SeTraStream: Semantic-Aware Trajectory Construction over Streaming Movement Data

Zhixian Yan[1]*, Nikos Giatrakos[2]**, Vangelis Katsikaros[2],
Nikos Pelekis**[2], and Yannis Theodoridis**[2]

[1]EPFL, Switzerland
zhixian.yan@epfl.ch
[2]University of Piraeus, Greece
{ngiatrak, vkats, npelekis, ytheod}@unipi.gr

Abstract. Location data generated from GPS equipped moving objects are typically collected as streams of spatiotemporal $\langle x, y, t\rangle$ points that when put together form corresponding *trajectories*. Most existing studies focus on building ad-hoc querying, analysis, as well as data mining techniques on formed trajectories. As a prior step, trajectory construction is evidently necessary for mobility data processing and understanding, including tasks like trajectory data cleaning, compression, and segmentation so as to identify semantic trajectory episodes like stops (e.g. while sitting and standing) and moves (while jogging, walking, driving etc). However, semantic trajectory construction methods in the current literature are typically based on offline procedures, which is not sufficient for real life trajectory applications that rely on timely delivery of computed trajectories to serve real-time query answers. Filling this gap, our paper proposes a platform, namely SeTraStream, for online semantic trajectory construction. Our framework is capable of providing real-time trajectory data *cleaning*, *compression*, *segmentation* over streaming movement data.

1 Introduction

With the growth of location-based tracking technology like GPS, RFID and GSM networks, an enormous amount of trajectory data are generated from various real life applications, including traffic management, urban planning and geo-social networks. A lot of studies have already been established on trajectories, ranging from data management to data analysis. The focus of trajectory data management includes building data models, query languages and implementation aspects, such as efficient indexing, query processing, and optimization techniques [12][25]; whilst the analysis aims at trajectory data mining including issues like classification, clustering, outlier detection, as well as trajectory pattern discovery (e.g. sequential, periodic and convoy patterns) [9][13][20][21].

* This work was conducted during a Short Term Scientific Mission (STSM) of the author sponsored by the COST MOVE project.
** Nikos Giatrakos, Nikos Pelekis and Yannis Theodoridis were partially supported by the EU FP7/ICT/FET Project MODAP.

Recently, semantic trajectory computation has attracted the research interest [1][3][29][30][31][32]. The focus of semantic trajectory construction is initially on the extraction of *meaningful* trajectories from the raw positioning data like GPS feeds. Moreover, sensory elements placed on vehicles can provide additional lower-scale information about their movement. Semantic trajectories manage to encompass both objects' spatiotemporal movement characteristics (at a certain level of abstraction) as well as useful information regarding objects' movement patterns (e.g dwelling, speeding, tailgating) and social activities (see Fig. 1) assigned to different time intervals throughout their lifespan. Current methods of such kind of trajectory construction are mainly offline [1][3][29][30][31][32], which is not enough for modern, real life applications, because positioning data of moving objects are continuously generated as streams and corresponding querying operations often demand result delivery in an online and continuous fashion.

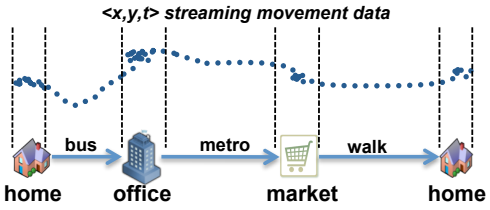

Fig. 1: From streaming movement data to semantic trajectory

Motivating Examples. Online semantic trajectory construction can be useful in many traffic monitoring scenarios where authorities are interested in identifying apart from recent (i.e., within a restricted time window) objects' trajectory representation, the behavior of the drivers by posing queries of the form: *"Report every τ secs the movement and driving behavior of the objects within area A during the last T minutes"*. In that, authorities are able to continuously diagnosing streets where the density of vehicles whose drivers tend to have aggressive (speeding, tailgating, driving at the edges of the lanes etc.) behavior has recently become high, thus enabling suitable placement and periodic rearrangement of traffic wardens and patrol cars. As another example, state-of-the-art navigation services (http://world.waze.com/) provide the potential for combining traditional routing functionality with social networking facilities. Online semantic trajectory construction allows users to acquire a compact picture of the movement and the social activities of interconnected friends around their moving area.

This paper proposes *SeTraStream*, a real-time platform that can progressively process raw mobility data arriving within a restricted time window and compute semantic-aware trajectories online. Before that, a number of data preparation steps need to be considered so as to render data easy to handle and ready to reveal profound movement patterns. The talk regards data cleaning and compression that precede the online segmentation and semantic trajectory computation procedures. Data cleaning is dealing with trajectory errors, including systematic errors (outlier removal) and random errors (smooth noise) [22][31]; compression considers data reduction because trajectory data grow rapidly and lack of compression sooner or later leads to exceeding system capacity [16][23]; segmen-

tation is used for dividing trajectories into episodes where each episode is in some sense homogeneous (e.g. sharing similar velocity, direction etc.) [3] and thus expresses unchanged movement pattern; semantic computation can further extract high-level trajectory concepts like stops/moves [29], and even provide additional tagging support like the activity for stops (e.g. home, office, shopping) and the transportation mode (e.g. metro, bus, walking) for moves [1][30][31][33].

Challenges. It is non-trivial to establish a real-time semantic trajectory computation platform. There exist new technical challenges compared to the existing offline solutions: (1) *Efficient Computation:* Large amounts of movement data are generated continuously, therefore we need to come up with more efficient algorithms which can handle different levels of trajectories in an acceptable time – including all data processing aspects like data cleaning, compression, segmentation, and semantic tagging; (2) *Suitable Trajectory Segmentation Decision Making:* Algorithms in offline trajectory construction typically tune a lot of thresholds placed on movement features (like *acceleration, direction alteration, stop duration etc.*) to find their most suitable values, sometimes in a per object fashion. However, in the real-time context the movement attribute distribution may tremendously vary over time and continuous parameter tuning is prohibitive for real-time semantic trajectory construction. Thus, suitable techniques should not rely on many predefined thresholds on certain movement features but instead consider pattern alterations during the trajectory computation process. (3) *Semantic Trajectory Tagging:* After trajectory segmentation, the outcomes should provide the potentials for semantic tags to be explored, e.g. characterization of the activity (shopping, work) or means of movement that is taking place in episodes (e.g. car, metro, bus in Fig. 1).

Contributions. Towards the objective of real-time semantic trajectory construction, the core contributions of our paper are:

- *Online Trajectory Preprocessing.* As a prior step for constructing semantic trajectories, we significantly redesign trajectory data preprocessing in the real-time context, including *online cleaning* and *online compression*. Our cleaning includes an one-loop procedure for removing outliers and alleviating errors based on a *Kernel smoothing* method. SeTraStream's compression scheme uses a combination of the *Synchronized Euclidean Distance* (*sed*) and the novel definition of a *Synchronized Correlation Coefficient* (*scc*).
- *Online Trajectory Construction.* We design techniques for finding division points which infer trajectory episodes during online trajectory segmentation. SeTraStream's segmentation outcomes are later easy to handle and a *semantic tagging* classifier can then be applied for tag assignment on identified episodes, e.g. "driving", "jogging", "dwelling for shopping" etc.
- *Implementation Platform & Evaluation.* We implement SeTraStream's multilayer procedure for semantic trajectory construction and evaluate it, considering different real life trajectory datasets. The results demonstrate the ability of SeTraStream to accurately provide computed semantic-aware trajectories in real-time, readily available for applications' querying purposes.

The rest of the paper proceeds as follows. In the upcoming section we discuss existing related works. Section 3 describes the preliminaries for semantic trajectory computation in SeTraStream, while in section 4 we present the data preparation procedures regarding incoming data cleaning and compression. In Section 5 we present SeTraStream's online segmentation algorithms and in Section 6 we experimentally evaluate our techniques. Eventually, section 7 includes concluding remarks and future work considerations.

2 Related Work

Trajectory construction is the procedure of reconstructing trajectories from the original sequence of spatiotemporal records of moving objects. Tasks involved in this procedure mainly include *data cleaning*, *data compression* and *data segmentation*. *Data cleaning* is dealing with trajectory errors which are quite common in GPS alike trajectory recordings. There are two types of errors: the outliers which are far away from the true values and need to be removed; the noisy data that should be corrected and smoothed. Several works [22][28][31] design specific filtering methods to remove outliers and smoothing methods to deal with small random errors. Regarding network-constrained moving objects, a number of map matching algorithms have been designed to refine the raw GPS records [2][16].

Trajectory data are generated continuously, in a high frequency and sooner or later grow beyond systems' computational and memory capacity. Therefore, *data compression* is a fundamental task for supporting scalable applications. The spatiotemporal compression methods for trajectory data can be classified into four types: i.e. *top-down*, *bottom-up*, *sliding window*, and *opening window*. The top-down algorithm recursively splits the trajectory sequence and selects the best position in each sub-sequence. A representative top-down method is the Douglas-Peucker (DP) algorithm [6], with many extended implementation techniques. The bottom-up algorithm starts from the finest possible representation, and merges the successive data points until some halting conditions are met. Sliding window methods compress data in a fixed window size; whilst open window methods use a dynamic and flexible window size for data segmentation. To name but a few methods: Meratnia et al. propose *Top-Down Time Ratio* (TD-TR) and *OPen Window Time Ratio* (OPW-TR) for the compression of spatiotemporal trajectories [23]. In addition, the work of [26] provides two sampling based compression methods: *threshold-guided sampling* and *STTrace* to deal with limited memory capacity.

Recently, semantic-based trajectory model construction has emerged as a hot topic for reconstructing trajectories, such as the stop-move concept in [29]. From a semantic point of view, a raw trajectory as a sequence of GPS points can be abstracted to a sequence of meaningful episodes (e.g. *begin*, *move*, *stop*, *end*). Yan et al. design a computing platform to progressively generate spatiosemantic trajectories from the raw GPS tracking feeds [31][32]. In that approach, different levels of trajectories are constructed, from *spatiotemporal trajectories*, *structured trajectories* to the final *semantic trajectories*, in four computational layers, i.e.

data preprocessing, *trajectory identification*, *trajectory structure* and *semantic enrichment*.

Trajectory episodes like stops and moves can be computed with given geographic artifacts [1] or only depend on spatiotemporal criteria like density, velocity, direction etc. [24][27][31]. Alvares et al. develop a mechanism for the automatic extraction of stops that is based on the intersection of trajectories and geometries of geographical features considered relevant to the application [1]. In this approach the semantic information is limited to geographic data that intersect the trajectories for a certain time interval. This approach is restricted to applications in which geographic information can help to identify places visited by the moving object which play the essential role.

Recently, more advanced methods use spatiotemporal criteria to perform trajectory segmentation and identify episodes like stops/moves: Yan et al design a velocity-based method providing a dynamic velocity threshold on stop computation, where the minimal stop duration is used to avoid false positives (e.g. congestions) [31]; several clustering-based stop identification methods have been developed, e.g. using the velocity [24] and direction features [27] of movement. Finally, Buchin et al. provide a theoretical trajectory segmentation framework and claim that the segmentation problem can be solved in $O(nlogn)$ time [3].

Online segmentation concepts can be traced back to the time series and signal processing fields [17], but not initially for trajectories. Although, some of the above works are capable of adapting to an online context [2][16], none of them focuses on revealing the profound semantics present in the computed trajectories in real-time. To the best of our knowledge, online algorithms for semantic trajectory construction are significantly missing. Our objective is to design such online computation methods for real-time semantic trajectory construction.

3 Preliminaries

3.1 Data and Semantic Trajectory Models

In our setting, a central server continuously collects the status updates of moving objects that move inside an area of interest – monitoring area of moving objects. First, such updates involving an object O_i contain spatiotemporal $\langle x, y, t\rangle$ points forming its "*Raw Location Stream*".

Definition 1 (Raw Location Stream) *The continuous recording of spatiotemporal points that update the status of a moving object O_i, i.e. $\langle Q_1^{\ell s}, Q_2^{\ell s}, \ldots, Q_n^{\ell s}\ \rangle$, where $Q_i^{\ell s} = \langle x, y, t\rangle$ is a tuple including moving object's O_i, position $\langle x, y\rangle$ and timestamp t.*

By means of the raw location stream, we can derive information of movement features such as acceleration, speed, direction etc., which make up a "*Location Stream Feature Vector*" ($Q^{\ell f}$). Moreover, depending on the application, updates include additional attributes such as heading, steering wheel activity, lane position, distance to headaway vehicle (e.g to assess tailgating), displacement and

so on. These features formulate a "*Complementary Feature Vector*" (Q^{cf}). Consequently, the two types of feature vectors combined together are forming the "*Movement Feature Vector*" ($\mathcal{Q} = \langle Q^{\ell f}, Q^{cf} \rangle$) of d dimension describing d attributes of O_i movement at a specific timestamp.

Definition 2 (Movement Feature Vector) *The movement attributes of object O_i at timestamp t can be described by a d-dimensional vector that is the concatenation of the location stream feature vector and the complementary feature vector $\mathcal{Q} = \langle Q^{\ell f}, Q^{cf} \rangle$.*
– Location Stream Feature Vector ($Q^{\ell f}$): The movement features of object O_i that can be derived from the raw location stream tuple $Q^{\ell s}$.
– Complementary Feature Vector (Q^{cf}): The movement features that cannot be derived from the location stream but are explicitly included in O_i's status updates.

To provide better understanding and mobility data abstraction, in [29][31] the concept of *semantic trajectories* is introduced, where the trajectory is thought of as a sequence of meaningful episodes (e.g. stop, move, and other self-contained and self-correlated trajectory portions).

Definition 3 (Semantic Movement) *A semantic movement or trajectory consists of a sequence of meaningful trajectory units, called "episodes", i.e. $\mathcal{T}_{sem} = \{e_{first}, \ldots, e_{last}\}$.*
– An episode (e) groups a subsequence of the location stream (a number of consecutive $\langle x, y, t \rangle$ points) having similar movement features.
– From a semantic data compression point of view, an episode stores the subsequence's temporal duration as well as its spatial extent $e_i = (time_{from}, time_{to}, geometry_{bound}, tag)$.

The $geometry_{bound}$ is the geometric abstraction of the episode, e.g. the bounding box of a stop area or the shape trace of roads that the moving object has followed. The term *tag* in the last part of the previous definition refers to the semantics of the episode, i.e. characterization of the activity or means of movement that is taking place in an episode (see Fig. 1).

3.2 Window Specifications

The window specification is a fundamental concept in streaming data processing [8]. In our context, the time window size T expresses the most recent portion of semantic trajectories the server needs to be informed about. An additional parameter τ specifies a time interval in which client side devices, installed on moving objects, are required to collect and report batches of their time ordered status updates [8]. Thus, $\frac{T}{\tau}$ batches are included in the window. Obviously, posed prerequisites are: 1) $\tau \ll T$ and 2) $T \mod \tau = 0$. As the window slides, for each monitored object O_i, the most aged batch expires and a newly received one is appended to it. The size of τ may vary from a few seconds to minutes depending on the application's sampling frequency. Small τ values enable fine-tuned episode extend determination with the make-weight of increased processing costs, while larger τ values reduce the processing load by increasing the granules that are assigned to episodes.

3.3 SeTraStream Overview

Having presented the primitive concepts utilized by our framework, in this subsection we outline SeTraStream's general function. Details will be provided in the upcoming sections. The whole process is depicted in Fig. 2. Upon the receipt of a batch containing the status updates including $Q^{\ell s}, Q^{cf}$ vectors at different timestamps in τ, a cleaning and smoothing technique is applied on it (Step 1 on the right part of the figure). Consequently, a novel compression method (Step 2) is applied on the batch considering both $Q^{\ell s}, Q^{cf}$ characteristics while performing the load shedding. Finally, at a third step $Q^{\ell f}, Q^{cf}$ feature vectors are extracted, a corresponding matrix is formed and the batch is buffered until it is processed at the SeTraStream's segmentation stage. During the segmentation stage (left part of Fig. 2), a previously buffered batch is dequeued and compared with other batches' feature matrices in O_i's window. SeTraStream seeks both for short and long term changes in O_i's movement pattern, and identifies an episode whenever feature matrices are found to be dissimilar based on the RV-Coefficient (to be defined later) and a specified division threshold σ.

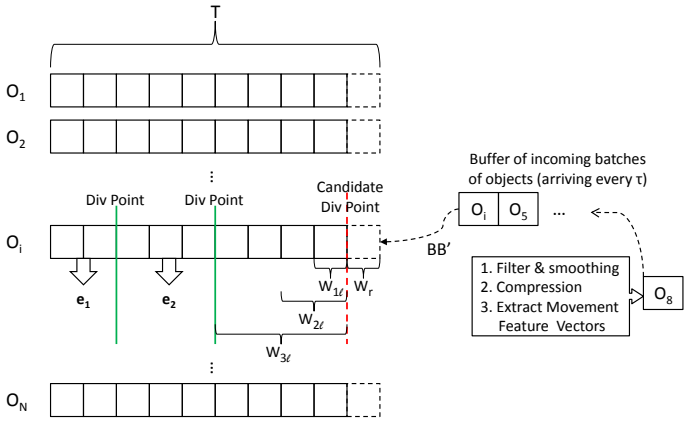

Fig. 2: The SeTraStream Framework

4 Online Data Preparation

As already described, arriving batches involving monitored objects contain their raw location stream, as well as complementary feature vectors. In this section, we discuss the initial steps of data preparation before proceeding to episode determination (i.e. trajectory segmentation). The talk regards three steps depicted in the right part of Fig. 2: (1) an *online cleaning* step that deals with noisy tuples, (2) an *online compression* stage that manages to reduce both the available memory usage and the processing cost in computing trajectories, and (3) extracting movement feature vectors, including both the location stream features and complementary features. Table 1 summarizes the symbology utilized in the current and the upcoming sections as well.

Symbol	Description
N	Number of monitored objects
T, τ	Window size and batch interval
d	Number of movement features
O_i	The i-th monitored object id
B_i	The i-th batch from a candidate div. point
$Q^{\ell s}$	Tuple including $\langle x, y, t\rangle$ triplet of an objects' raw location stream
$Q^{\ell f}$	Feature vector derived from the raw location stream at t
Q^{cf}	Complementary feature vector at timestamp t
$\delta_{outlier}, \delta_{smooth}, \sigma$	Filtering, smoothing and segmentation thresholds respectively
res	The residual between the smoothed and the true value
sed, scc	Synchronous Euclidean Distance and Correlation Coefficient
W_ℓ, W_r	A left and right workpiece respectively
e_i	The i-th episode in an object's window

Table 1: Notations of symbols

4.1 Online Cleaning

The main focus of trajectory data cleaning is to remove GPS errors. Jun et al. [14] summarize two types of GPS errors: *systematic errors* (i.e. the totally different GPS positioning from the actual location which is caused by low number of satellites in view, Horizontal Dilution Of Position HDOP etc.) and *random errors* (i.e. the small errors up to ±15 meters which can be caused by the satellite orbit, clock or receiver issues). These systematic errors are also named "outliers", where researchers usually design *filtering* methods to remove them; whilst random errors are small distortions from the true values and their influences can be decreased by *smoothing* methods. Many offline GPS data cleaning works can be found such as [14][28][31].

In the context of streaming data, *online filtering & smoothing* of streaming tuples has become a hot topic [5][10][11][15][19]. Different from the focus of prior works on data accuracy and distribution estimation, our primary concern of cleaning streaming movement data is refining the data points that have substantial distortion of movement features for computing semantic trajectories[1].

For efficient data cleaning, we need to combine *online filtering* and *online smoothing* in a single loop. When a new batch B regarding object O_i arrives (right part of Fig.2), we do the following cleaning steps:

1. Build a kernel based smoothing model: $(\widehat{x}, \widehat{y}) = \frac{\sum_i k(t_i)(x_{t_i}, y_{t_i})}{\sum_i k(t_i)}$ where $k(t)$ is a function with the property $\int_0^{|B|} k(t)dt = 1$. The kernel function describes the weight distribution, with most of the weight in the area near the point. In our experiments, as in [28], we apply the Gaussian kernel $k(t_i) = e^{-\frac{(t_i - t)^2}{2ß^2}}$, where ß refers to the bandwidth of the kernel.
2. Calculate the residual between the model prediction and the true value $\langle x, y\rangle$ of the examined point $Q_p^{\ell s}$, i.e. $res = \sqrt{(\widehat{x} - x)^2 + (\widehat{y} - y)^2}$.

[1] Q^{cf} values are not examined as the micro-sensory devices of vehicles usually possess self-calibrating capabilities.

3. By using a speed limit v_{limit} and the speed $v_{Q^{\ell s}_{p-1}}$ at the previous point $Q^{\ell s}_{p-1}$, respectively compute the outlier bound ($\delta_{outlier} = v_{limit} \times (t_{Q^{\ell s}_{p}} - t_{Q^{\ell s}_{p-1}})$) and the smooth bound ($\delta_{smooth} = v_{Q^{\ell s}_{p-1}} \times (t_{Q^{\ell s}_{p}} - t_{Q^{\ell s}_{p-1}}) \times 120\%$[2]).
4. Filter out the point if the residual is more than the outlier bound, i.e. $res > \delta_{outlier}$, or replace the location of the point $\langle x, y \rangle$ with the smoothed value $\langle \widehat{x}, \widehat{y} \rangle$ if the residual is between the outlier bound and the smooth bound, i.e. $\delta_{smooth} < res < \delta_{outlier}$. Otherwise, we keep the original $\langle x, y \rangle$ of the point.

This cleaning method has taken both advantages of the distance based outlier removal and the local-weighted kernel smoothing method with linear memory requirements of $O(|B|)$, where $|B|$ is the size of a batch.

4.2 Online Compression

A primary concern when operating in a streaming setting regards the load shedding with respect to incoming tuples. In the context of semantic trajectory computation, this happens both for limiting the available buffer usage as well as to reduce the processing cost [4][16][23][26]. In our approach, as both Definitions 2, 3 imply, the approximation quality of the mere spatiotemporal trajectories is not our only concern. Semantic trajectories will be extracted based on additional features other than those derived from spatiotemporal $\langle x, y, t \rangle$ points. On the other hand, if we overlook the spatiotemporal trajectory approximation quality, the portion of the movement features that rely on the pure location stream will later be uncontrollably distorted. To cope with the previous requirements, we propose a method and define a *significance score* suitable to serve our purposes.

Assume that a batch regarding object O_i is processed (step. 2 at right part of Fig.2) and $(Q^{\ell s}_{p-1}, Q^{\ell s}_{p})$ is the last examined pair of points in it. When a new point $Q^{\ell s}_{p+1}$ is inspected, we first obtain the significance of $Q^{\ell s}_{p}$ from a spatiotemporal viewpoint by fostering the *Synchronous Euclidean Distance*, defined as [23][26]: $sed(Q^{\ell s}_{p}, Q^{\ell s}_{p-1}, Q^{\ell s}_{p+1}) = \sqrt{(x_{Q'^{\ell s}_{p}} - x_{Q^{\ell s}_{p}})^2 + (y_{Q'^{\ell s}_{p}} - y_{Q^{\ell s}_{p}})^2}$, with $x_{Q'^{\ell s}_{p}} = x_{Q^{\ell s}_{p-1}} + v^{x}_{Q^{\ell s}_{p-1}Q^{\ell s}_{p+1}} \cdot (t_{Q^{\ell s}_{p}} - t_{Q^{\ell s}_{p-1}})$ and $y_{Q'^{\ell s}_{p}} = y_{Q^{\ell s}_{p-1}} + v^{y}_{Q^{\ell s}_{p-1}Q^{\ell s}_{p+1}} \cdot (t_{Q^{\ell s}_{p}} - t_{Q^{\ell s}_{p-1}})$ while v^x, v^y refer to the velocity vector (please refer to [26] for further details).

Nevertheless, *sed* constitutes an absolute number that lacks the ability to quantify the particular significance of a point with respect to other spatiotemporal points within the current batch. In order to appropriately derive the aforementioned significance quantification, in SeTraStream's compression scheme we normalize *sed* and define the relative spatiotemporal significance Sig^{SP}:

$$Sig^{SP}(Q^{\ell s}_{p}) = \frac{sed(Q^{\ell s}_{p}, Q^{\ell s}_{p-1}, Q^{\ell s}_{p+1})}{max_{sed}} \tag{1}$$

[2] Here, we increase the smooth bound by 20% of the location prediction provided by the speed of the previous point.

with $0 \leq Sig^{SP}(Q_p^{\ell s}) \leq 1$. The denominator max_{sed} denotes the current maximum *sed* of points in the batch. Obviously, increased $Sig^{SP}(Q_p^{\ell s})$ estimations represent points of higher spatiotemporal significance.

Carefully inspecting *sed*'s formula, we can conceive that the intuition behind its definition is to measure the amount of distortion that can be caused by pruning the spatiotemporal point $Q_p^{\ell s}$. That is, having omitted $Q_p^{\ell s}$ we could virtually infer the respective data point at timepoint $t_{Q_p^{\ell s}}$ using the preceding and succeeding ones $(Q_{p-1}^{\ell s}, Q_{p+1}^{\ell s})$. And calculating $Q_p'^{\ell s}$, $sed(Q_p^{\ell s}, Q_{p-1}^{\ell s}, Q_{p+1}^{\ell s})$ measures the incorporated distortion.

Thus, as regards the complementary feature vectors of O_i we choose to base the measure of their significance on the *Correlation Coefficient (corr)* metric. First, fostering an attitude similar to that in *sed*'s calculation as explained in the previous paragraph, we estimate the value at the i-th position of vector $Q_p'^{cf}$ as: $[Q_p'^{cf}]_i = [Q_{p-1}^{cf}]_i + \frac{[Q_{p+1}^{cf}]_i - [Q_{p-1}^{cf}]_i}{t_{Q_{p+1}^{cf}} - t_{Q_{p-1}^{cf}}}(t_{Q_p^{cf}} - t_{Q_{p-1}^{cf}})$. Then, based on *corr* we define the *Synchronized Correlation Coefficient (scc)* between $(Q_p'^{cf}, Q_p^{cf})$ of complementary feature vectors:

$$scc(Q_p'^{cf}, Q_p^{cf}) = \frac{E(Q_p'^{cf} Q_p^{cf}) - E(Q_p'^{cf})E(Q_p^{cf})}{\sqrt{(E((Q_p'^{cf})^2) - E^2(Q_p'^{cf}))(E((Q_p^{cf})^2) - E^2(Q_p^{cf}))}} \quad (2)$$

where $E()$ refers to the mean and $-1 \leq scc(Q_p'^{cf}, Q_p^{cf}) \leq 1$.

The choice of *scc* is motivated by the fact that its stem, *corr*, possesses the ability to indicate the similarity of the trends that are profound in the examined vectors rather than relying on their absolute values [5][10][11][19]. Hence, it provides an appropriate way to identify (dis)similar patterns in the complementary vectors and can be generalized in order to detect similar patterns between movement feature vectors in their entirety. Values of *scc* that are close to -1 exhibit high dissimilarity between $(Q_p'^{cf}, Q_p^{cf})$, indicating that omitting Q_p^{cf} results in higher pattern distortion. Calculating $1 - scc$ enables higher measurements to account for more dissimilar patterns and taking one step further, min-max normalization on $1 - scc$ allows (dis)similarity values lie within $[0, 1]$. Thus, we eventually compute the relative significance of the complementary feature vector:

$$Sig^C(Q_p^{cf}) = \frac{1 - scc(Q_p'^{cf}, Q_p^{cf})}{2max\{(1 - scc)\}} \quad (3)$$

In the context of our compression scheme, the more dissimilar $(Q_p'^{cf}, Q_p^{cf})$ are, the higher the probability to be included in the window should be. As a result, the overall significance $Sig(Q_p)$ of Q_p can be estimated by the combination of both the location stream feature $Sig^{SP}(Q_p^{\ell s})$ and the complementary feature $Sig^C(Q_p^{cf})$. The weight balance between them is application dependent, though we choose to treat them equally important [20]:

$$Sig(Q_p) = \frac{1}{2}(Sig^{SP}(Q_p^{\ell s}) + Sig^C(Q_p^{cf})) \quad (4)$$

Eventually, for a threshold $0 \leq Sig_{thres} \leq 1$, Q_p remains in the batch when $Sig(Q_p) \geq Sig_{thres}$, or it is removed for compression purposes otherwise.

5 Semantic Trajectory Construction

We now describe the core of SeTraStream, the online trajectory segmentation stage. This stage comes after data cleaning and compression utilizing the *extracted feature vectors* of a batch (step. 3 at right part of Fig.2).

5.1 Online Episode Determination - Trajectory Segmentation

Upon deciding the data points of a batch that are to be included in the window as devised in the previous subsection, SeTraStream proceeds by examining episode existence in T. To start with, we assume the simple case of the current window consisting of a couple of τ-sized batches (i.e. $T = 2\tau$). We will henceforth refer to each part of the window composed of a number of compressed batches as *workpiece*. Intuitively, distinguishing episodes is equivalent to finding a *division point*, where the movement feature vectors on its left and right sides are uncorrelated and thus correspond to different movement patterns. In our simple scenario, a *candidate division point* is placed in the middle of the available workpieces.

Hence, we subsequently need to dictate a suitable measure in order to determine movement pattern change existence. We already noted the particular utility of the correlation coefficient on the discovery of trends [5][10][11][19], and thus (in our context) patterns in the movement data. In this processing phase movement feature vectors composing each workpiece essentially form a pair of matrices for which correlation computation needs to be conducted. As a result, we will reside to the *RV-coefficient* which constitutes a generalization of the correlation coefficient for matrix data. We organize W_ℓ into a $d \times m$ matrix, where d is the number of movement features and m represents a number of vectors (at different timestamps) that are the columns of the matrix. Similarly, W_r is organized in a $d \times n$ matrix i.e. n columns exist. The *RV-Coefficient* between $\langle W_\ell, W_r \rangle$ is defined as:

$$RV(W_\ell, W_r) = \frac{Tr(W_\ell W'_\ell W_r W'_r)}{\sqrt{Tr([W_\ell W'_\ell]^2) Tr([W_r W'_r]^2)}} \tag{5}$$

where W'_ℓ, W'_r refer to the transpose matrices, $Tr()$ denotes the trace of a matrix and $0 \leq RV \leq 1$. RV values closer to zero are indicative of uncorrelated movement patterns. Based on a division point threshold σ workpieces W_ℓ, W_r can be assigned to a pair of different episodes $e_\ell = (0, T - \tau, geometry_{bound})$, $e_r = (T - \tau + 1, T, geometry_{bound})$ when:

$$RV(W_\ell, W_r) \leq \sigma \tag{6}$$

or to a single episode $e = (0, T, geometry_{bound})$ otherwise.

Now, consider the general case of T covering an arbitrary number of batches. It can easily be conceived that in a larger time window an alteration in the movement pattern may happen: (a) instantly as a sharp change, or (b) in a more smooth manner as time passes. As a result, upon the arrival of a new

workpiece W_r, we initially check for short-term changes in the patterns of movement. We thus place a candidate division point between the newly received workpiece and the last of the existing ones. Then the correlation between the movement feature vectors present in $\langle W_{1\ell}, W_r \rangle$ is computed. Notice that $W_{1\ell}$ this time possesses an additional subscript which denotes the step of the procedure, as will be shortly explained. Similarly to our discussion in the previous paragraphs, when $RV_1(W_{1\ell}, W_r)$ is lower than the specified division threshold, a division point exists and signals the end of the previous episode e_ℓ and starts a new one e_r.

No short-term change existence triggers our algorithm to proceed by seeking long-term dis-correlations. For this purpose, we first examine $RV_2(W_{2\ell}, W_r)$ doubling the time scale of the left workpiece by going 2τ units back in the window from the candidate division point. In case RV_2 does not satisfy Inequality 6, this procedure continuous by *exponentially expanding* the time scale of the left workpiece in a way such that at the i-th step of the algorithm the size of $W_{i\ell}$ is $2^{(i-1)}\tau$ units and $RV_i(W_{i\ell}, W_r)$ is calculated. When Inequality 6 is satisfied the candidate division point is a true division point which bounds the previous episode $e_i = (time_{from}, time_{to}, geometry_{bound})$ and constitutes the onset of a new. Otherwise, W_r is rendered the current bound of the last episode by being appended to it. If no long-term change is detected, the aforementioned expansion ceases when either the beginning of the last episode or the start of T (in case all previous batches have been attributed to the same episode) is reached, i.e. no data points of the penultimate episode are considered since its extend has already been determined.

The exponential workpiece expansion fostered here is inspired by the *tilted time window* definition [8] as a general and rational way to seek movement pattern changes in different time granularities. Other expansion choices can also be applied. All of these options are orthogonal to our approaches and do not affect the generic function of SeTraStream. Our approach manages to effectively handle sliding windows as a slide of τ time units results in: (1) the expiration of the initial batch of the first episode e_{first} of O_i which affects its $(time_{from}, geometry_{bound})$ attributes and (2) the appendage of a newly received batch that either extends the last episode e_{last} (when no division point is detected) or starts a new episode. The outcome of the online segmentation consists of tuples $\mathcal{T}_{O_i} = \{e_{first}, \ldots, e_{last}\}$ representing objects' semantic trajectories.

5.2 Time and Space Complexity

The introduced trajectory segmentation procedure, premises that a newly appended batch will be compared with left workpieces that may be (depending on whether a division point is detected) exponentially expanded until either the previous episode end or the start of the window is reached. Based on this observation, the lemma below elaborates on the complexity of the checks required during candidate division point examination.

Lemma 1. *The time complexity of SeTraStream's online segmentation procedure, for N monitored objects, under exponential $W_{i\ell}$ expansion is $O(Nlog_2(\frac{T}{\tau}))$ per candidate division point.*

Proof. For a single monitored object, the current window is composed of $\frac{T}{\tau}-1$ batches (excluding the one belonging to W_r). The worst case scenario appears when no previous episode exists in the window and the candidate division point is not proven to be an actual division point. By considering the exponential workpiece expansion, comparisons (i.e., σ checks) may reach a number of $k = min\{i \in \mathbb{N}^* : \frac{\frac{T}{\tau}-1}{2^{(i-1)}} \geq 1\}$ at most. Adopting logarithms on the previous expression and summing for N objects completes the proof. □

Now, recalling the definition of the *RV-Coefficient* measure, it can easily be observed that its computation relies on the multiplication of the bipartite matrices with their transpose. Assume that the number of d-dimensional movement feature vectors in a cleaned and compressed batch are n. Based on the above observation we can see that instead of maintaining the original form of the vectors which requires $O(d \cdot n)$ memory space, we can reduce the space requirements during episode determination by computing the product of the $d \times n$ matrix of the batch with its transpose. This reduces the space requirements to $O(d^2)$ per batch since in practice $d \ll n$. So, to check a short-term change in the movement patterns we do not need to store the full matrices of $W_{1\ell}, W_r$ which in this case are composed of one batch each, but only the matrix products as described above.

However, this point may not be of particular utility since left workpieces are expanded during the long-term pattern alteration checks. A natural question that arises regards whether or not the product $W_{i\ell}W'_{i\ell}$ can be expressed by means of the multiplication of single batch matrices, with their transposes.

Lemma 2. *$W_{i\ell}W'_{i\ell}$ is the sum of batch matrix products with their transposes: $W_{i\ell}W'_{i\ell} = \sum_{j=1}^{2^{(i-1)}} B_jB'_j$, where B_j is used to notate the matrix formed by the vectors in the j-th batch (from a candidate division point to the end of $W_{i\ell}$).*

Proof. Let $W_{i\ell} = [B_1|B_2|\cdots|B_{2^{(i-1)}}]$ the matrix of the (i-th) left workpiece during the current division point check. B_js are used to denote sub-matrices belonging to individual batches that were appended to the workpiece. It is easy to see that the transpose matrix can be produced by transposing these submatrices: $W'_{i\ell} = [B'_1|B'_2|\cdots|B'_{2^{(i-1)}}]$. And then $W_{i\ell}W'_{i\ell}$ can be decomposed into $B_jB'_j$ products: $W_{i\ell}W'_{i\ell} = B_1B'_1 + B_2B'_2 + \cdots + B_{2^{(i-1)}}B'_{2^{(i-1)}} = \sum_{j=1}^{2^{(i-1)}} B_jB'_j$ □

Thus, for each batch we only need to store a square $d \times d$ matrix[3], which determines the space complexity of online segmentation leading to Lemma 3.

[3] We also keep the geometry bound of the batch that is utilized in the final episode geometry bound determination as well as some additional aggregate statistics, of minor storage cost, for classification and tag assignment in the next step.

Lemma 3. *During the online episode determination stage of SeTraStream, the memory requirements per object O_i are $O(d^2\frac{T}{\tau})$ and assuming N objects are being monitored the total space utilization is $O(d^2N\frac{T}{\tau})$.*

5.3 Episode Tagging

Having detected an episode e_i, SeTraStream manages to specify in an online fashion the triplet $(time_{from}, time_{to}, geometry_{bound})$ describing its spatio-temporal extend. The final piece of information associated with an episode regards its *tag* as it was described in Section 3.1. Given application's context, possible *tag* instances form a set of movement pattern classes and notice that the instances of the classes are predetermined for the applications we consider (Section 1). Hence, the problem of episode tag assignment can be smelted to a trivial classification task, where the classifier can be trained in advance based on the collected episodes (with features like segment *distance*, *duration*, *density*, *avg. speed*, *avg. acceleration*, *avg. heading* etc.) and the detected episode e_i can be timely classified based on the trained model and the episode features. Suitable techniques include decision trees, boosting, SVM, neural or Bayesian networks [7]. Additional Hidden Markov Model based trajectory annotation can be referred to [30].

6 Experiments

In this section, we present our experimental results in real-time construction of semantic trajectories from streaming movement data.

Experimental Setup. We utilize two different datasets: *Taxi Data* - this dataset includes taxi trajectory data for 5 months with more than 3M GPS records, which do not have any complementary features. We mainly use taxi data to validate compression. It is non-trivial to get real life on-hand dataset with both complementary features and the underlying segment ground-truth tags. Therefore, we collect our own trajectory data by developing Python S60 scripts deployed in a Nokia N95 smartphone, which can generate both GPS data and accelerometer data from the embedded sensors. We calculate GPS features (e.g. *transformed longitude, latitude, speed, direction*) as the location stream vectors ($Q^{\ell f}$) and accelerometer features (e.g. *mean, variance, magnitude, covariance of the 3 accelerometer axis*) as the complementary feature vectors (Q^{cf}). We term the latter dataset as *Phone Data* within which, we also provide our own real segment tags (e.g. *standing, jogging, walking*) to validate the online segmentation accuracy. For *Phone Data*, we also work on the GPS data from the data campaign organized by Nokia Research Center - Lausanne, which has collected 185 users' phone data with about 7M records in total [18][30].

Data Cleaning. As described previously, our online data cleaning needs to consider two types of GPS data errors, i.e. filtering outliers as systematic errors and smoothing the random errors. The experimental cleaning results are shown in Fig. 3: (a) sketches the original trajectory data; (b) identifies the outliers during

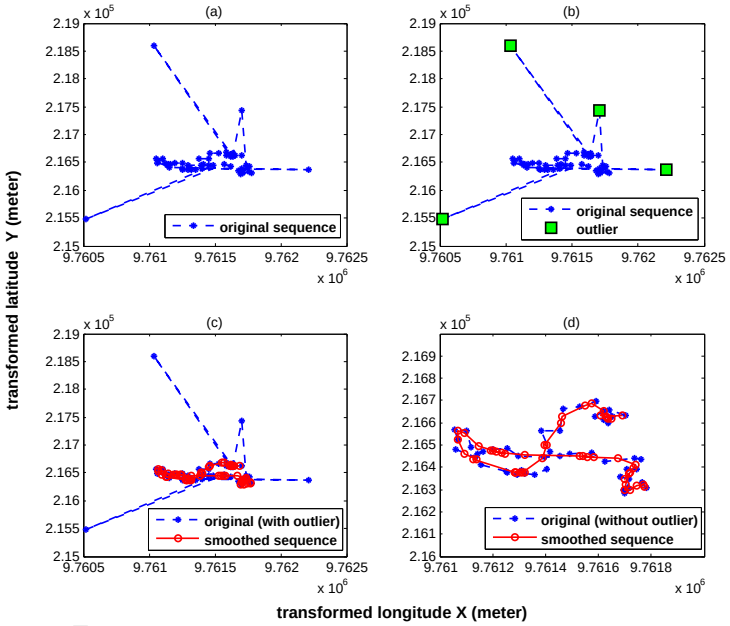

Fig. 3: Data cleaning (outlier removal and smoothing)

the online cleaning process; (c) and (d) present the original movement sequences together with the final smoothed trajectories, where (c) includes the outliers in the original sequences, whilst (d) removes them for better visualization.

Online Compression. Technically, compression makes sense when dealing with large data sets, however both *Taxi Data* and the big part of *Phone Data* have no complementary features (Q_p^{cf}) available but only the GPS features ($Q_p^{\ell s}$). Thus, our current experiment validates the sensitivity of data compression rate with respect to the spatiotemporal significance $Sig^{SP}(Q_p^{\ell s})$ on location streams, without considering the significance of the complementary features $Sig^{C}(Q_p^{cf})$. As shown in Fig. 4, we plot the compression rate sensitivity when applying different thresholds on $Sig^{SP}(Q_p^{\ell s})$. The results are proportional when using the *Phone Data* with respective $Sig(Q_p)$ thresholds.

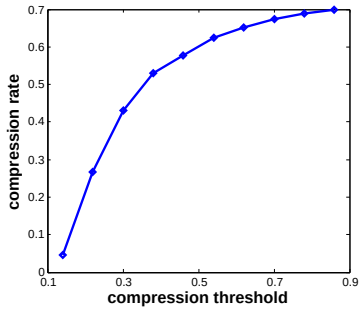

Fig. 4: Data compression rate w.r.t. different thresholds $Sig^{SP}(Q_p^{\ell s})$

Online Segmentation. SeTraStream's procedure in online trajectory segmentation relates to (1) initially computing the RV-coefficient between two work-

pieces $RV(W_\ell, W_r)$ and (2) expanding W_ℓ if $RV(W_\ell, W_r)$ is bigger than the given threshold σ (otherwise, we identify a division point between two episodes). Results are shown in Fig. 5, where for $T = 60s$ we can discover two main division points (with RV-coefficient< 0.6 and batch size $\tau < 16$), which is consistent with the underlying ground-truth tags. The stars in the figures are the real division points in the streaming data, which indicate when user changes their movement behaviors e.g. from jogging to walking and finally to standing.

Fig. 5 analyzes the sensitivity of using different batch sizes, where the best outcome (i.e accurate episode extend determination) is $\tau = 8s$; when $\tau = 16s$, we actually identify three division points, which is partially correct, since as we can see there are only two real division points in the stream. Similarly, we also investigate the segmentation sensitivity regarding different division thresholds σ in Fig. 6. The best segmentation result is achieved when $\sigma = 0.6$.

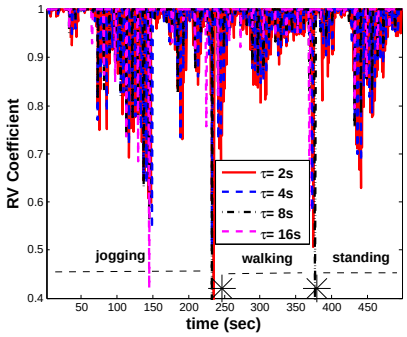

Fig. 5: Episode identification varying batch size, for $\sigma = 0.6$

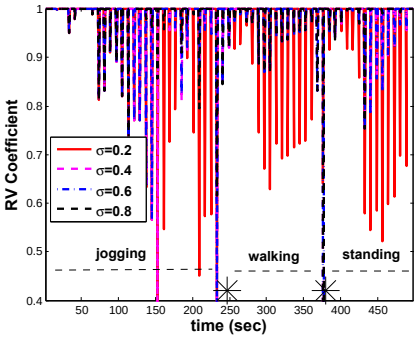

Fig. 6: Sensitivity of RV w.r.t. different σ at $\tau = 8s$

Finally, we evaluate the time performance of SeTraStream's trajectory segmentation module. We measure the segmentation latency with 25 users in the *Phone Data*. In the experiments, we used a laptop with 2.2 Ghz CPU and 4 Gb of memory. From Fig.7 and Fig. 8, we can see the segmentation time is almost linear, in both situations with different batch sizes (τ) and different division thresholds (σ), which is quite consistent with Lemma 1.

7 Conclusions and Future Work

In this paper, we proposed a novel and complete online framework, namely *SeTraStream* that enables semantic trajectory construction over streaming movement data. As far as we know, this is the first method proposed in the literature tackling with this problem in real-time streaming environments. Moreover, we considered challenges occurring in real world applications including data cleaning and load shedding procedures before accurately identifying trajectory episodes in objects' streaming movement data.

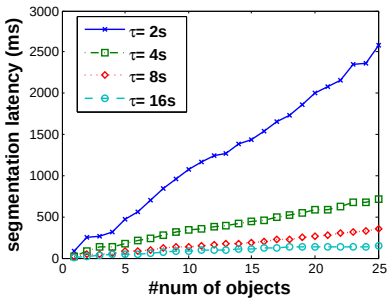

Fig. 7: Segmentation latency with different τ sizes (σ=0.6)

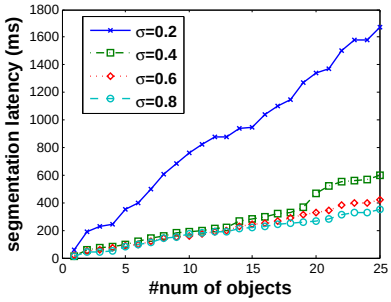

Fig. 8: Segmentation latency with different σ thresholds ($\tau = 8s$)

Our future work is to further evaluate this method with larger datasets including more complementary features and ground-truth tags. In addition, we are planning to extend SeTraStream to (1) handle multiple window types for online trajectory segmentation, and (2) perform real-time trajectory construction in distributed settings often encountered in large scale application scenarios.

References

1. L. O. Alvares, V. Bogorny, B. Kuijpers, J. Macedo, B. Moelans, and A. Vaisman. A Model for Enriching Trajectories with Semantic Geographical Information. In *GIS*, 2007.
2. S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *VLDB*, 2005.
3. M. Buchin, A. Driemel, M. V. Kreveld, and V. Sacristan. An Algorithmic Framework for Segmenting Trajectories based on Spatio-Temporal Criteria. In *GIS*, 2010.
4. H. Cao, O. Wolfson, and G. Trajcevski. Spatio-Temporal Data Reduction With Deterministic Error Bounds. *The VLDB Journal*, 15(3), 2006.
5. A. Deligiannakis, Y. Kotidis, V. Vassalos, V. Stoumpos, and A. Delis. Another Outlier Bites the Dust: Computing Meaningful Aggregates in Sensor Networks. In *ICDE*, 2009.
6. D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2), 1973.
7. U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.
8. C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu. *Mining Frequent Patterns in Data Streams at Multiple Time Granularities*. MIT Press, 2002.
9. F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory Pattern Mining. In *KDD*, 2007.
10. N. Giatrakos, Y. Kotidis, and A. Deligiannakis. PAO: Power-efficient Attribution of Outliers in Wireless Sensor Networks. In *DMSN*, 2010.
11. N. Giatrakos, Y. Kotidis, A. Deligiannakis, V. Vassalos, and Y. Theodoridis. TACO: Tunable Approximate Computation of Outliers in Wireless Sensor Networks. In *SIGMOD*, 2010.
12. R. Güting and M. Schneider. *Moving Objects Databases*. Morgan Kaufmann, 2005.

13. H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen. Discovery of Convoys in Trajectory Databases. In *VLDB*, 2008.
14. J. Jun, R. Guensler, and J. Ogle. Smoothing Methods to Minimize Impact of Global Positioning System Random Error on Travel Distance, Speed, and Acceleration Profile Estimates. *Transportation Research Record: Journal of the Transportation Research Board*, 1972(1), jan 2006.
15. B. Kanagal and A. Deshpande. Online Filtering, Smoothing and Probabilistic Modeling of Streaming data. In *ICDE*, 2008.
16. G. Kellaris, N. Pelekis, and Y. Theodoridis. Trajectory Compression under Network Constraints. In *SSTD*, 2009.
17. E. Keogh, S. Chu, D. Hart, and M. Pazzani. An Online Algorithm for Segmenting Time Series. In *ICDM*, 2001.
18. N. Kiukkoneny, J. Blom, O. Dousse, D. Gatica-Perez, and J. Laurila. Towards Rich Mobile Phone Datasets: Lausanne Data Collection Campaign. In *ICPS*, 2010.
19. Y. Kotidis, V. Vassalos, A. Deligiannakis, V. Stoumpos, and A. Delis. Robust management of outliers in sensor network aggregate queries. In *MobiDE*, 2007.
20. J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory Clustering: a Partition-and-Group Framework. In *SIGMOD*, 2007.
21. Z. Li, B. Ding, J. Han, R. Kays, and P. Nye. Mining Periodic Behaviors for Moving Objects. In *KDD*, 2010.
22. G. Marketos, E. Frentzos, I. Ntoutsi, N. Pelekis, A. Raffaetà, and Y. Theodoridis. Building real-world trajectory warehouses. In *MobiDE*, 2008.
23. N. Meratnia and R. A. de By. Spatiotemporal Compression Techniques for Moving Point Objects. In *EDBT*, 2004.
24. A. T. Palma, V. Bogorny, B. Kuijpers, and L. O. Alvares. A Clustering-based Approach for Discovering Interesting Places in Trajectories. In *SAC*, 2008.
25. N. Pelekis, E. Frentzos, N. Giatrakos, and Y. Theodoridis. HERMES: Aggregative LBS via a Trajectory DB Engine. In *SIGMOD*, 2008.
26. M. Potamias, K. Patroumpas, and T. Sellis. Sampling Trajectory Streams with Spatiotemporal Criteria. In *SSDBM*, 2006.
27. J. A. M. R. Rocha, V. C. Times, G. Oliveira, L. O. Alvares, and V. Bogorny. Db-Smot: a Direction-Based Spatio-Temporal Clustering Method. In *Intelligent Systems*, 2010.
28. N. Schüssler and K. W. Axhausen. Processing GPS Raw Data Without Additional Information. *Transportation Research Record: Journal of the Transportation Research Board*, 8, 2009.
29. S. Spaccapietra, C. Parent, M. L. Damiani, J. A. de Macedo, F. Porto, and C. Vangenot. A Conceptual View on Trajectories. *Data and Knowledge Engineering*, 65(1), 2008.
30. Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, and A. Karl. SeMiTri: A Framework for Semantic Annotation of Heterogeneous Trajectories. In *EDBT*, 2011.
31. Z. Yan, C. Parent, S. Spaccapietra, and D. Chakraborty. A Hybrid Model and Computing Platform for Spatio-Semantic Trajectories. In *ESWC*, 2010.
32. Z. Yan, L. Spremic, D. Chakraborty, C. Parent, S. Spaccapietra, and A. Karl. Automatic Construction and Multi-level Visualization of Semantic Trajectories. In *GIS*, 2010.
33. Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.-Y. Ma. Understanding transportation modes based on GPS data for web applications. *Transactions on the Web (TWEB)*, 4(1), 2010.