observe two important points: First of all, the complexity of the first three phases depends on the number n of points in the input trajectory, while the runtime of the last three phases depends on the number $m \leq n$ of points in the subtrajectories corresponding to the motifs selected in Phase (3). For all practical scenarios, we expect that $m \ll n$. Since these phases are likely to be re-run multiple time, this implies a significant practical improvement in runtime over non-pipelined algorithms, e.g., those using the Fréchet distance [4]. Related to this, the second observation is that the dominating step of our algorithm is the initial run of the min-♯ simplification algorithm—see Table 2.[2] We point out, however, that this algorithm is run separately for each trajectory, i.e., an x-fold increase in trajectories increases the runtime by a factor of x (instead of by a factor of x^2). This can be observed in practice, too, as the the analysis of the football player's two trajectories (one per half of the game) with approximately 28,000 points each take much longer than the analysis of the Hurricane data set where 3,353 trajectories with 31 points on average even though the total number of vertices in the Hurricane data set (106,090) is almost twice as much as the number of vertices in the football player's data set (57,000)—see Table 3.

Phase	Player 1		Hurricane	
[Load data]	0.8 s	10.0%	1.8 s	15.7%
Simplify	2.2 s	28.3%	0.3 s	2.7%
Smooth/segment/classify	1.2 s	15.2%	2.9 s	25.1%
Extract motifs	0.1 s	0.3%	0.1 s	0.5%
Extract features	1.5 s	18.9%	1.2 s	10.3%
DBSCAN	0.2 s	2.6%	1.0 s	8.6%
Compute representation	0.1 s	0.3%	0.3 s	3.0%
[Clean up]	1.9 s	24.3%	3.9 s	34.1%
Total	8.0 s	100%	11.5 s	100%

Table 3: Breakdown of runtimes for the Player 1 and Hurricane data sets (averaged over ten runs).

Even though the implementation was not optimized for speed, we note that a non-trivial amount of the runtime goes into loading the data from disk and cleaning up the memory allocated throughout the lifetime of the algorithm; the latter task requires up to roughly 1/3 of the total runtime.

3.3 Comparative Studies

To assess our proposed algorithm, we compared it against two previous algorithms that had been using in the respective application domains. For the team sports analysis scenario, we compared against an implementation of the subtrajectory clustering algorithm of Buchin *et al.* [4] (see also [16]), and for the weather analysis scenario, we refer to the experimental study of Lee *et al.* [23]. As mentioned in the introduction, the hardness of accurately defining clustering problems on (sub)trajectories results in most studies being

[2] If one is content with heuristics, there are fast simplification algorithms, e.g., Snoeyink and Hershberger [18] presented an algorithm that needs $\mathcal{O}(n)$ space and $\mathcal{O}(n \log^* n)$ time.

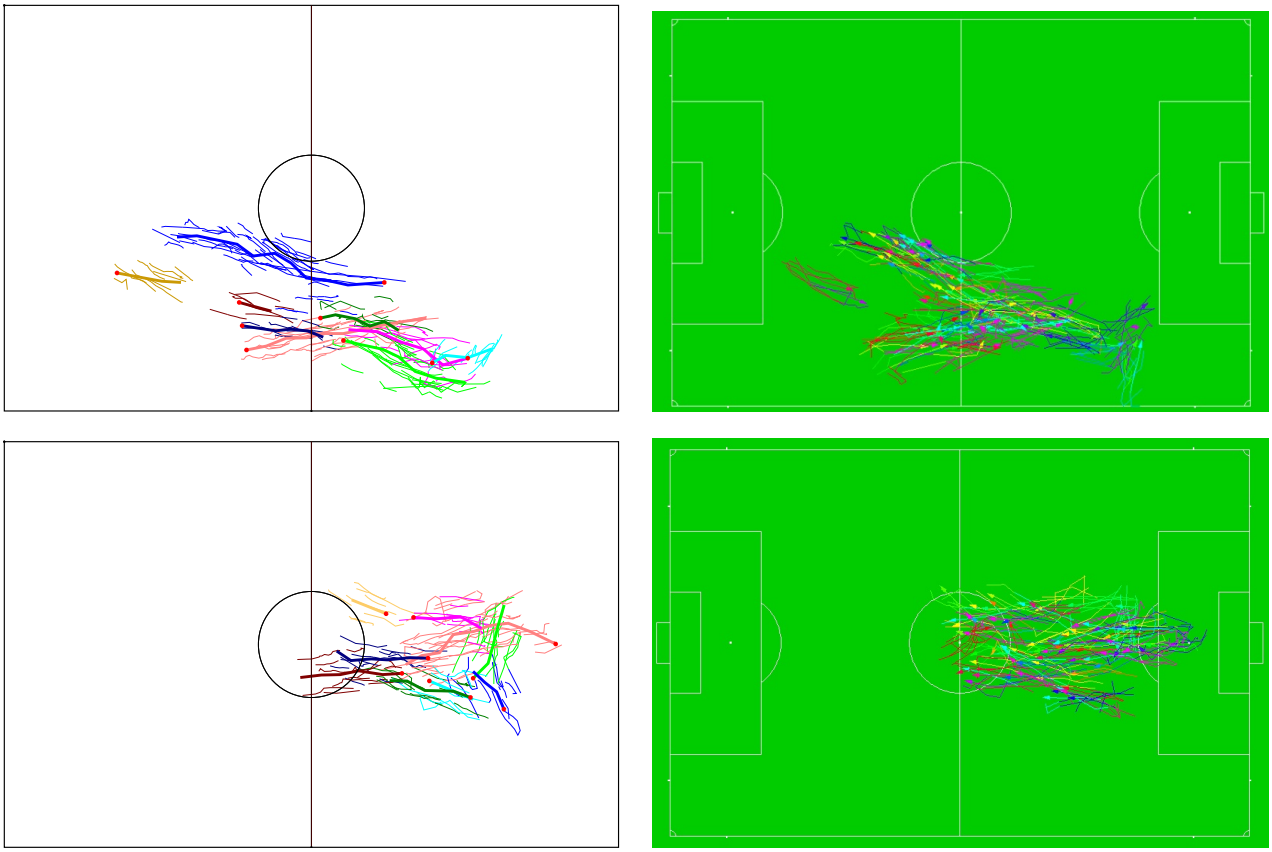

Figure 8: Team sports analysis scenario. Motif clustering (left) compared to clustering with Fréchet distance [4] (right). The top-10 clusters (excluding the "noise" cluster) of the motif clustering are shown; red dots indicate the start of a representative subtrajectory. Parameters for Fréchet distance: $\varepsilon = 3$ m, at least 6 curves in a cluster, representative subcurve in the cluster of length at least 8 m.

done using visual tools. This in turn implies that there is no "ground truth" data to compare clustering results against. As a consequence, we present our results by visualizing them side-by-side with the results of the competitors.

Team Sports Analysis.

As an example for the team sports analysis scenario, we present the analysis of the recordings of a right defender (Player 1; Figure 8, top) and an attacker (Player 2; Figure 8, bottom). Each data set was processed with the following parameters for the motif analysis: $\varepsilon = 0.5$ cm [simplification], $\kappa = 4$ [moving average], motifs {S,L,R}, features {F2, F3, F4, F5, F10}, *MinPts* $= 7$, and $\bar{\varepsilon} = 0.015$ [DBSCAN]. The following table shows the resulting complexities:

Data set	Player 1	Player 2
Input size n	56,957	56,943
Size n' after simplification	15,161	15,634
Clustered subtrajectories	1,341	1,326
Size m of clustered subtrajectories	11,218	12,655

A visual inspection reveals that the shape and the distribution of the clustered trajectories is the roughly the same for both our algorithm and the algorithm of Buchin *et al.* [4].

An intermediate result which may be of independent interest for a domain expert is the distribution of the motion patterns. Table 4 lists the top-5 motifs of length $\ell = 1, 2$ for the Player 1 data set and shows that the predominant motion pattern of this player appears to consist of relatively straight motions; this information may be exploited by domain experts to automatically classify the player by its motion pattern and to select features especially suited to cluster this particular type of motion.

Motif		Frequency	Motif	Frequency
S	long straight	1472	SS	635
T	short straight	1408	TT	455
L	wide left	555	TS	363
R	wide right	554	ST	357
Q	short right	195	RS	194

Table 4: Top-5 motifs for the Player 1 data set.

Table 5 details for each cluster the classifications of the subtrajectories clustered together. Since we did not use Feature F6 (motifs mapped to numerical values) to strongly separate motifs, clusters may consist of straight segments and wide curves as long as the general direction is similar. However, there are still some clusters that consist of subtrajectories of at most two different types; in such cases, we have a very strong evidence of a predominant behavior.

Figure 9: Weather analysis scenario. Motif clustering (left) compared to TraClus clustering (right, Figure 18 in [23]). The middle part of the figure shows a close-up of the right-turn blue cluster.

Player 1				Hurricane			
Size	S	R	L	Size	S	R	L
67	45	13	9	1550	1057	315	178
46	27	13	6	822	316	387	119
45	26	12	7	135	39	42	54
22	13	6	3	59	0	59	0
12	4	4	4	48	1	29	18
12	8	2	2	42	42	0	0
12	3	4	5	41	0	29	12
11	6	3	2	41	23	0	18
9	7	1	1	39	12	27	0

Table 5: Breakdown of the top non-noise clusters of Figures 8 (top) and 9 with respect to the motifs.

Weather Analysis.

As mentioned in Section 3.1, our example for the weather analysis scenario is the set of the trajectories of all atlantic hurricanes from 1954 to 2004. For the motif analysis, we worked with the following parameters: $\varepsilon = 0.1$ units [simplification], $\kappa = 4$ [moving average], motifs {S,L,R}, features {F2, F3, F4, F5, F10}, *MinPts* $= 10$, and $\bar{\varepsilon} = 0.01$ [DBSCAN]. After the simplification algorithm, the number of vertices had been reduced from $n = 106,090$ to $n' = 49,739$; the feature extraction algorithm worked on $m = 39,327$ points from 5,518 subtrajectories.

Figure 9 compares the top-10 clusters (excluding the "noise" cluster) with the visualization of the TraClus algorithm of Lee *et al.* [23] as shown in the original publication. As observed by Lee *et al.*, "some hurricanes move along a curve, changing their direction from east-to-west to south-to-north, and then to west-to-east. On the other hand, some hurricanes move along a straight east-to-west line or a straight west-to-east line." [23, p. 601]. This observation is confirmed by the distribution of the motifs: Table 6 shows that the predominant motion patterns are either straight lines or (wide or short) right curves. One of the clusters consisting of right curves is depicted in the middle part of Figure 9. This cluster, which is located in the middle part of the data set, is not detected by TraClus. This particular cluster also demonstrates a weakness in Lee *et al.*'s algorithm for computing a representative trajectory: a set of curves will be mapped to a relatively straight representative trajectory, if the curvature is large with respect to the extent of the set along the average direction vector—see the almost vertical black representative trajectory computed for the cluster in the left part of Figure 9.

Motif		Frequency	Motif	Frequency
S	long straight	2379	SS	704
R	wide right	2161	RS	620
T	short straight	1821	SR	587
L	wide left	1041	RR	570
Q	short right	919	ST	487

Table 6: Top-5 motifs for the Hurricane data set.

4. SUMMARY

Despite an impressive body of literature, the problem of how to cluster movement data is still wide open, mainly because the problem's exact nature varies very much depending on the application domain. In response to this, we have presented a modular, pipelined algorithm that can be adapted to the domain expert's focus of analysis. From an algorithmic point of view, two features are of particular importance: the algorithm guarantees a linear space requirement which is crucial in the light of the increasing quality and quantity of motion data and it allows for restarting phases of the algorithm with changed parameters without the need of rerunning the more expensive initial phases.

5. REFERENCES

[1] M. I. Abouelhoda, S. Kurtz, and E. Ohlebusch. Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms*, 2:53–86, 2004.

[2] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International Journal on Computational Geometry and Applications*, 5:75–91, 1995.

[3] Amisco. `www.sport-universal.com`.

[4] K. Buchin, M. Buchin, J. Gudmundsson, M. Löffler, and J. Luo. Detecting commuting patterns by clustering subtrajectories. *International Journal of Computational Geometry and Applications*, 21(3):253–282, 2011.

[5] M. Buchin, A. Driemel, M. J. van Kreveld, and V. Sacristan. An algorithmic framework for segmenting trajectories based on spatio-temporal criteria. In *Proc. 18th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems*, pp. 202–211, 2010.

[6] F. Chazal, D. Chen, L. J. Guibas, X. Jiang, and C. Sommer. Data-driven trajectory smoothing. In *Proc. 19th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems*, pp. 251–260, 2011.

[7] D. Z. Chen and O. Daescu. Space-efficient algorithms for approximating polygonal curves in two dimensional space. In *Proc. 4th Annual International Conference on Computing and Combinatorics*, pp. 45–54, 1998.

[8] K. Chu and M. Wong. Fast time-series searching with scaling and shifting. In *Proc. 18th ACM Symposium on Principles of Database Systems*, pp. 237–248, 1999.

[9] A. Efrat, S. Venkatasubramanian, and Q. Fan. Curve matching, time warping, and light fields: New algorithms for computing similarity between curves. *Journal on Mathematic Imaging and Vision*, 27(3):203–216, 2007.

[10] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. Second International Conference on Knowledge Discovery and Data Mining*, pp. 226–231, 1996.

[11] A. U. Frank. Socio-economic units: Their life and motion. In A. U. Frank, J. Raper, and J. P. Cheylan, editors, *Life and motion of socio-economic units*, pp. 21–34. Taylor & Francis, London, 2001.

[12] S. Gaffney, A. Robertson, P. Smyth, S. Camargo, and M. Ghil. Probabilistic clustering of extratropical cyclones using regression mixture models. *Climate Dynamic*, 29(4):423–440, 2007.

[13] S. Gaffney and P. Smyth. Trajectory clustering with mixtures of regression models. In *Proc. 5th ACM International Conference on Knowledge Discovery and Data Mining*, pp. 63–72, 1999.

[14] J. Gudmundsson, P. Laube, and T. Wolle. Computational movement analysis. In W. Kresse and D. Danko, editors, *Springer Handbook of Geographic Information*, pp. 725–741. Springer, 2012.

[15] J. Gudmundsson and N. Valladares. A GPU approach to subtrajectory clustering using the Fréchet distance. *Proc. 20th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems*, to appear, 2012.

[16] J. Gudmundsson and T. Wolle. Football analysis using spatio-temporal tools. *Proc. 20th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems*, to appear, 2012.

[17] J. Harguess and J. K. Aggarwal. Semantic labeling of track events using time series segmentation and shape analysis. In *16th IEEE International Conference on Image Processing*, pp. 4317–4320, 2009.

[18] J. Hershberger and J. Snoeyink. Cartographic line simplification and polygon CSG formulae in $O(n \log^* n)$ time. *Computational Geometry: Theory and Applications*, 11(3–4):93–103, 1998.

[19] J. Kärkkäinen and P. Sanders. Simple linear work suffix array construction. In *Proc. 30th International Colloquium on Automata, Languages and Programming*, pp. 943–955, 2003.

[20] E. J. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, pp. 358–386, 2005.

[21] M. P. Kwan. Interactive geovisualization of activity-travel patterns using three dimensional geographical information systems: A methodological exploration with a large data set. *Transportation Research Part C*, 8(1–6):185–203, 2000.

[22] J. Larson, E. Bradlow, and P. Fader. An exploratory look at supermarket shopping paths. *Intl. Journal of Research in Marketing*, 22(4):395–414, 2005.

[23] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: A partition-and-group framework. In *Proc. ACM International Conference on Management of Data*, pp. 593–604, 2007.

[24] J. Lin, E. J. Keogh, S. Lonardi, and B. Y.-C. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proc. 8th ACM Workshop on Research Issues in Data Mining and Knowledge Discovery*, pp. 2–11, 2003.

[25] N. Mamoulis, H. Cao, G. Kollios, M. Hadjieleftheriou, Y. Tao, and D. Cheung. Mining, indexing, and querying historical spatiotemporal data. In *Proc. 10th International ACM Conference On Knowledge Discovery and Data Mining*, pp. 236–245, 2004.

[26] H. Messel, A. Wells, and W. J. Green. *The alligator region river systems*. Pergamon Press, 1979.

[27] R. Nathan. An emerging movement ecology paradigm. *PNAS*, 105(49):19050–1, 2008.

[28] G. Navarro and M. Raffinot. *Flexible Pattern Matching in Strings – Practical on-line search algorithms for texts and biological sequences*. Cambridge University Press, 2002.

[29] S. Rasetic, J. Sander, J. Elding, and M. A. Nascimento. A trajectory splitting model for efficient spatio-temporal indexing. In *Proc. 31st International Conference on Very Large Data Bases*, pp. 934–945, 2005.

[30] C. Rutz and G. C. Hays. New frontiers in biologging science. *Biology letters*, 5(3):289–292, 2009.

[31] Y. Sakurai, M. Yoshikawa, and C. Faloutsos. FTW: fast similarity search under the time warping distance. In *Proc. 24th ACM Symposium on Principles of Database Systems*, pp. 326–337, 2005.

[32] G. K. Sandve and F. Drabløs. A survey of motif discovery methods in an integrated framework. *Biology Direct*, 11(1), 2006.

[33] J. Shamoun-Baranes, E. van Loon, R. Purves, B. Speckmann, D. Weiskopf, and C. Camphuysen. Analysis and visualization of animal movement. *Biology Letters*, 8(1):6–9, 2012.

[34] J. J. Thomas and K. A. Cook. A visual analytics agenda. *IEEE Computer Graphics and Applications*, 26(1):10–13, 2006.

[35] M. Vlachos, D. Gunopoulos, and G. Kollios. Discovering similar multidimensional trajectories. In *Proc. 18th International Conference on Data Engineering*, pp. 673–682, 2002.

[36] G. J. W. Webb and H. Messel. Movement and dispersal patterns of crocodylus porosus in some rivers of arnhem land, northern australia. *Australian Wildlife Research*, 5(2):263–283, 1978.

[37] G. Welch and G. Bishop. An introduction to the Kalman filter. Technical Report TR 95-041, Dept. of Computer Science, UNC Chapel Hill, 1995. Updated June 2006, `http://www.cs.unc.edu/~welch/kalman`.

[38] Y. Zhou and T. S. Huang. 'Bag of Segments' for motion trajectory analysis. In *15th IEEE International Conference on Image Processing*, pp. 757–760, 2008.