J Intell Inf Syst
DOI 10.1007/s10844-011-0159-2

Visually exploring movement data via similarity-based analysis

Nikos Pelekis · Gennady Andrienko · Natalia Andrienko · Ioannis Kopanakis · Gerasimos Marketos · Yannis Theodoridis

Received: 20 May 2010 / Revised: 30 March 2011 / Accepted: 30 March 2011

Abstract Data analysis and knowledge discovery over moving object databases discovers behavioral patterns of moving objects that can be exploited in applications like traffic management and location-based services. Similarity search over trajectories is imperative for supporting such tasks. Related works in the field, mainly inspired from the time-series domain, employ generic similarity metrics that ignore the peculiarity and complexity of the trajectory data type. Aiming at providing a powerful toolkit for analysts, in this paper we propose a framework that provides several trajectory similarity measures, based on primitive (space and time) as well as on derived parameters of trajectories (speed, acceleration, and direction), which quantify the

N. Pelekis (✉)
Department of Statistics and Insurance Science, University of Piraeus, Piraeus, Greece
e-mail: npelekis@unipi.gr

G. Andrienko · N. Andrienko
Fraunhofer Institute Intelligent Analysis and Information Systems, Sankt Augustin, Germany

G. Andrienko
e-mail: gennady.andrienko@iais.fraunhofer.de

N. Andrienko
e-mail: natalia.andrienko@iais.fraunhofer.de

I. Kopanakis
Tech. Educational Institute of Crete, Ierapetra Crete, Greece
e-mail: i.kopanakis@emark.teicrete.gr

G. Marketos · Y. Theodoridis
Department of Informatics, University of Piraeus, Piraeus, Greece

G. Marketos
e-mail: marketos@unipi.gr

Y. Theodoridis
e-mail: ytheod@unipi.gr

Published online: 19 April 2011

distance between two trajectories and can be exploited for trajectory data mining, including clustering and classification. We evaluate the proposed similarity measures through an extensive experimental study over synthetic (for measuring efficiency) and real (for assessing effectiveness) trajectory datasets. In particular, the latter could serve as an iterative, combinational knowledge discovery methodology enhanced with visual analytics that provides analysts with a powerful tool for "hands-on" analysis for trajectory data.

Keywords Trajectory databases · Similarity measures · Visual analytics

1 Introduction

A Moving Object Database (MOD) consists of spatial and temporal information about objects whose location changes over time (e.g. moving humans or vehicles). MOD management has emerged due to the integration of positioning technologies into wireless devices that appears nowadays, and has posed great challenges to the data mining community (Giannotti and Pedreschi 2008). One of the research problems is how to measure the similarity between trajectories of moving objects. Generally, the notion of *similarity* plays an important role in exploratory analysis of large data collections. Visualization and interactive techniques, the primary instruments of data exploration, do not straightforwardly scale to increasing amounts of data. This is especially true for mobility data: even a few trajectories of moving objects (i.e., the sequences of their locations with respect to time) when projected on a 2D map or visualized in a 3D view may be hard to perceive and analyze due to their high overlap. Visual Analytics (VA) (Thomas and Cook 2005), a research area that has recently emerged in response to the demand from real-life applications, looks for synergies between visual and computational techniques enabling human analysts to make sense from large amounts of data (Keim 2005). One of the possible approaches is based on *clustering*, i.e., grouping data items with respect to their similarity and considering groups (represented in a summarized way) instead of individual items. For this purpose, the analyst needs appropriate methods for assessing the similarity between two items.

A VA framework for the analysis of movement data is proposed in (Andrienko et al. 2007). One of its principal components is a spatial clustering tool, which groups trajectories by similarity. Trajectories may be considered as similar in a number of different respects: they may fully or partly coincide in space, or just have similar shapes, or have common starts and/or ends; they may be fully or partly synchronous, or they may be disjoint in time but with similar dynamic behaviour (speed, acceleration, etc.). It depends on the application and goals of analysis, which of these respects are relevant. Therefore, the clustering tool is organized in such a way that cluster building is separated from the computation of distances and neighbourhood. Using the tool implies the availability of a kit of similarity measures, or *trajectory distance functions*, where each function assesses distances between trajectories in its own way. The analyst may choose a suitable distance function depending on the goal of analysis or try different functions for a more comprehensive understanding of the data.

Furthermore, Andrienko et al. (2007) and Rinzivillo et al. (2008) advocate a kind of analysis called *progressive clustering*: the analyst obtains clusters by means of one of the distance functions and re-applies the clustering tool to a selected cluster or a few clusters using another distance function (or different parameter settings), going deeper and deeper in her analysis.

The efficient support of trajectory similarity search in MOD is very important for the quality of data. This justifies the fact that during the last decade there has been a lot of work in the literature regarding trajectory similarity search (Agrawal et al. 1993; Korn et al. 1997; Yi et al. 1998; Chan and Fu 1999; Vlachos et al. 2002b; Chen and Ng 2004; Chen et al. 2005), to cite a few. The common characteristic of those works is that they focus on the movement shape of the trajectories, which are usually considered as 2D or 3D time series data. In other words, what is important in the above cited works is only the sequence of the sampled positions while the absolute information over the temporal dimension is ignored, leaving this information out of the knowledge discovery process. In real world applications though, trajectories are represented by finite sequences of time-referenced locations. What is more, such a sequence may result from *time-based* (e.g. every 30 s), *change-based* (e.g. when the location of an entity deviates from the previous one by a given threshold), *location-based* (e.g. when a moving object is close to a sensor), *event-based* recording (e.g. when a user requests for localization), or combinations of these basic approaches. A different perspective is required therefore, capable of coping with real-world application scenarios.

In this work, we study the problem of trajectory similarity search in MOD, where, given the trajectories of two moving objects we detect and quantify their (dis-)similarity or distance. Having in hand a powerful set of distance operators, each of them describing semantically different interrelation properties between trajectories, we also investigate their utilization in data mining (clustering and classification) tasks and exploratory analysis, which fundamentally rely on the notion of distance among the data under analysis.

To the best of our knowledge, there is no related work on defining a palette of different similarity functions based on these underlying features of the trajectories. Instead, our approach takes under consideration various factors characterizing a trajectory (locality, temporality, directionality, rate of change) and formulates a flexible framework for the comparison of trajectories based on the above factors. Furthermore, the incremental and local nature of the defined distance operators makes them capable of comparing trajectories locally, identifying similarities even in portions of their route. This is in contrast to other approaches that only define global metrics for the whole lifespan of the trajectories.

This paper improves and extends (Pelekis et al. 2007). In outline, the major contributions of this paper are the following:

- We consider several types of similarity between trajectories, focusing on the most interesting motion properties inherent in MOD, i.e. (a) fundamental types of (time-aware or not) spatial similarity search over trajectories and (b) interesting variations exploiting on derived information (speed, acceleration, direction). For each type we formally define appropriate distance operators and propose respective query processing algorithms, which include major improvements with respect to those originally proposed in (Pelekis et al. 2007).
- We study the metric properties of the proposed distance functions.

- Exploiting on a state-of-the-art visual analytics tool (Andrienko et al. 2007), we demonstrate the efficiency of the proposed similarity-based query framework through an iterative knowledge discovery process over a real dataset from the fleet management domain.
- We conduct an extensive experimentation over synthetic trajectory datasets, in order to evaluate our approach with respect to its efficiency as well as its usefulness on data mining (classification and clustering) tasks.

The rest of the paper is structured as follows: Section 2 discusses the motivation for our work as well as related work. In Sections 3 and 4, which constitute the core of the paper, we present formal definitions and algorithms for the proposed distance functions. Section 5 presents the exploitation of the proposed similarity types on a VA application over a real dataset as a proof-of-concept and Section 6 evaluates the efficiency of our proposals through an extensive experimental study over synthetic datasets. Section 7 discusses related work in comparison with our proposal. Finally, Section 8 concludes the paper and provides ideas for future work.

2 Motivation

Through the example that follows we intend to give an idea about the kind of analysis where diverse trajectory similarity measures are required.

Let us consider a city traffic manager concerned with the movement of heavy trucks transporting materials for construction over the city: they produce much noise and air pollution and often obstruct the movement of the public transport. She wants to investigate the current movement of the trucks in order to understand its specifics and then develop a feasible policy (e.g., set of rules for the supply or transportation companies owning the trucks) for alleviating the problems. For developing such a policy, it is necessary to know (a) what are frequent origins and destinations of the trips; (b) how the origins and destinations are connected; (c) what are the coverage areas of different service providers; (d) whether there are distinct traffic zones with different characteristics of the truck movement; (e) what are the typical routes of the trucks of different providers; (f) whether two or more alternative routes between the same locations are used and whether the choice depends on the time; (g) what are the typical speed and acceleration patterns (which are related to the noise and pollution) in different areas, on different routes, and in different times.

To find answers to these questions, the traffic manager uses a suitable dataset consisting of trajectories of multiple trucks that were GPS-tracked over a sufficiently long time period. In our example, we use a dataset with positions of 50 trucks transporting concrete in Athens, Greece, between August and September 2002 (the dataset illustrated in Fig. 1 is publicly available at R-tree Portal 2011). There are 112,300 position records consisting of the truck identifiers, dates and times, and geographical coordinates. The temporal spacing is regular and equals 30 s (i.e. rate of sampling from GPS devices). Out of this raw dataset, a number of 1,100 trajectories can be identified by splitting the recordings of a truck in subsets when a temporal gap larger than 15 min appears between two consecutive recordings.

Since the dataset is quite big, the traffic manager cannot explore in detail each trajectory and therefore uses clustering, which can effectively help her to find answers to her questions. Thus, to find the frequent origins and destinations of the

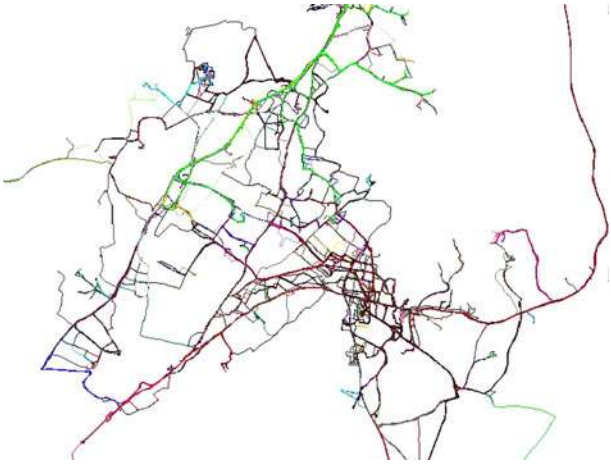

Fig. 1 The 'trucks' dataset (source: R-tree Portal 2011)

trips, the analyst would cluster the trajectories by spatial closeness of their starts and ends. The discovery of starts and ends is an interesting stand-alone problem that is essential for applications interested in the distribution of origins. Although important our approach does not directly address it. However, in the literature, there have been proposed specialised approaches that address this issue (Marketos et al. 2008). After the grouping of trajectories w.r.t. starts and ends, the analyst would focus on particular clusters to investigate their distribution in space and in time, reveal typical and alternative routes, typical dynamics of speed, times and places of stops, etc. This may be done by applying clustering with different similarity measures to the trajectories of the selected clusters, according to the idea of progressive clustering (Rinzivillo et al. 2008).

The analysis process would be appropriately supported by an interactive visual tool, such as the one illustrated in Fig. 2, where the analyst can conveniently select the clusters to view and compare and to analyze further in more detail. The clusters may be represented by different colouring of the trajectory lines (Fig. 2a) or in a schematic form (Fig. 2b and c), giving an idea about the major directions and volumes of the movement.

In this example application and in many others, the analysts need to run a variety of cluster analysis tasks, such as:

- Task 1: *spatiotemporal similarity*: Find clusters of objects that follow similar routes (i.e., projections of trajectories on 2D plane) during the same time interval (e.g. co-location and co-existence from 3 to 6 P.M.) or
- Task 2: *(time-relaxed) spatial-only similarity*: Find clusters of moving objects taking only their route into consideration (i.e., irrespective of time)

as well as variations of the above, such as:

- Task 3: *speed- (or acceleration-) pattern based spatial similarity*: Find clusters of objects that follow similar routes and, additionally, move with a similar speed (or acceleration) pattern, or
- Task 4: *directional similarity*: Find groups of objects that follow a given direction pattern (e.g. first NE and then W), either concurrently or not.

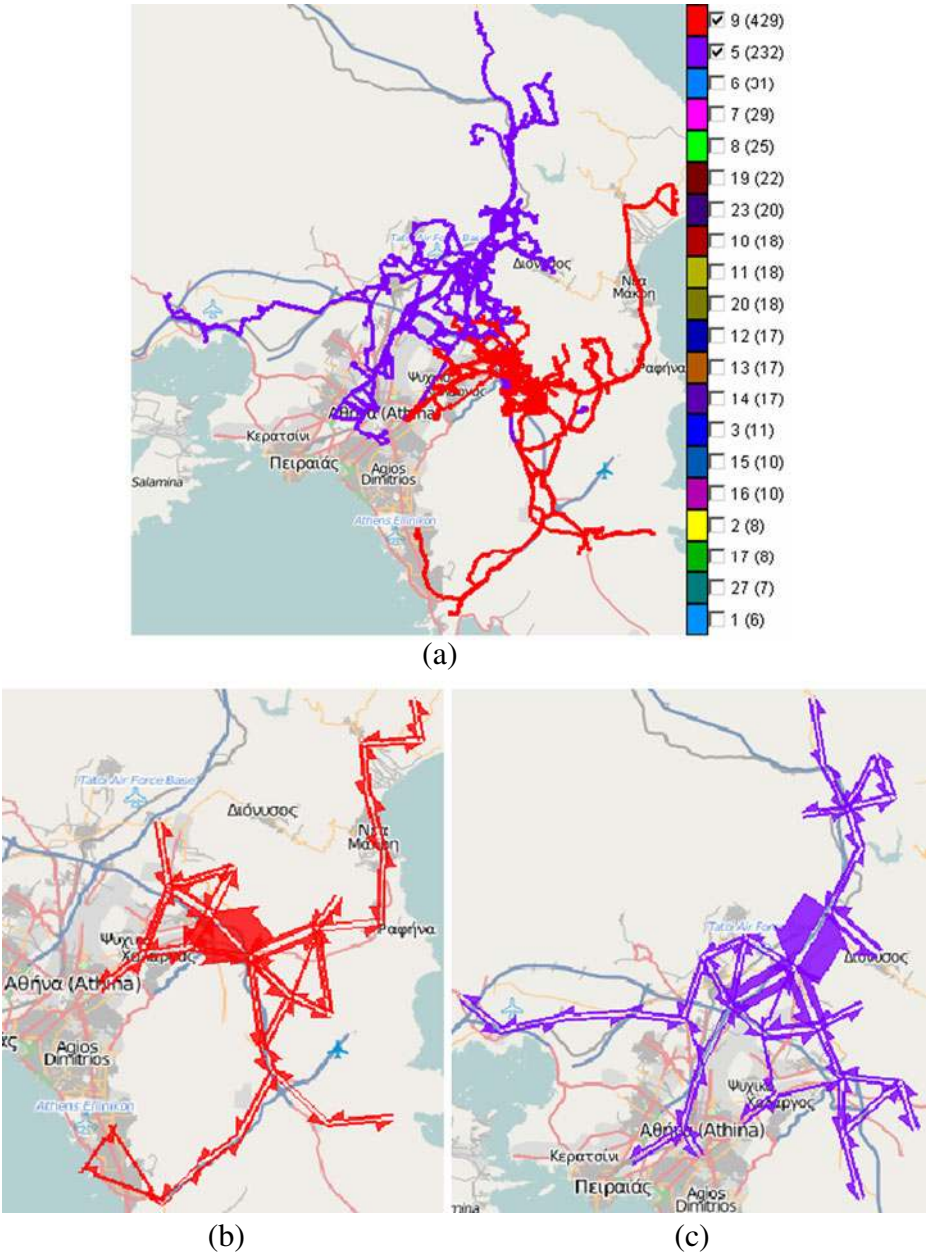

Fig. 2 **a–c** An interactive visual tool for exploring trajectory clusters

Since our framework is based on query processing operators running on top of Hermes MOD engine (Pelekis and Theodoridis 2006; Pelekis et al. 2006, 2008, 2011), the novelty of our approach is augmented by two inter-related facts: (1) the combination of the tasks (using AND/OR clauses) provides analysis functionality

unmatched so far (e.g. "find trajectories that moved closely in space but with very dissimilar speed patterns"); (2) the output of each of the supported operators defines similarity patterns that can be utilized to reveal local similarity features (e.g. "find the most similar portions between two, in general, dissimilar trajectories").

3 Trajectory similarity search

Before we define the different types of similarity search to be addressed in this paper, we first present the notations utilized hereafter. Let D be a database of N moving objects with object ids $\{o_1, o_2, \ldots, o_N\}$. The trajectory T_i of a moving object o_i consists of a sequence of $|T_i|$ *3D Line Segments* (3DLS), where each 3DLS represents the continuous development of the moving object during two distinct sampled locations assuming linear interpolation between the two locations. In other words, the movement of an object from a starting position (x_s, y_s) to an ending position (x_e, y_e) during a time period $[t_s, t_e]$ is described by a linear function of time $f(t)$. Projecting T_i on the spatial 2D plane (temporal 1D line), we get the *route* r_i (the *lifespan* l_i, respectively) of o_i; an example appears in Fig. 3. Moreover, additional motion parameters can be derived by $f(t)$, including speed s, acceleration a, etc. Obviously, no assumptions of equal distanced time intervals between the sampled points are posed.

Definition 1 (most-similar trajectory) Let D be a database of trajectories T_i and Q be a (reference) trajectory consisting of a set of 3DLS, the cardinality of which being $|Q| \geq 1$ and $|T_i| \geq 1 \, \forall T_i \in D$. The *Most-Similar-Trajectory* (MST) S in D with respect to Q is the one that minimizes a distance measure $Dist(Q,T_i)$.

As already argued in the previous section, the distance measure $Dist(Q,T_i)$ is application-driven and may involve any combination of trajectory features, such as spatial projection (route), temporal projection (lifespan), speed, and direction. Taking both route and lifespan into consideration, $Dist(Q,T_i)$ addresses Task 1 presented in Section 2; considering only route, Task 2 is addressed, and so on.

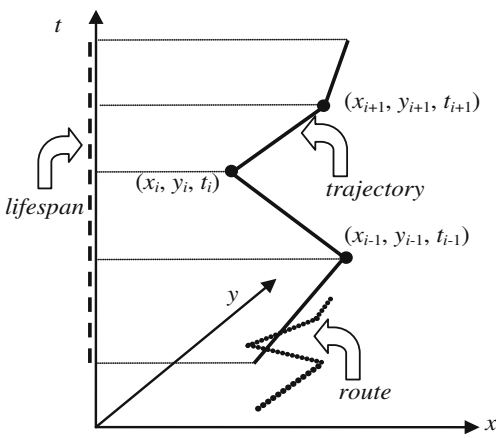

Fig. 3 A trajectory (*solid line*), its route (*dotted line*), and its lifespan (*dashed line*), assuming linear interpolation between sampled positions

Route, lifespan, speed, acceleration, and direction of a moving object trajectory are classified as *motion dependent* parameters. There also exist *data dependent* parameters that affect similarity search, such as length, scale, shift, sampling rate and outliers' existence, which have been addressed as the main research issues in related work. In this paper, we focus on the former class, as we treat the problem from a MOD perspective.

In the rest of this section, we present a palette of distance functions each of which is tailored to support similarity search by giving emphasis on one of the previous mentioned parameters. Before we define the spatiotemporal similarity between trajectories (Section 3.2), we tackle the problem of measuring the spatial similarity of two moving objects (Section 3.1) as a pre-required step.

3.1 (Time-relaxed) spatial similarity

Intuitively, two moving objects are considered *spatially similar* when they move close (i.e., their routes approximate each other) at the same place, irrespective of time. As such, we propose a novel distance operator, called *Locality In-between Polylines* (LIP), which defines a distance function upon the (projected on the Cartesian plane) routes of the trajectories. The idea is to calculate the area of the shape formed by the two 2D polylines that correspond to the routes of the two trajectories. An example is illustrated in Fig. 4 where the five shaded areas are the ones that contribute in *LIP*.

Graphically, the area between two polylines is the one *traversed* by the one polyline when it appropriately moves, shrinks and/or extents itself towards the other polyline so as to perfectly match each other.

Furthermore, in order to employ the area between two polylines as an intuitive measure of their distance we need to account the cases where such a measure is meaningful. Dividing two polylines into appropriate sub-polylines where the area in-between may serve for calculating their distance is critical in our methodology, therefore it is presented first (Section 3.1.1), second, we discuss specific cases (Section 3.1.2) and, third, we propose the actual procedure for calculating the LIP distance measure (Section 3.1.3).

3.1.1 Detecting 'good' vs. 'bad' pairs of segments

Considering the area formed between two polylines as a representative measure of their distance (i.e., dissimilarity) does not always make sense. Definitely, it does so when the two polylines follow, on the average, a stable trend (e.g. like the polylines S and Q, illustrated in Fig. 4) with no dramatic rotations, or if they are rotating, at

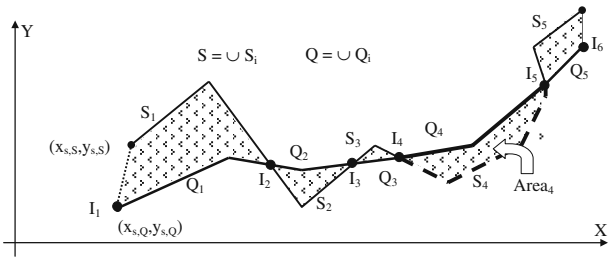

Fig. 4 Illustration of locality in-between polylines (LIP) distance between the routes of two trajectories, S and Q

least they do so by turning similarly during their motion (see the respective S and Q in Fig. 5e). However, there exist other cases where using the area to measure dissimilarity is misleading; for example, let us consider the special cases illustrated in Fig. 5a and b, where the polylines consist of two single segments each. In Fig. 5b, where the segments are heading to the same direction, the area of the vertically stripped polygon is representative of their distance. The same stands for the two out of the three cases depicted in Fig. 5a (see the three triangles formed when connecting S_iS_e segment with the ending point of Q_1Q_2, Q_1Q_3, Q_1Q_4, respectively). In other

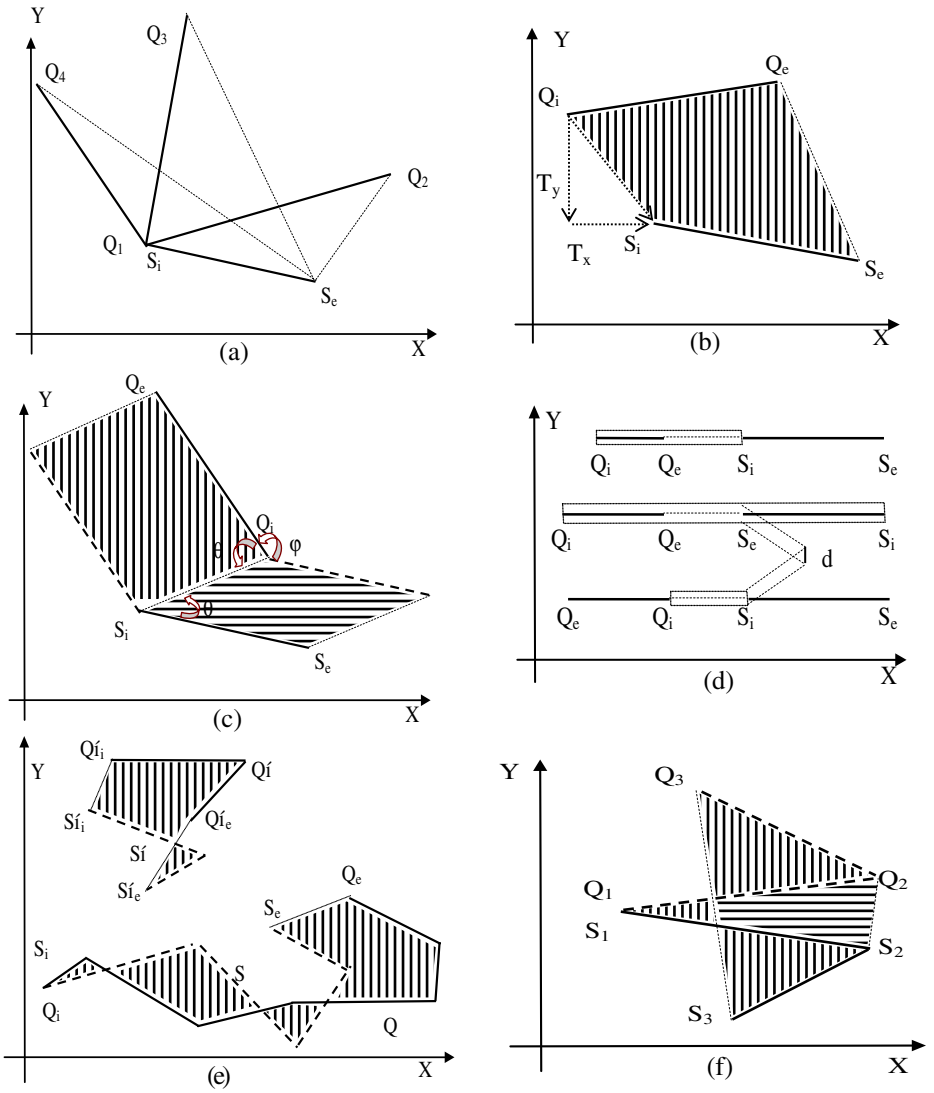

Fig. 5 **a–f** Special cases for LIP distance

words, in the cases where the angle φ between the two segments is less than 90°, the area of the triangle formed by the two segments is a meaningful measure (intuitively, for segments of a certain length the area is proportional to the angle).

However, if φ exceeds 90° (e.g. see triangle $S_i S_e Q_4$ in Fig. 5a) the area of the triangle decreases down to zero (in other words, in this case the area is inversely proportional to the angle); several 'bad' cases of this type are also illustrated in Fig. 5d. Recall, however, that in the latter case there exists a counterpart $\varphi' = 180° - \varphi < 90°$, resulting in regions with equal area when simply connecting their ending points (e.g. the area of triangle $S_\mathrm{i} S_\mathrm{e} Q_4$ is equal to the one of triangle $S_\mathrm{i} S_\mathrm{e} Q_2$ in Fig. 5a), while the segments of the second case should intuitively have smaller distance.

In the general case, where the polylines consist of two or more segments, a procedure is required that would traverse these segments trying to calculate a meaningful in-between area by taking into account the relative direction of the two polylines. For example, in Fig. 5f, polyline S rotates clockwise while Q rotates counter clockwise. Connecting their ending points would result in a set of regions (i.e., the vertically stripped regions), the area of which is a rather meaningless distance measure (or even indefinite, in extreme cases). Intuitively, the traversal algorithm should identify this abnormal case and split the two trajectories (at point S_2 and Q_2, respectively) into two sub-trajectories, which pair-wise formulate the simple polygons $S_1 S_2 Q_2$ and $S_2 S_3 Q_3 Q_2$ that can act as rational indicators of their distance. In this way, the horizontally stripped region is also taken into account. The above discussion makes clear that the key issue that results in abnormal cases is the fact that *connecting the ending points the resulting regions are self-intersected* (i.e., non-simple polygons). The same situation might happen not only when the polylines turn in different directions but also when they turn similarly as in the case of Q' and S' depicted in Fig. 5e.

Technically speaking, *in order to calculate a meaningful area between two polylines, they should be segmented into a sequence of sub-polylines, for each pair of which there is an appropriate way to measure the area in any of the above mentioned cases*. This segmentation should be performed during traversing of the segments of the polylines as soon as a *goodness criterion* is violated. The basic property that this criterion should satisfy is that the regions formed by the pair of sub-polylines (interconnecting their starting and ending points, if necessary) must be *simple polygons*. However, as already depicted, this is a necessary but not sufficient condition. The notion of direction should be involved to assure that trajectories do not fall in the counter intuitive cases previously described, and when these are identified follow a different tactic to measure the in-between area. Below, we formalize such a goodness criterion, called *LIP criterion*.

Initially, in order to take into account the direction of two trajectories we define the *local directional* distance *DIR* between two segments of Q and S characterized by the angle φ they form.

Definition 2 (local directional distance between two segments) The local directional distance $DIR(Q_\mathrm{q}, S_s)$ between two segments Q_q and S_s, $Q_\mathrm{q} \in Q$ and $S_s \in S$, forming an angle $\varphi \in [0°, 180°]$ is defined as follows:

$$DIR\left(Q_q, S_s\right) = \frac{1 - \cos\left(\varphi\right)}{2} \tag{1}$$

with its value ranging from 0 (total similarity, in case $\varphi = 0°$) to 1 (total dissimilarity, in case $\varphi = 180°$).

Let us now formally define the so-called *LIP criterion* that, in the discussion that will follow, will classify a pair of segments as either *good* or *bad* with respect to whether this pair satisfies this criterion or not.

Definition 3 (LIP criterion for classifying two segments as either good or bad) Given the routes (polylines) Q and S of two trajectories, and the pair (Q_q, S_s) of segments $Q_q \in Q$, $S_s \in S$, which are the subsequent of two already certified sub-polylines Q'_j and S'_j ($|Q'_j| \geq 0$, $|S'_j| \geq 0$); two polylines are considered certified if their segments have already passed the LIP criterion successfully. The pair (Q_q, S_s) satisfies the *LIP criterion* w.r.t. to Q'_j and S'_j and is marked as *good* if:

(a) (*local directional property*) the local direction between the two segments forms an angle $0° < \varphi < 90°$, i.e., $0 < DIR(Q_q, S_s) < 0.5$, and
(b) (*local simplicity property*) if Q'_j and S'_j are not empty (i.e., $|Q'_j| > 0$, $|S'_j| > 0$), the segment $QS_{qs} = [(Q_q.x_e, Q_q.y_e), (S_s.x_e, S_s.y_e)]$), which is the segment that connects the ending points of the under examination segments Q_q and S_s, crosses neither $Q'_{j\cup} Q_q$ nor $S'_{j\cup} S_s$;

To exemplify the above definition, let us consider the two polylines Q and S depicted in Fig. 6. In order to apply the LIP criterion for the first segments Q_1 and S_1 we only need to consider the first part of the definition (Q'_j and S'_j are empty since no segment of Q and S has been checked yet). The result of the examination is positive since it holds that $0 < DIR(Q_1, S_1) < 0.5$, so Q_1 and S_1 are considered certified sub-polylines for the subsequent segments. Assuming an iterative procedure (which will be described in detail in Section 3.1.3), we come to a point that we have marked pairs (S_i, Q_i), $1 \leq i \leq 3$, as pairs of *good* segments. For instance, note that $0 < DIR(Q_3, S_3) < 0.5$ and QS_{33} crosses neither polyline $\{Q_1, Q_2, Q_3\}$ nor $\{S_1, S_2, S_3\}$. At this point the procedure identifies (S_4, Q_4) as a pair of *bad* segments, since both conditions of *LIP criterion* fail (i.e., $DIR(Q_4, S_4) < 0.5$ and QS_{44} crosses $\{S_1, S_2, S_3, S_4\}$ at S_3; however, note that only the first condition needs to be tested).

Having defined the LIP criterion which classifies pairs of segments as either *good* or *bad*, we define a simple yet effective criterion that segments the routes (polylines) Q and S of two trajectories into two respective sequences of sub-polylines whose

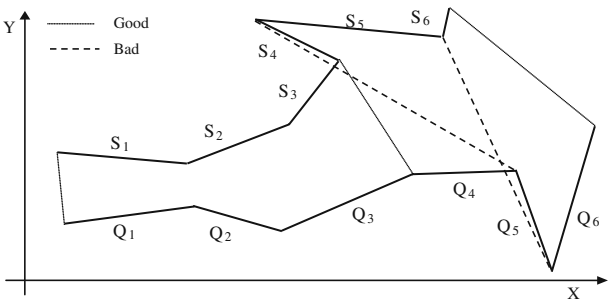

Fig. 6 Examples of 'good' and 'bad' segments

one-by-one combination forms a sequence of pairs of sub-polylines, each one satisfying this criterion. The idea is that during traversing polylines Q and S we may identify the so-called *bad* segments that violate the *local directional* and *local simplicity* properties (defined above) of the under-construction regions, as they exhibit diverging rotation in contrast to their coupling trajectory. As soon as we detect these segments, we segment the trajectories at the initial points of the *bad* segments. Then, we calculate the LIP for the already investigated portions of the polylines that for sure satisfy our conditions. Afterward, we deal with the *bad* segments and we continue recursively in a similar manner with the remaining portions of the initial polylines. (Note here that concurrently traversing the two polylines does not necessarily involve a one-by-one comparison between the segments of Q and the respective segments of S. Instead, a segment from the one side could be compared with an already examined segment from the other side, and this happens in order for the algorithm to process similar lengths, thus allowing us to handle datasets collected with different sampling rates or trajectories resulted from motions of different speeds (though same sampling rates). This procedure will be presented in detail in Section 3.1.3).

Definition 4 (trajectory segmentation according to LIP criterion) Given the routes (polylines) Q and S of two trajectories consisting of $|Q|$ and $|S|$ segments, respectively, we define their segmentation that does not violate the LIP criterion, as the one that results in two corresponding sequences Q' and S' of z sub-trajectories, $Q' = \langle Q'_1,\ Q'_2, \ldots, Q'_z \rangle$ and $S' = \langle S'_1, S'_2, \ldots, S'_z \rangle$, so that:

(a) The union of the sub-trajectories $Q'_{\mathrm{j}} \in Q'$, $S'_{\mathrm{j}} \in S'$, $j = 1, \ldots, z$, forms the initial trajectories Q and S, i.e., $\bigcup_{j=1}^{z} Q'_j = Q, \bigcup_{j=1}^{z} S'_j = S$;

(b) Each sub-trajectory $Q'_{\mathrm{j}} \in Q'$, $S'_{\mathrm{j}} \in S'$, $j = 1, \ldots, z$, which consists of a subset of consecutive segments $Q_{\mathrm{q,j}}, S_{\mathrm{s,j}} \left(q = 1, \ldots, \left|Q'_{\mathrm{j}}\right| \text{ and } s = 1, \ldots, \left|S'_{\mathrm{j}}\right| \right)$ of Q and S, respectively, is a non self-intersecting sub-trajectory, except only at the ending and starting points of consecutive segments, i.e. for Q'_{j}, $\forall q'$, $q = 1, ..., |Q'_j|$ if $q' \neq q + 1$ then $Q_{q,j} \bigcap Q_{q',j} = \varnothing$ else $Q_{q,j} \bigcap Q_{q',j} = \left(x_{s,Q_{q',j}}, y_{s,Q_{q',j}}\right)$. Similarly for S'_{j}. Obviously $\bigcup_{q=1}^{|Q'_j|} Q_{q,j} = Q'_j$ and $\bigcup_{s=1}^{|S'_j|} S_{s,j} = S'_j$;

(c) The $z \geq 1$ pairs of sub-trajectories $pair_{\mathrm{j}} = \left(Q'_{\mathrm{j}},\ S'_{\mathrm{j}}\right)$, $j = 1, \ldots z$, corresponding to $z-1$ splitting points, occur at the so-called *bad segments*, namely, those segments that violate the *LIP criterion* as it was defined in Definition 3, i.e., $Q_{|Q'_j|,j}$ and $S_{|S'_j|,j}$, $j = 1, \ldots, z-1$.

(d) If the length of the route of Q is larger than the length of S (i.e. $r_Q > r_S$), then the final segment of S (i.e. $S_{|S|}$) is used as the coupling sub-trajectory in the LIP criterion evaluation, against the remaining sub-trajectory of Q. This implies that there are $\xi \leq z$ pairs that are formed as follows: $pair_{\mathrm{j}} = \left(Q'_{\mathrm{j}}, S_{|S|}\right)$, $j = z - \xi + 1, \ldots, z$. Similarly, if $r_Q < r_S$.

Note that, at the present point, the above definition illustrates in a formal way the properties of the final pairwise segmentation of two trajectories. The detailed

description and the algorithmic steps taking into advantage of the above definition are presented in the following two subsections.

3.1.2 Calculating LIP distance

Combining Definitions 3 and 4 allows us to calculate the distance between the routes of two trajectories by computing the area between pairs of sub-polylines $pair_{\mathrm{j}} = \left(Q'_{\mathrm{j}}, S'_{\mathrm{j}}\right)$, $j = 1, \ldots, z$, which either satisfy the *LIP criterion* (i.e., *good* case; in this case, Q'_j, S'_j are two polylines, each consisting of one or more segments) or not (i.e., *bad* case; in this case, Q'_{j}, S'_{j} are two single segments). Each case is handled differently. Subsequently, we present our approach for calculating the area for a *good pair*$_j$, suppressing for the moment the way we use the *LIP criterion* in a generic algorithm.

First, we should note that in the above discussion (Definitions 3 and 4) there is no assumption whether a segment of sub-polyline Q'_{j} intersects/crosses a segment of sub-polyline S'_{j} in a $pair_{\mathrm{j}}$ or not; there is only a requirement that each of Q'_{j}, S'_{j} is not self-intersecting. Furthermore, any two segments belonging to Q'_{j} andS'_{j}, respectively, will fall in one of the following three cases: a) they are disjoint (having no intersection points), b) they are not disjoint and not collinear (presenting only one intersection point), and c) they are collinear and they overlap (in this case, there are at most two intersection points, namely the starting and ending points of their common part; if their common part is a single point then this is their unique intersection point).

Given the above, for the 'good' case in which the *LIP criterion* is satisfied between $pair_{\mathrm{j}} = \left(Q'_{\mathrm{j}}, S'_{\mathrm{j}}\right)$, and in order to work with closed polygons, we add two additional segments connecting the starting and ending points of the sub-polylines (see dashed segments in Fig. 4). Obviously, these segments do not cross any of Q'_{j}, S'_{j} due to the LIP criterion. Considering that these additional segments are extensions of one of the sub-polylines (hereafter and without loss of generality, let us assume that these segments are extensions of S'_{j}), then regarding the first of these segments, its initial point is now considered to be the initial point of both polylines, while regarding the second segment, its ending point coincides with the last point of the other sub-polyline (i.e. Q'_{j}). Obviously, the above discussed points imply that Q'_{j} andS'_{j} have at least two intersection points.

In the general case, Q'_{j} andS'_{j} may intersect in $n \geq 2$ intersection points $I_1, I_2, \ldots, I_{\mathrm{n}}$ (with I_1 andI_{n} being the above-mentioned intersection points), and the *LIP criterion* guarantees the formulation of a simple polygon between two consecutive intersection points. As such, in the general case there is a need to calculate the area of a series of $n-1$ simple polygons. Given the above, below we formally define the LIP distance measure for the case of two sub-trajectories that satisfy the LIP criterion.

Definition 5 (LIP distance—'good' pair) The *Locality In-between Polylines* distance for a pair $pair_{\mathrm{j}} = (Q'_{\mathrm{j}}, S'_{\mathrm{j}})$ of sub-polylines that (a) satisfy the LIP criterion, and (b) present $n \geq 2$ intersection points I_1, I_2, … I_{n}, where I_1 andI_{n} are artificially introduced after interconnecting at their initial and final points, is defined as follows:

$$LIP_{good}\left(Q'_j, S'_j\right) = \sum_{\forall\, polygon_i \in \Pi} LIP_i \tag{2}$$

where LIP_{i} for a *simple polygon*$_{\mathrm{i}}$ is defined as:

$$LIP_i = Area_i \cdot w_i \tag{3}$$

with $Area_{\mathrm{i}}$ denoting the area of *polygon*$_{\mathrm{i}}$ belonging to the set Π of *simple polygons*, and w_i being the contributions (weights) of each polygon area in the overall distance. The set Π is defined as:

$$\begin{aligned}\Pi = \{&poly(I_k, ipoints(Q'_j, I_k, I_{k+1}), I_{k+1}, ipoints(S'_j, I_{k+1}, I_k), I_k)\\ &|\forall k = 1, .., n-1\}\end{aligned} \tag{4}$$

where *poly*() is a polygon constructor that takes as input a set of points and constructs a closed polygon, and *ipoints*() is a function that returns the *intermediate* sampled points of a trajectory (i.e., first argument) between two points (i.e., second and third arguments) in the sequence implied by them. Regarding the $Area_{\mathrm{i}}$ of each *polygon*$_{\mathrm{i}}$, it is calculated as usual.

Finally, if $r_{\mathrm{Q_{k,k+1}}}$ and $r_{\mathrm{S_{k,k+1}}}$ are the lengths of the routes traversed by Q'_{j} and S'_{j}, respectively, between two intersection points I_{k} and $I_{\mathrm{k+1}}$, i.e., the portions of Q and S that participate in the construction of *polygon*$_i$, and if r_Q and r_S are the lengths of the whole routes of Q and S, respectively, the contribution $w_{\mathrm{i}} \in [0..1]$ of *polygon*$_{\mathrm{i}}$ in $LIP(Q'_{\mathrm{j}}, S'_{\mathrm{j}})$ is:

$$w_i = \frac{r_{\mathrm{Q_{k,k+1}}} + r_{\mathrm{S_{k,k+1}}}}{r_{\mathrm{Q}} + r_{\mathrm{S}}} \tag{5}$$

As already mentioned, intuitively the area between two polylines is the one traversed by the one polyline towards the other polyline in a way so as to perfectly match. When the *LIP criterion* is not satisfied, following the approach of *good* pairs would lead to an area that is not a meaningful distance between the two sub-polylines. For the 'bad' case, where $pair_{\mathrm{j}} = (Q'_{\mathrm{j}}, S'_{\mathrm{j}})$ does not satisfy the LIP criterion (recall that, in this case, the two sub-polylines consist of a single segment each), we follow a two-step approach. In the first step, we translate each segment towards the other as to have common starting points, while at the second step we take into advantage the obtuse angle formed between the two segments by using their monotonically increasing directional distance (1).

Regarding the first step, such a translation shifts the segment by a different constant at each dimension. Figure 5b illustrates the translation of $Q_{\mathrm{i}}Q_{\mathrm{e}}$ towards $S_{\mathrm{i}}S_{\mathrm{e}}$ by translating it by a vector $\vec{T}_x + \vec{T}_y$ in order to match the starting point of $Q_{\mathrm{i}}Q_{\mathrm{e}}$ to that of $S_{\mathrm{i}}S_{\mathrm{e}}$. This initial step comes with a cost, which, in terms of area, is equal to the traversed region during translation of the segment, and which is proportional to the lengths of the segments and the length of vector $\vec{T}_x + \vec{T}_y$. Formally:

Definition 6 (traversed area) The traversed area that is required in order to make the segments of a *bad pair*$_{\mathrm{j}} = (Q'_{\mathrm{j}}, S'_{\mathrm{j}})$, to have common starting points is the area of the maximum region traversed by either translating Q'_{j} towards S'_{j} or S'_{j} towards Q'_{j}. This area corresponds to the area of the parallelogram which is created by translating the segment along vector $\vec{T}_x + \vec{T}_y$, which connects the beginning of Q'_{j} (S'_{j}) with the initial point of S'_{j} i.e. $(S_{1,j}.x_i, S_{1,j}.y_i)$ (Q'_{j} i.e. $(Q_{1,j}.x_i, Q_{1,j}.y_i)$, respectively). The area of the parallelogram equals $\left|\vec{T}_x + \vec{T}_y\right| \cdot r_{Q'_j} \cdot \sin(\theta)$, where $\left|\vec{T}_x + \vec{T}_y\right|$ and $r_{Q'_j}$ are

the lengths of the translation vector and the translated segment Q'_j, while θ is the angle between them (see Fig. 5c). In the degenerated case that $\theta = 0°$ (or 180°), where the area of the region traversed is zero, we set the translated area to be equal to $d_T = \left(\left|\vec{T}_x + \vec{T}_y\right| + \left|r_{Q'_j} - r_{S'_j}\right|\right) \cdot d$ that is the length of the translation vector plus the absolute difference of the lengths of the segments, multiplied by some minimal distance d, which corresponds to the vertical distance of the segments as if they were not colinear (assuming e.g. an angle θ a bit bigger (less) than 0° (180°, respectively; see Fig. 5d). The above are formalized as follows:

$$translate\left(Q'_j, S'_j\right) = \max\left\{TA\left(Q'_j, \left(S_{1,j}.x_i, S_{1,j}.y_i\right)\right), TA\left(S'_j, \left(Q_{1,j}.x_i, Q_{1,j}.y_i\right)\right)\right\} \quad (6)$$

where

$$TA\left(Q'_j, \left(S_{1,j}.x_i, S_{1,j}.y_i\right)\right) = \begin{cases} \left|\vec{T}_x + \vec{T}_y\right| \cdot r_{Q'_j} \cdot \sin(\theta), & \text{if } 0° < \hat{\theta} < 180° \\ d_T, & \text{if } \hat{\theta} = 0° \text{ or } \hat{\theta} = 180° \end{cases} \quad (7)$$

Regarding the second step, the idea is to use the *DIR* distance between the two segments so as to overcome the unintuitive case of Fig. 5a. The following definition presents the way we handle a *bad* pair of segments:

Definition 7 (LIP distance—'bad' pair) The *Locality In-between Polylines* distance for a pair of segments $pair_{\mathrm{j}} = (Q'_{\mathrm{j}}, S'_{\mathrm{j}})$ that does not satisfy the LIP criterion is:

$$LIP_{bad}\left(Q'_j, S'_j\right) = \left(translate\left(Q'_j, S'_j\right) + \mathrm{r}_{Q'_{\mathrm{j}}} \cdot \mathrm{r}_{S'_{\mathrm{j}}} \cdot DIR\left(\mathrm{Q}'_{\mathrm{j}}, \mathrm{S}'_{\mathrm{j}}\right)\right) \cdot w \quad (8)$$

where $r_{Q'_j}$ and $r_{S'_j}$ are the lengths of Q'_{j} and S'_{j}, respectively, and w is defined similarly as w_{i} of Definition 5.

$$w = \frac{r_{Q'_j} + r_{S'_j}}{r_{\mathrm{Q}} + r_{\mathrm{S}}} \quad (9)$$

Recalling Fig. 5, we can argue that the above definitions cover all special cases illustrated there. In particular, the three different cases of Fig. 5a are treated uniformly (note that translation area is zero). When the angle $\varphi = 90°$, $DIR\left(\mathrm{Q}'_{\mathrm{j}}, \mathrm{S}'_{\mathrm{j}}\right) = 0.5$ corresponding to the area of the orthogonal triangle, while when $\varphi > 90°$, $DIR\left(\mathrm{Q}'_{\mathrm{j}}, \mathrm{S}'_{\mathrm{j}}\right) > 0.5$ intuitively results in an even bigger area. In this case the value of LIP is up to $\mathrm{r}_{Q'_{\mathrm{j}}} \cdot \mathrm{r}_{S'_{\mathrm{j}}}$, which is the case when $\varphi = 180°$. Figure 5c depicts the traversed area that is the maximum between the vertical and the horizontal stripped regions. Figure 5d illustrates three special cases, i.e., objects move towards the same direction, as well as the same versus opposite destinations. Note that in these cases LIP acts similarly to Euclidean distance as it depends on the length of the translation vector and the absolute difference of the lengths of the segments. Finally, the case where $\varphi = 0°$ is handled as a *bad* case. More specifically, if the two segments are moving parallel in space ($\theta \neq 0°$) LIP_{bad} counts only the translation area (as the *DIR* distance is zero in this case), which is the same as following the approach of a *good* pair of segments. On the other hand, if the two segments are moving along the same line ($\theta = 0°$), LIP_{bad} is d_T, which implies an area analogous to $\left|\vec{T}_x + \vec{T}_y\right| + \left|r_{Q'_j} - r_{S'_j}\right|$ (i.e. zero for

identical segments—equi-length segments that start together and move on the same line—and non-zero for non identical segments).

The procedure for computing LIP distance between a pair of polylines Q' and S' is illustrated in Fig. 7. In summary, the procedure for computing LIP for a $pair_j$ (below, subscript j is suppressed for clarity) depends on whether Q' and S' satisfy the *LIP criterion* (lines 3–17) or not (lines 1–2). In the former case, the algorithm initially finds possible intersection points between the two sub-trajectories (line 4) and then iteratively detects the above discussed simple polygons, whereas each polygon is constructed as a list of points. This list is separated into two sets. The first set consists of points that reside on Q' (solid line in Fig. 4), while the second set consists of points that reside on S' (dashed line). The algorithm follows forwardly the solid line, adding points from Q' to the current polygon, and when finding an intersection point, follows backwardly the dashed line, adding points from S' to the polygon until the previous intersection point is visited. Intersection points I_k between Q' and S' are computed and stored once in a priority queue PQ, where priority implies an ordering of I_k in time axis. Together with the priority queue, a positioning array (PA) is maintained keeping the line segments of both Q' and S' where upon the intersection points reside. *LIPgram* is a list maintaining information for each polygon.

Regarding the complexity of LIP Algorithm, we note the following: Assuming $N = |Q'| + |S'|$ line segments and K intersection points, finding intersection points I_k between Q' and S' (line 4) corresponds to the red-blue intersection problem (Chan 1994), which can be solved in $O(N\log N + K)$ time using $O(N + K)$ space (Chazelle and Edelsbrunner 2002). PQ stores K elements (equals to the number

Algorithm **LIP**(*Q'*, *S'*, *case*)
Input: *Q'* and *S'* are polylines that either satisfy the *LIP criterion* or not; the latter is signalled by the *case* flag
Output: The area between polylines *Q'* and *S'*

```
01. IF case == 'bad' THEN
02.   RETURN LIP_bad(Q',S')  // See Eq. 8
03. ELSE // case == 'good'
      // Find intersection/crossing points I_k
04.   PQ=IntersectPoints(Q',S',PA)
      // Form polygons between consecutive I_k
05.   FOR k = 1 to |PQ| - 1
06.     polygon = ∅
07.     start = pos(PA, Q', PQ(k))
08.     end = pos(PA, Q', PQ(k+1))
09.     T1 = ipoints(Q',start,end)
10.     start = pos(PA, S', PQ(k+1))
11.     end = pos(PA, S', PQ(k))
12.     T2 = ipoints(S',start,end)
13.     polygon =poly(PQ(k),T1,PQ(k+1),T2,PQ(k))
14.    update(LIPgram,polygon,k,k+1, r_Q(k,k-1), r_S(k,k-1))
15.   NEXT
16.   RETURN sum(LIPgram) // See Eq. 2
17. END IF
```

Fig. 7 LIP algorithm

of the intersection points) and for each one, two positions are pointed out in PA. So $O(K)$ space is required for maintaining the two structures. Since the number of polygons defined over the K intersection points is K, $O(K)$ time is required to calculate the areas of polygons. In total, LIP(Q',S') computation is $O(N\log N)$ time and $O(N)$ space complexity, assuming that K and N are of the same order.

3.1.3 Generalized LIP distance

The LIP algorithm described so far works with pairs of trajectories that either satisfy the *LIP criterion* or not. The way that this criterion is utilized as to provide a generic algorithm that makes LIP operator universal was suppressed in the previous discussion. The idea for such an algorithm is to traverse the trajectories applying the LIP criterion and when a segmentation point is identified, compute the area for the already investigated sub-polylines, and recursively continue with the rest of the polylines. To this line, algorithm *GenLIP*, illustrated in Fig. 8, searches for *bad* segments and acts as a driver for the invocation of the LIP distance operator. In other words, the aim of this algorithm is to calculate the summation of LIP between all $pair_j$ resulted by the above segmentation.

Algorithm **GenLIP**(Q, S, p)
Input: Q, S polylines; p integer
Output: The area between polylines Q and S

```
01. result = 0
02. Q'=S'=Ø // Initiate two empty trajectories
03. q=s=1 // q,s are indices to segments of Q,S
04. WHILE q < Q.LAST OR s < S.LAST
05.   IF NOT Bad(Q', S', Qq, Ss) THEN // See Def. 3
06.     Mark Qq,Ss as 'good' & add them to Q',S'
07.   ELSE
08.     FOR k=1 to p // look the next p segments
09.       Mark Qq,Ss as 'good' & add them to Q',S'
10.       // Apply the LIP criterion to Qq + k, Ss + k
11.       IF NOT Bad(Q', S', Qq + k, Ss + k) THEN
12.         GOTO line 20
13.       END IF
14.     NEXT
15.     IF p==0 OR all p segments fail THEN
16.       Remove p segments from Q' & S'
17.       GOTO line 22
18.     END IF
19.   END IF
20.   next(Q', S', q, s)
21. END WHILE
22. result = result + LIP(Qq, Ss, 'bad')
23. result = result + LIP(Q', S', 'good')
24. Q = Q - Q'
25. S = S - S'
26. RETURN result + GenLIP(Q, S, p)
```

Fig. 8 GenLIP algorithm

Definition 8 (generalized LIP distance) The *Generalized LIP* (GenLIP) distance between Q and S, given their segmentation due to the *LIP criterion* into z pairs of sub-trajectories $pair_{\mathrm{j}} = \left(Q'_{\mathrm{j}}, S'_{\mathrm{j}}\right)$, $j = 1, \ldots, z$, is defined as follows:

$$GenLIP(Q, S) = \sum_{j=1}^{z} LIP(Q'_j, S'_j) \tag{10}$$

where $\mathrm{LIP}(Q'_{\mathrm{j}}, S'_{\mathrm{j}})$ is the LIP distance between two polylines Q'_{j} and S'_{j} as implemented by Algorithm LIP.

The procedure for computing $\mathrm{GenLIP}(Q, S)$ distance between two trajectories Q and S is illustrated in Fig. 8. More specifically, the algorithm starts by performing the necessary initializations (lines 1–3). Then, it traverses the segments of Q and S until the last segment of both Q and S is reached (line 4). At each step, it applies the *LIP criterion* (function *Bad*, line 5), and in case the examined segments fulfil the *LIP criterion* conditions, they are marked as *good*, and they are appended to two polylines Q' and S', representing the certified portions of Q and S, respectively. As already mentioned, polylines Q and S are not traversed segment by segment but according to a dynamic approach implemented by function *next* (line 20). In detail, *next* takes as input the certified polylines Q' and S' and the current indices q and s, and appropriately increases q and/or s by one (the possibly incremented values of which are the output of the function), having as criterion whether the following two inequalities hold: $\mathrm{r}_{Q' \cup Q_{q+1}} \geq \mathrm{r}_{S'}$ and $\mathrm{r}_{S' \cup S_{s+1}} \geq \mathrm{r}_{Q'}$. In other words, if extending Q' by $Q_{\mathrm{q+1}}$ (S' by $S_{\mathrm{s+1}}$) results in a polyline with length larger than S' (Q', respectively), then q (s, respectively) is increased (by one) while s (q, respectively) is not. Obviously, the above inequalities may concurrently be true, having as a result the increment of both q and s by one. In case that the examined segments fail to satisfy the LIP criterion (line 7), a chance is given to the next $p \geq 0$ pairs of segments to "correct" this, perhaps local, irregularity by checking whether one of them fulfils the *LIP criterion*. If $p > 0$ and this *look-ahead test* succeeds to find at least one *good* pair of segments, then all pairs up to that point (either *bad* or *good*) are marked as *good* and appended to Q' and S', and the procedure continues in the same way (lines 8–14). Otherwise (i.e., if the look-ahead test fails) the procedure ignores the previous processing by removing p segments from Q' and S' (lines 15–18), invokes the LIP operator for the *bad* segments as well as for Q' and S' (lines 22–23), and recursively calls *GenLIP* for the rest of the initial polylines (lines 24–26).

We should note that the purpose of the parameter p is only to prevent the multitudinous segmentation of the trajectories due to jerky movements that fail the criterion but only temporarily. This would result in huge *LIPgram* lists with small $LIPgram_i$. In this way, we minimize the storage cost while we allow bigger and more meaningful *local similarity indicators* (i.e., $LIPgram_i$), acquainting that the calculated area may be affected. To present the effect of p through an example, let us recall Fig. 6 that illustrates the routes of two trajectories Q and S for which the *GenLIP* procedure marks (S_i, Q_i), $1 \leq i \leq 3$, as pairs of *good* segments and subsequently identifies (S_4, Q_4) as a pair of *bad* segments. If no look-ahead test is performed (i.e., $p = 0$) then the LIP distance function will be initially applied for the *bad* pair (S_4, Q_4), then for the *good* pair $Q' = \{Q_1, Q_2, Q_3\}$ and $S' = \{S_1, S_2, S_3\}$ (i.e., measuring the area of polygon $\{Q_1, Q_2, Q_3, S_3, S_2, S_1, Q_1\}$) and, then, it will

recursively continue for (S_i, Q_i), $i \geq 5$. On the other hand, if a look-ahead test is enabled (e.g. $p = 2$) then the *bad* pair (S_4, Q_4) will be temporarily "ignored" expecting that the irregularity will be "corrected" in one of the subsequent $p = 2$ pairs. Indeed, although (S_5, Q_5) still appears to be a *bad* pair, (S_6, Q_6) turns out to be *good*, and it is $Q' = \{Q_1, Q_2, Q_3, Q_4, Q_5, Q_6\}$ and $S' = \{S_1, S_2, S_3, S_4, S_5, S_6\}$ that will be given as input to LIP Algorithm (Fig. 7).

It is easily implied that GenLIP is a distance operator that may be intuitively utilized by a user (e.g. posing a query of the form "find similar trajectories that passed a few building blocks (say, less than 10 km^2) away from my route"). A powerful feature of the LIP operator is that it does not only provide a global measure for the similarity of two trajectories; furthermore it quantifies the distance among portions of the trajectories. These portions are not statically predefined. For example, the implicit output of the LIP operator is a distance list of the form $LIPgram = \{LIPgram_1, \ldots, LIPgram_{\mathrm{i}}, LIPgram_{\mathrm{i+1}}, \ldots, LIPgram_n\}$, where $LIPgram_{\mathrm{i}} = (I_{\mathrm{k}}, I_{\mathrm{k+1}}, LIP_{\mathrm{i}})$, is a triplet that implies the distance LIP_{i} between points I_{k} and $I_{\mathrm{k+1}}$. It is a trivial task therefore to perform queries upon the resulted *LIPgram* so that to find parts of trajectories that diverge or converge and, as such, to cluster subsets of the above trajectories.

Regarding the time complexity of GenLIP algorithm, it is $O(z \cdot (N \log N + p \cdot \mathrm{N}'))$, where z is the depth of the recursion, $O(N\log N)$ is the cost of LIP function as discussed earlier, p is the number of the look-ahead tests multiplied by N′ which is the maximum number of segments to be examined in order for GenLIP to decide whether the under examination pair of segments is either 'good' or 'bad'. In the one extreme case, $\mathrm{N}' = \mathrm{N}$, but in this case $\mathrm{z} = 1$ and as $p << N$ the $O(p{\cdot}\mathrm{N})$ cost is $O(\mathrm{N})$. Similarly, in the other extreme case, z is order of N (all pairs of segments are marked as '*bad*' – but in this case the cost of LIP function (line 22) is negligible) whereas $p << N$. Overall, we can safely argue that GenLIP Algorithm is $O(N\log N)$ time complexity.

3.2 (Time-aware) spatiotemporal similarity

So far, the GenLIP distance operator does not take temporal information into consideration since it compares the projections of moving objects to the Cartesian plane. However it is designed so as the temporal dimension can be embodied to GenLIP in a uniform way. In this section, we extend GenLIP by proposing a novel distance operator, called *Generalized Spatio Temporal LIP* (GenSTLIP), which measures the spatio-temporal similarity between two trajectories. Intuitively, two moving objects are considered similar in both space and time when they move close at a concurrent mode. This type of similarity is formalized as follows:

Definition 9 (generalized spatio-temporal LIP distance) The GenSTLIP distance between two trajectories Q and S, given a segmentation of their routes according to the *LIP criterion* into z pairs of sub-trajectories $pair_{\mathrm{j}} = \left(Q'_{\mathrm{j}}, S'_{\mathrm{j}}\right)$, $j = 1, \ldots, z$, is defined as follows:

$$GenSTLIP(Q, S) = \sum_{j=1}^{z} STLIP\left(Q'_j, S'_j, k_t, \delta\right) \tag{11}$$

where:

$$STLIP\left(Q'_j, S'_j, k_t, \delta\right) = \begin{cases} \sum\limits_{\forall\, polygon_i \in \Pi} STLIP_i, & if\ \text{pair}(Q'_j, S'_j) = \text{`}good\text{'} \\ LIP_{bad} \cdot (1 + k_t \cdot TLIP_i(\delta)), & if\ \text{pair}(Q'_j, S'_j) = \text{`}bad\text{'} \end{cases} \tag{12}$$

Π is defined as in Definition 5 and $STLIP_i$ for $polygon_i \in \Pi$ is defined as:

$$STLIP_i = LIP_i \cdot (1 + k_t \cdot TLIP_i(\delta)) \tag{13}$$

where $k_\mathrm{t} \geq 0$.

In other words, GenSTLIP is defined to be the summation of all STLIP defined per $pair_\mathrm{j}$, where in its turn a STLIP for a *good* $pair_\mathrm{j}$ is the summation of all $STLIP_i$ defined per $polygon_\mathrm{i}$, that is a multiple of LIP_i (by a factor greater than 1), implying the temporal distance of the corresponding LIP_i. *Temporal LIP* ($TLIP_\mathrm{i}$) is a measure in the range [0, 1] modeling the local temporal distance, and participates to the $STLIP_\mathrm{i}$ measure by a penalty factor k_t which represents user's assigned importance to the time-factor. STLIP is defined similarly in case of a *bad* $pair_\mathrm{j}$.

In order to define the local temporal distance $TLIP_\mathrm{i}$ and associate it with the corresponding LIP_i, we need to find the timepoints when Q' and S' cross each other. Let Qt_I_k be the timepoint when Q' passes from intersection point I_k and $Qt\_I_{\mathrm{k}+1}$ be the timepoint when Q' reaches the next intersection point $I_{\mathrm{k}+1}$. Respectively, St_I_k and $St\_I_{\mathrm{k}+1}$ are the corresponding timepoints for S'. These pairs of timepoints define the periods that each point needs to traverse its route from one intersection to the other. Let $Qp_\mathrm{k} = \left[Qt\_I_\mathrm{k},\ Qt\_I_{\mathrm{k}+1}\right)$ and $Sp_\mathrm{k} = \left[St\_I_\mathrm{k},\ St\_I_{\mathrm{k}+1}\right)$ be these periods. We define the temporal distance $TLIP_i(\delta) \in [0, 1]$as follows:

$$TLIP_i(\delta) = \left\| 1 - 2 \cdot \frac{MDI_i(\delta)}{l_{\mathrm{Q_{k,k+1}}} + l_{\mathrm{S_{k,k+1}}}} \right\| \tag{14}$$

where $l_{\mathrm{Q_{k,k+1}}}$ and $l_{\mathrm{S_{k,k+1}}}$, are the lifespans of the Q'_j andS'_j, respectively, between two intersection points I_k and $I_{\mathrm{k}+1}$, (namely, the duration *dur* of Qp_k, Sp_k), and the *maximum duration intersection* $MDI_i(\delta)$ is defined as:

$$\begin{aligned} MDI_i(\delta) = \max \{ & dur\left(Qp_k \cap Sp_k\right), \\ & dur\left(Qp_k \cap \left[St\_I_k,\ St\_I_{k+1} + \delta\right)\right), \\ & dur\left(Qp_k \cap \left[St\_I_k - \delta,\ St\_I_{k+1}\right)\right) \} \end{aligned} \tag{15}$$

That is, *MDI* is the *maximum duration* of the temporal period (representing the lifespan of a particular section of the motion) of the *intersection* between Qp_k and one of the following three alternatives: a) temporal period Sp_k, b) Sp_k stretched towards future by a temporal interval δ, c) Sp_k stretched to the past by δ. To this end, *MDI* has been incorporated in order to support *almost concurrent* movements. This happens by introducing parameter δ, a time window (tolerance in the past as well as in the future) in which two trajectories, though not moving concurrently, are considered temporally close. Finally, recall that even for trajectories do not really cross each other, there are the two artificial intersection points I_1 and I_n that cover the whole lifespans of the trajectories.

In terms of implementation, the algorithm that computes GenSTLIP is very similar to that of GenLIP, except of two simple modifications.

First, Q and S are restricted at their common lifespan, as it is meaningless to search for spatiotemporal similarity in periods that there is not mutual movement. To exemplify this, the following three lines of pseudocode may be added before line 1 of GenLIP algorithm:

```
...
00.1 p = Sp ∩ Qp // Find period of mutual movement
00.2 Q = Q.at_period(p) // Restricts Q at period p
00.3 S = S.at_period(p) // Restricts S at period p
...
```

Second, the computation of the temporal periods Qp_{k} and Sp_{k} and the respective lifespans $l_{\mathrm{Q_{k,k+1}}}$ and $l_{\mathrm{S_{k,k+1}}}$, of Q' and S' is trivial for a *bad* case (i.e., line 2 of LIP algorithm is changed by direct application of the second branch of (12)), while for a *good* case it is performed concurrently with the computation of the intersection points I_{k} and $I_{\mathrm{k+1}}$, and are also stored at the positioning array PA (line 4). This implies that the function *pos* (e.g. line 7) does not only return the position (i.e., the (x,y) coordinates) of the intersection point PQ(k), but also the corresponding time instance. Finally, *LIPgram* is updated (line 14) accordingly to maintain MDI, $l_{\mathrm{Q_{k,k+1}}}$ and $l_{\mathrm{S_{k,k+1}}}$, which are used at the cumulative step (line 16) as delineated by (11–13).

Considering time complexity, the cost of calculating GenSTLIP is $O(N\log N)$ since GenLIP is the dominant factor in the procedure.

3.3 Discussion about metric properties

In this section, we study the metric properties of the proposed distance functions. As we prove in the following theorems, the GenLIP distance (and its associated functions, GenSTLIP, etc.), fail to be metrics due to the triangular inequality criterion.

Theorem 1 *The distance measure GenLIP on a dataset D of trajectories satisfies the non-negativity, the symmetry, the identity of indiscemibles metric conditions, but not the triangle inequality.*

Proof sketch The *non-negativity* condition (i.e. GenLIP$(x,y) \geq 0$ for all x, y in D) is self-evident that it stands as all measurements are areas of regions. The *symmetry* condition (i.e. GenLIP$(x,y) =$ GenLIP(y,x)) as well as the *identity of indiscemibles* (i.e. GenLIP$(x,y) = 0$ if and only if $x = y$) are trivial to prove. Note that the only two issues that could spoil symmetry is the translation of the segments in the *bad* case, and the weighting of the polygon areas. For the first, we count the maximum area of the two possible translations, while for the second we multiply by a factor (see (5), (9)) that takes into account the sum of the lengths of the involved trajectories. Regarding identity of indiscemibles, $x = y$ implies GenLIP$(x,y) = 0$ as there are only *bad* pairs of identical segments that all of them result in zero areas. Reversely, if GenLIP$(x,y) = 0$ then if it was $x \neq y$ then there would exist at least one segment differentiating the two trajectories. But in this case, this segment would result in a

non-zero area either due to translation or difference on the lengths of the segments in case of a *bad* pair (recall d_T even for the special case that the segment lies on the same line with the translation vector), or due to the formation of a region, the area of which is intuitively the measure of change so as for the trajectory including this segment to be equal to the other trajectory. Regarding the *triangle inequality* condition (i.e., GenLIP $(x, y) \leq$ GenLIP (x, z) + GenLIP (z, y) for all x, y, z in D) we show that it does not hold via an anti-paradigm. Consider the case where x, y, z are single segments, x, y are collinear with same starting point, and the length of y is twice the length of x, and z is parallel to the other two and its length is equal to the length of x, while their perpendicular distance is equal to its length). In this case, GenLIP(x,y) is considered to be a bad case, and is respective to $d_T = \left|r_x - r_y\right| \cdot d = r_x \cdot d$ since the translation vector $\left|\vec{T}_x + \vec{T}_y\right|$ is zero. However, it is obvious that for a small d the GenLIP(y,z) is larger than GenLIP(x,z) + GenLIP(x,y), as $r_y > r_x = r_z$. □

Being non-metric implies that GenLIP is non-indexable by traditional distance-based indexing methods, but it does not mean that GenLIP is not a "good" distance function. Being metric would speed up MST type of queries by pruning the search space with the proper use of the triangle inequality metric property. On the other hand, there are robust distance functions that have been proposed in the literature (e.g. in the domain of image retrieval, such as the Hausdorff distance (Huttenlocher et al. 1993) and the Dynamic Partial Function (DPF) (Goh et al. 2002), but also in the trajectory retrieval field, such as EDR (Chen et al. 2005), LCSS (Vlachos et al. 2002b) and DTW (Berndt and Clifford 1996)) that do not follow triangle inequality. Furthermore, an interesting line of research that is left for future work, is to appropriately adapt the proposed distance functions, so as to obey weaker versions of the triangle inequality metric condition (i.e. *near triangle inequality* Chen et al. 2005), which would still allow us to use distance-based indexing methods.

Regarding the time-aware variant of GenLIP (i.e. GenSTLIP) we need first to study the properties of the local temporal distance TLIP.

Theorem 2 *The distance measure TLIP between two periods is a metric for $\delta = 0$, while GenSTLIP on a dataset D of trajectories satisfies the non-negativity, the symmetry, the identity of indiscemibles metric conditions, but not the triangle inequality.*

Proof sketch It is trivial to prove that TLIP satisfies the first three conditions for $\delta = 0$, while regarding the triangle inequality it is also satisfied as *MDI* is the intersection of two L_1 intervals and TLIP is a monotonous increasing composite function on *MDI*. Similarly with Theorem 1, one can easily prove the first three properties of GenSTLIP. However, althouth TLIP is a metric, GenSTLIP is not a metric as it does not obey the triangle inequality due to that GenLIP also does not hold this condition. □

3.4 A real-world example

In order to provide a better insight of the distance operators presented above, we utilize as a running example the routes Q (green line) and S (red line) of two trajectories from the 'trucks' dataset (discussed in Section 1), illustrated in Fig. 9. The two trucks travel from one depot (point A, lower right in the figure) to the other

Fig. 9 A running example for the calculation of the distance operators

depot (point F, upper left in the figure). For simplicity and ease of exposition of the calculations we include only a few points (see the yellow pins with the white labels) per each route. The usefulness of the points (i.e. pins) having the same color with the routes will be discussed in Section 4.4. Clearly, the routes of the two trucks appear to have three (including the common start and end) intersection points, thus forming two polygons, E_1, E_2 (first E_1, orange polygon and second E_2, yellow polygon). In this case the GenLIP distance is identical with LIP as the *LIP criterion* is always satisfied during traversing Q and S.

Table 1 summarizes all necessary calculations of the different operators that can be applied over the two trajectories illustrated in Fig. 9. Let us assume that both trucks depart at 8:00 A.M. and arrive at their destinations at 8:30 A.M.. Furthermore, truck Q passes from *intermediate* point (i.e., *ipoints*) C at 8:15 A.M., while truck S

Table 1 Calculating LIP–TLIP–STLIP distance functions over real movement data

		E_1	E_2	$\sum$
$r_{Q_{k,k+1}}$		6.54	4.63	11.17 km
$r_{S_{k,k+1}}$		4.33	5.37	9.7 km
LIP_i		2.64	2.07	4.71 km^2
$TLIP_i(\delta)$, $\delta = 0$		0.19	0.14	
$STLIP_i$	$k_t = 0.5$	2.89	2.21	5.1 km^2
	$k_t = 1$	3.14	2.36	5.5 km^2
$TLIP_i(\delta)$, $\delta = 0.05$		0.05	0.14	
$STLIP_i$	$k_t = 1$	2.77	2.36	5.12 km^2

passes from C at 8:10 A.M.. The second and the third row in the table show the distances (in kilometers) between two consecutive intersection points while the rows below show LIP_i, calculated (in square kilometers) according to (2), (3), (4) and (5), $TLIP_i(\delta)$, calculated for $\delta = 0$ or $\delta = 3$ min, according to (14) and (15), and $STLIP_i$, calculated for $k_t = 0.5$ or $k_t = 1$, according to (11) and (13).

As expected, $STLIP \geq LIP$, which is a direct implication of (13), while $STLIP$ increases with k_t. It is also evident that setting δ to a value greater than zero may result to smaller $TLIP_i$, further reducing $STLIP_i$. From a different perspective, the calculated LIP_i (4,71 km^2) can be viewed as a percentage of the maximum LIP that could have been calculated for the two trajectories if they were straight lines starting from a common point and directing to opposite directions (i.e., $\varphi = 180°$, which makes an area of 11,17 $\times$ 9,7 km^2). Under this perspective, the actual LIP_i results in a (normalized in unit space) dissimilarity value of 0.0435 between Q and S.

4 Trajectory similarity search—variations

Expanding our framework, in this section we describe three variations of the previously proposed operators; these variations taking into consideration the rate of change (i.e., speed, acceleration) and directional (i.e., turn) characteristics of the trajectories.

4.1 Speed-pattern based similarity

Two interesting scenarios, which find realistic applications, are the following (we discuss speed-oriented scenarios but one could easily think of their acceleration-oriented counterparts):

- *1st scenario*: The analyst is not interested in time dimension; what she cares about Q and S is their speed v at different segments *seg* in their route. In this case, the problem is to find the similarity between $Q = \{(seg_{Q,1}, v_{Q,1}), \ldots (seg_{Q,n}, v_{Q,n})\}$ and $S = \{(seg_{S,1}, v_{S,1}), \ldots (seg_{S,n}, v_{S,n})\}$.
- *2nd scenario*: The analyst is not interested in space dimension; what she cares about Q and S is their speed v at different time periods *per*. The problem here is to find the similarity between $Q = \{(per_{Q,1}, v_{Q,1}), \ldots (per_{Q,n}, v_{Q,n})\}$ and $S = \{(per_{S,1}, v_{S,1}), \ldots (per_{S,n}, v_{S,n})\}$.

The above scenarios can be thought of as special cases of time-series similarity search, and several well-known methods to perform such tasks can be found in the literature. Another common assumption in those works is that the points are moving with constant speed, which is the case of synthetic motions. On the other hand, we aim to search for similarities of objects moving with fluctuated speed or acceleration and may be randomly sampled, which is the realistic case.

In order to support the first variation, we introduce a new distance operator, called *Genaralized Speed-Pattern STLIP* (GenSPSTLIP), to measure dissimilarity between trajectories with respect to the speed pattern they follow, relaxing either space or time features. To define GenSPSTLIP we follow a similar approach as with GenSTLIP:

Definition 10 (generalized speed-pattern STLIP) Given a segmentation of two trajectories Q and S, due to the *LIP criterion*, into z pairs of sub-trajectories $pair_{\mathrm{j}} = \left(Q'_{\mathrm{j}}, S'_{\mathrm{j}}\right)$, $j = 1, \ldots, z$, the GenSPSTLIP distance between Q and S is defined as follows:

$$GenSPSTLIP(Q, S) = \sum_{j=1}^{z} SPSTLIP\left(Q'_j, S'_j, k_t, k_{SP}, \delta\right) \tag{16}$$

where the relative distance for each sub-trajectory is defined as:

$$SPSTLIP\left(Q'_j, S'_j, k_t, k_{sp}, \delta\right) = \begin{cases} \sum\limits_{\forall\, polygon_i \in \Pi} SPSTLIP_i, & if \text{ pair } \left(Q'_j, S'_j\right) = \text{`}good\text{'} \\ LIP_{bad} \cdot (1 + k_t \cdot TLIP_i(\delta)) \cdot \left(1 + k_{sp} \cdot SPLIP_i\right), & if \text{ pair } \left(Q'_j, S'_j\right) = \text{`}bad\text{'} \end{cases} \tag{17}$$

Π is defined as in Definition 5 and $SPSTLIP_i$ for $polygon_i \in \Pi$ is defined as:

$$SPSTLIP_i = LIP_i \cdot (1 + k_t \cdot TLIP_i) \cdot \left(1 + k_{sp} \cdot SPLIP_i\right) \tag{18}$$

where $k_{\mathrm{sp}} \geq 0$ is a penalty factor playing the same role as k_t does in the definition of $STLIP_i$.

The local Speed-Pattern distance ($SPLIP$) is defined as:

$$SPLIP_i = \frac{\left\| \frac{r_{\mathrm{Q_{k,k+1}}}}{l_{\mathrm{Q_{k,k+1}}}} - \frac{r_{\mathrm{S_{k,k+1}}}}{l_{\mathrm{S_{k,k+1}}}} \right\|}{SP_{\max}} \tag{19}$$

where $r_{\mathrm{Q_{k,k+1}}}$ ($r_{\mathrm{S_{k,k+1}}}$) is the length of the route traversed by Q (S, respectively) between intersection points I_k and I_{k+1}, while $l_{\mathrm{Q_{k,k+1}}}$ ($l_{\mathrm{S_{k,k+1}}}$) is the duration of the lifespan of Q (S, respectively) restricted at their movement from I_{k} to $I_{\mathrm{k+1}}$.

In other words, we adopt the traditional definition of speed as the ratio of the distance traveled over the required travel time. Moreover, in order for $STLIP_{\mathrm{i}}$ to be normalized in [0..1], we divide the absolute difference of local speeds by $SP_{\max}$, which is an application-driven upper bound for speed (e.g. for cars, 50 m/sec or 180 km/h; for boats, 50 mi/h). Also, note that if we omit the $(1 + k_t \cdot TLIP_i)$ factor in (18), the operator becomes time-relaxed as it estimates the distance among trajectories taking into consideration the spatial and speed parameters, irrelevantly to their lifespans.

The algorithm that computes GenSPSTLIP is identical to that for GenSTLIP with the only amendment to take place at the aggregation step (line 16) as delineated by (18–19). Regarding its complexity, it is clear that, like GenSTLIP, GenSPSTLIP presents $O(N\log N)$ time complexity.

4.2 Acceleration-pattern based similarity

In a similar way, we define *Generalized Acceleration-Pattern STLIP* (GenACSTLIP) to measure dissimilarity between trajectories with respect to the acceleration pattern they follow, relaxing either space or time features. Formally:

Definition 11 (generalized acceleration-pattern STLIP) Given a segmentation of two trajectories Q and S, due to the *LIP criterion*, into z pairs of sub-trajectories $pair_{\mathrm{j}} = \left(Q'_{\mathrm{j}}, S'_{\mathrm{j}}\right), j = 1, \ldots, z$, the GenACSTLIP distance between Q and S is defined as follows:

$$GenACSTLIP(Q, S) = \sum_{j=1}^{z} ACSTLIP\left(Q'_j, S'_j, k_t, k_{ac}, \delta\right) \tag{20}$$

where the relative distance for each sub-trajectory is defined as:

$$ACSTLIP\left(Q'_j, S'_j, k_t, k_{ac}, \delta\right) = \begin{cases} \sum\limits_{\forall\, polygon_i \in \Pi} ACSTLIP_i, & if \text{ pair}\left(Q'_j, S'_j\right) = \text{`}good\text{'} \\ LIP_{bad} \cdot (1 + k_t \cdot TLIP_i(\delta)) \cdot (1 + k_{ac} \cdot ACLIP_i), & if \text{ pair}\left(Q'_j, S'_j\right) = \text{`}bad\text{'} \end{cases} \tag{21}$$

Π is defined as in Definition 5 and $ACSTLIP_i$ for $polygon_i \in \Pi$ is defined as:

$$ACSTLIP_i = LIP_i \cdot (1 + k_t \cdot TLIP_i) \cdot (1 + k_{ac} \cdot ACLIP_i) \tag{22}$$

where $k_{ac} \geq 0$ is a penalty factor.

The local Acceleration-Pattern distance ($ACLIP_i$) is given by the following equation:

$$ACLIP_i = \frac{\left\| \frac{\frac{r_{O_{k,k+1}} - r_{O_{k-1,k}}}{t_{O_{k,k+1}} - t_{O_{k-1,k}}}}{l_{O_{k,k+1}}} - \frac{\frac{r_{S_{k,k+1}} - r_{S_{k-1,k}}}{t_{S_{k,k+1}} - t_{S_{k-1,k}}}}{l_{S_{k,k+1}}} \right\|}{2 \cdot AC_{\max}} \tag{23}$$

where the lengths of routes and the durations of lifespans are as described in Section 4.1.

In other words, we adopt the traditional definition of acceleration as the ratio of the change of speed over time. Moreover, in order for $ACLIP_i$ to be normalized in [0..1], we divide the absolute difference of local accelerations by $2 \times AC_{\max}$, which is an application-driven upper bound for acceleration (e.g. for cars, 7 m/s^2 or in other words 0–100 km/h in 4 s).

Again, the algorithm for GenACSTLIP only requires the computation of $ACLIP_i$ of (23) at the aggregation step of GenSTLIP. As all measures are already calculated and stored in *LIPgram*, it is clear that, like GenSTLIP, GenACSTLIP presents $O(N \log N)$ time complexity.

4.3 Directional similarity

A third variation supports similarity of the form: "Find similar trajectories according to their heading". Based on such kind of similarity we could further cluster trajectories that move e.g. initially NW (during period A in place B) and then NE (during period C in place D).

Definition 12 (directional distance) The *Directional Distance* (DDIST) between two trajectories Q and S is defined as follows:

$$DDIST(Q, S) = \sum_{\forall \phi_i} DDIST_{\phi_i} \tag{24}$$

where

$$DDIST_{\phi_i} = DIR\left(Q_{\phi_i}, S_{\phi_i}\right) \cdot w_{\phi_i} \tag{25}$$

is the local directional distance DIR between two equi-length segments Q_{ϕ_i} and S_{ϕ_i}, which are the parts of Q and S, respectively, that $\hat{\varphi}_i \in [0°, 180°]$ is the angle between them; multiplied by a weight w_{ϕ_i} that corresponds to the percentage of Q and S trajectories that participate in the distance. So, given that $r_{Q_{\phi_i}}$ and $r_{S_{\phi_i}}$ ($r_{Q_{\phi_i}} = r_{S_{\phi_i}}$), are the lengths of Q_{ϕ_i} and S_{ϕ_i}, w_{ϕ_i} is defined as:

$$w_{\phi_i} = \frac{r_{Q_{\phi_i}} + r_{S_{\phi_i}}}{r_Q + r_S} \tag{26}$$

Figure 10 depicts that the angle φ_i formed between two segments of Q and S change to φ_{i+1} whenever either Q or S changes direction. This means that a change occurs at the ending points of each segment of Q and S.

The procedure for computing DDIST starts traversing the coordinates' list of both Q and S projected polylines until it examines all the segments of both of them. Two indexes control the access to the points' lists. Only one of the two indexes is advanced at each step depending on which polyline changes its direction. At each step, the angle formed between the segments Q_{ϕ_i} and S_{ϕ_i} starting from the points already reached, and ending either to a point residing at the longer of the two segments and at a distance (from the starting point) equal to the length of the shorter segment (end point of the bold portion of S_{ϕ_1} in Fig. 10), or the end point of the shorter under investigation segments (end point of Q_{ϕ_1}).

The algorithm presented in Fig. 11 simply traverses the polylines and depending on the length of the segments decides which segment to clip. It calculates the local distance and continues with the remaining polylines. Similarly to GenLIP, when one of the two trajectories (i.e. S) comes to its final segment (i.e. its length is smaller), the latter (i.e. $S_{|s|}$) is used as the coupling segment, until the other trajectory also reaches its final segment.

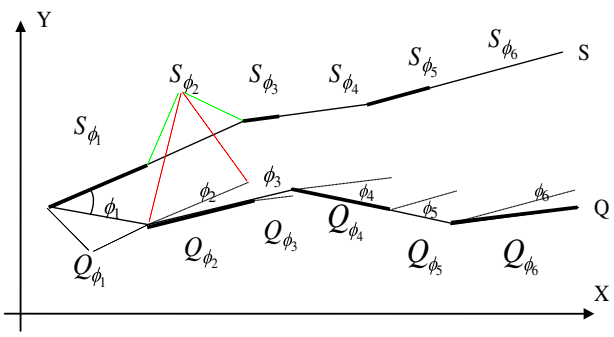

Fig. 10 Directional distance on projected trajectories

Algorithm **DDIST**(*Q*, *S*)
Input: *Q* and *S* are polylines
Output: The Directional Distance between *Q* and *S*

```
01. q=s=1; // q,s are indices to segments of Q,S
02. Qφ = Qq; Sφ = Ss // initialize first segments
03. WHILE q <= Q.LAST OR s <= S.LAST
04.   IF rQφ < rSφ THEN
05.     d=distance(Qφ,s , Qq,e)
06.     Sφi =clip(Sφi , d)
07.     result = result + DIR(Qφi,Sφi)·wφi
08.     q=next(q); Qφi = Qq
09.   ELSE
10.     d=distance(Sφi,s , Ss,e)
11.     Qφi =clip(Qφi , d)
12.     result = result + DIR(Qφi,Sφi)·wφi
13.     s=next(s); Sφi = Ss
14.   END IF
15. NEXT
16. RETURN result
```

Fig. 11 DDIST algorithm

DDIST is time-relaxed in the sense that it does not consider the temporal information of trajectories. In the following, we study the time-aware case of directional similarity.

Definition 13 (temporal directional distance) The *Temporal Directional Distance* (TDDIST) between two trajectories Q and S is defined as follows:

$$TDDIST(Q,S) = \frac{\max\left\{\sum_{\forall Q_i} DDIST_{\phi_i}\left(Q_i, S_{Q_i}\right), \sum_{\forall S_i} DDIST_{\phi_i}\left(S_i, Q_{S_i}\right)\right\}}{\#i} \quad (27)$$

where $DDIST_{\phi_i}\left(Q_i, S_{Q_i}\right)$ is the DDIST between Q_i, i.e., the projection of each 3DLS of Q in the Cartesian plane, and S_{Q_i}, i.e., the corresponding projection of S sub-motion restricted at the period that Q needs to traverse Q_i.

In other words, $DDIST_{\phi_i}\left(Q_i, S_{Q_i}\right)$ is an indicator of how similarly S follows the direction of Q_i. The overall TDDIST is defined as the average DDIST introduced by each pair (Q_i,S_{Qi}). To implement this function we simply restrict the lifespans of Q and S inside the temporal period where there is concurrent development. Subsequently, for each segment Q_i we project the S restricted to the time period of Q_i motion and invoke the DDIST operator for the two sections of Q and S trajectories.

Considering time complexity, in order to calculate directional distance (either DDIST or TDDIST), a single pass over the segments of the two trajectories is required; hence the cost of either DDIST or TDDIST is $O(\max(|Q|,|S|))$.

Similar to Section 3.3, we can easily prove that GenSPSTLIP and GenACSTLIP have the same properties as GenLIP has, regardless of the fact that their building components (i.e. SPLIP and ACLIP, respectively) are metrics. Likewise, DDIST and TDDIST distance functions supporting directional similarity between trajectories, both share the same properties as GenLIP. The key points to prove the properties are: (a) the *local directional* distance *DIR* between two segments, upon which they base, is a symmetrical, increasing monotonous, positive function; (b) the weighting function is symmetrical. Finally, for the triangle inequality consider the antiparadigm of Theorem 1 with segments, x, y having an angle φ with segment y, instead of being parallel. It is straightforward to see that DDIST(y,z) is larger than DDIST(x,z)+DDIST (x,y).

4.4 A real-world example (cont'd)

Continuing the example presented in Section 3.3, we provide all necessary calculations for computing the dissimilarity between the two trajectories on the distance function variations proposed in this section. In particular, the values presented in Table 2 were calculated according to (16–19) for speed dissimilarity, (20–23) for acceleration dissimilarity.

As expected, $SPSTLIP \geq STLIP$ and $ACSTLIP \geq STLIP$ which is a direct implication of (18) and (22), respectively. It is also observed that the differences between *SPSTLIP* and *STLIP* are negligible in this specific example, while this is not the case for the differences between *ACSTLIP* and *STLIP*, which are significant. Of course, these observations are due to the velocity and acceleration dynamics of the two moving objects under examination.

Similarly, in the following table we demonstrate the calculations for directional dissimilarity (i.e. DDIST) according to (24–26). Calculations for the TDDIST variant are omitted as they base on successive invocations of DDIST. Recall that according to the DDIST algorithm there is need to clip segments so as to be equi-length when the DDIST of each pair is calculated. The points where the clipping is performed are depicted in Fig. 9, while the resulting clipped segments are presented in the fourth row of Table 3. As observed, the DIR calculations of each pair of segments present normalized, smooth and intuitive (w.r.t. to their visual illustration in Fig. 9) representation of the segment-by-segment directional distance between the two

Table 2 Calculating SPLIP–SPSTLIP, ACLIP–ACSTLIP distance functions over real movement data

	E_1	E_2	$\sum$
$SPLIP_i$			
($SP_{\max}$ = 120 km/h)	0.01	0.02	
$SPSTLIP_i$			
($k_t = 1, k_{\text{sp}} = 1, \delta = 0$)	3.16	2.40	5.56
$ACLIP_i$			
($AC_{\max}$ = 50 km/h^2)	0.45	0.03	
$ACSTLIP_i$			
($k_t = 1, k_{\text{ac}} = 1, \delta = 0$)	4.56	2.42	6.98

Table 3 Calculating DDIST distance function

Pairs of segments	AB–AC	AC–BC	CD–BC	CD–CE	CE–DF	EF–DF		∑
Angles φ_i	64.26°	36.13°	65.47°	38.52°	20.42°	48.27°		
DIR	0.28	0.10	0.29	0.11	0.03	0.17		
Equi-length pairs	AB–AA′	A′C–BB′	B′C–CC′	C′D–CC″	C″E–DD′	D′F–EE′	E′F–D′F	
Weight w_{φ_i}	0.247	0.168	0.212	0.049	0.249	0.004	0.142	
$DDIST_{\varphi_i}$	0.070	0.016	0.062	0.005	0.008	0.001	0.024	0.186

trajectories, while the corresponding DDIST appropriately adjusts the previous results according to the weight of each pair.

5 A case study over a real movement dataset

In this section we describe a knowledge discovery methodology for analyzing movement data. First we present some architectural aspects that will allow us to figure out the overall proposed methodology. More specifically, Fig. 12 presents the software components utilized in order to perform the appropriate tasks as well as the possible data flows between the components.

In particular, there is a Data Repository consisting of a Moving Object Database (*MOD*) that stores raw trajectory data and a database that stores process related data (spatial objects, such as points / areas of interest, etc.). There is also a *Data Mining Toolkit* consisting of different data mining techniques. In this work we utilize clustering techniques that can be applied on raw trajectory data. These techniques are enriched with the palette of similarity functions proposed in this paper (*Similarity Toolkit*) in order to perform clustering tasks with different semantics. The produced clusters are visualized using the *Visualization Toolkit*. The *Map Repository*

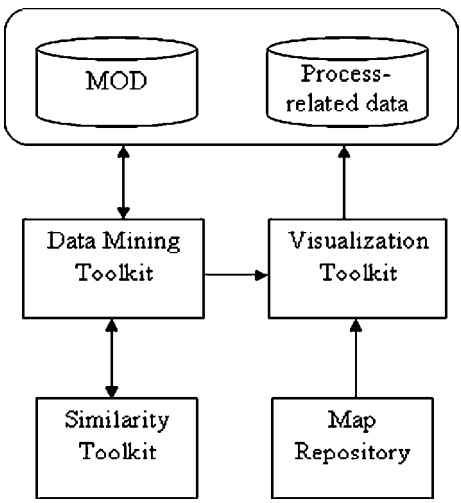

Fig. 12 Architecture of the KD framework for the analysis of movement data

stores maps that are used for visualization purposes while intermediate geographical objects may be stored back to the Data Repository.

5.1 A typical procedure for movement analysis

The proposed procedure consists of seven steps:

Step 1: CS ← Discover common sources in D;
Step 2: CD ← Discover common destinations in D
Step 3: IF CS ≠ ∅ THEN CL ← Link sources CS to destinations CD
ELSE IF CD ≠ ∅ THEN CL ⇓ Link destinations CD to sources CS
Step 4: TZ ← Find traffic zones in CL
Step 5: TR ← Discover typical routes in TZ
Step 6: AR ← Discover alternative routes in TZ
Step 7: TD ← Investigate temporal and dynamic characteristics for TZ and/or TR+AR

Note that this procedure is by no means suggested as a universal and exhaustive method for analysing movement data. The main purpose for designing the procedure has been to define appropriate ways of applying our distance functions in real data analyses. Still, the procedure corresponds to a number of common tasks in movement analysis since it deals with the basic features of trajectories: origin and destination, spatial extent, route, position in time, and temporal dynamics (variation of position, direction, speed, etc.). In particular, the procedure can support the analysis tasks of a city traffic manager introduced in the motivation section (Section 2).

The procedure has been tested on several use cases with the data about the movement of a single individual during a long time period, of courier vehicles, of school buses, and of trucks transporting concrete. It proved to be useful in all these cases, which are characterized by relatively large numbers of repeated trips from/to the same places along the same or similar paths. However, we anticipate that the procedure may not give useful results in cases of less regular character of the movement.

The procedure is based on the use of the proposed distance functions: GenLIP (defined in Section 3.1), GenSTLIP (defined in Section 3.2), GenSPSTLIP (defined in Section 4.1), GenACSTLIP (defined in Section 4.2), DDIST and TDDIST (defined in Section 4.3) as well as three primitive yet useful distance functions: "common starts", which computes the distance in space between the starting points of two trajectories, "common ends", which computes the distance in space between the ending points of two trajectories, and "common starts and ends", which returns the average of the distances between the starting points and between the ending points of two trajectories.

In detail, at Step 1 we aim to detect whether the trajectories originate from a small number of places (common sources). The methodology followed is clustering using the similarity function "common starts" with an appropriate distance threshold (it should be close to the expected size of a place) and finding out whether the sources have their clear-cut "influence areas" with little overlap between them.

At Step 2 we aim to detect whether the trajectories go to a small number of places (common destinations). Again, the methodology followed is clustering using the similarity function "common ends" with an appropriate distance threshold (it should

be close to the expected size of a place) and finding out whether the destinations have their clear-cut "catchment areas" (i.e., areas where all trips are directed toward these destinations) with little overlap between them.

At Step 3 we aim to link sources with destinations. The methodology is as follows (two alternatives): Select the clusters of trajectories defined in Step 1 (Step 2) one by one and apply the analysis described in Step 2 (Step 1) separately to the trajectories of each cluster. This is a case of progressive clustering, i.e., refinement of clusters once obtained through further clustering. In case of detecting typical destinations (sources), compare the sets of destinations (sources) associated with each source (destination).

The procedure continues with the Step 4, where we aim to find out whether the trajectories can be divided into subsets covering different geographic areas (further called "traffic zones"). The methodology is clustering with the use of the distance function GenLIP, which estimates the distances between trajectories according to their geographical positions and extents. Each traffic zone is then analysed in more detail. Thus, Steps 1 and 2 are applied to find out whether there are typical sources and/or destinations of the trips; see also Step 7.

At Step 5 we aim to discover typical routes of movement. The proposed methodology is clustering with the use of the distance function DDIST estimating the distances according to the directions of the movement.

At Step 6 we search for alternative routes between linked pairs of source and destination discovered at Step 3. The proposed methodology is applying DDIST to the trajectory clusters according to the closeness of their starts and ends, according to the idea of progressive clustering.

At Step 7, the temporal variants of the distance functions (STLIP, TDDIST, SPSTLIP, ACSTLIP) are progressively applied to the earlier detected clusters (traffic zones, typical routes, alternative routes) in order to investigate the temporal and dynamic characteristics (temporal intervals of movement, speed distribution, intermediate stops, acceleration patterns, etc.) of the trajectories and find out whether they are nearly uniform within a cluster or significantly vary and whether the variation depends on time (e.g. time of the day or day of the week) or space. Note that Step 7 does not necessarily come at the end of the procedure but may also be performed after Step 4 and Step 5 (in application to the clusters discovered at these steps).

Note that the suggested sequence of steps minimizes the analysis time at the cost of applying simple and computationally inexpensive distance functions to the whole set of trajectories at earlier steps while the following steps apply sophisticated and therefore computationally more demanding functions to earlier obtained subsets.

The steps of the suggested procedure can be matched to the questions of the traffic manager stated in Section 2 in the following way. Question *a* (frequent origins and destinations) can be answered in Steps 1 and 2 and question *b* (connections between origins and destinations) in Step 3. Question *c* (coverage areas of different providers) can be answered in Step 1 and question *d* (traffic zones) in Step 4. Questions *e* and *f* (typical and alternative routes) are answered in Steps 5 and 6. Finally, Step 7 is meant to answer questions *d* (movement characteristics in the traffic zones), *f* (relation of the alternative routes to time), and *e* (speed and acceleration patterns).

Note that in real analyses, it is obvious that some steps may be omitted, depending on the properties of the data and goals of analysis.

5.2 Demonstrating the usage of the proposed distance functions

As a proof-of-concept for the procedure described above, in the following we demonstrate its application in a real world case study using the 'trucks' dataset illustrated in Fig. 1 and trying to find answers to the questions of the traffic manager. We realized the architecture depicted in Fig. 12 by implementing the distance functions as SQL operators into Hermes MOD engine (Pelekis and Theodoridis 2006; Pelekis et al. 2006, 2008, 2011). For clustering trajectories we used the OPTICS (Ankerst et al. 1999) algorithm, while for the visualization we appropriately extended the functionality of the Visual Analytics Toolkit (Andrienko et al. 2007).

To find the origins and destinations of the trips (question *a*), the analyst groups the trajectories by spatial closeness of their starts and ends using the distance function "common starts and ends" (defined in Section 5). The results of the clustering show the analyst that a great majority of the trips are round trips from two places; these trips have been grouped into two clusters with 429 and 232 trajectories, respectively (see Fig. 2). Moreover, most of the trajectories in the remaining clusters either start or end in one of these places. This means that these two places play a special role as truck stations or depots. They will be henceforth referred to as depot 1 and depot 2. It is not likely that depot 1 and depot 2 belong to different companies as there are trips between them. These findings provide also an answer to question *b* about the connections between the sources and destinations.

Next, the analyst is interested how the trips originating from depot 1 and depot 2 are distributed over the territory and whether each depot has its "area of influence" where it dominates (question *c*). The analyst applies clustering according to the positions of the trip starts (by choosing the distance function "common starts"). In the result, the tool produces a cluster of 308 trips originating from depot 1, a cluster of 558 trips originating from depot 2, and a number of much smaller clusters. On a map display, the analyst sees that depot 1 mainly serves the northern and western parts of the territory while depot 2 mainly serves the southern and eastern parts, but the "areas of influence" overlap.

Now, the analyst wants to investigate the traffic zones and the routes of the trucks from each of the depots (questions *d* and *e*). She would like to detect frequently occurring routes but also see how many trips were unique. The analysis is done separately for each depot. Figure 13 presents the results of the application of GenLIP method (with the distance threshold set to 6800) to the subset of the trajectories originating from depot 2. The figure contains screenshots with all clusters obtained (top) and schematic representations of the major clusters, which consist of 38 (red), 37 (blue), 35 (green), and 12 (pink) trajectories. The widths of the arrows indicate the frequencies of the occurrences of the track segments. The results show the major traffic zones.

Similarly, Fig. 14 presents the results of the application of DDIST method (with the distance threshold set to 0.15) to the same subset of the trajectories for discovering the typical routes. For DDIST, the largest clusters, which are represented in a schematic form at the bottom of Fig. 14, include 62 (violet), 57 (orange), 26 (pink), 16 (light blue), 13 (red), and 12 (brown) trajectories.

Figures 13 and 14 illustrate the difference between the approaches: GenLIP groups the trajectories according to the areas where the major movements occurred whereas DDIST groups the trajectories according to the major directions of the

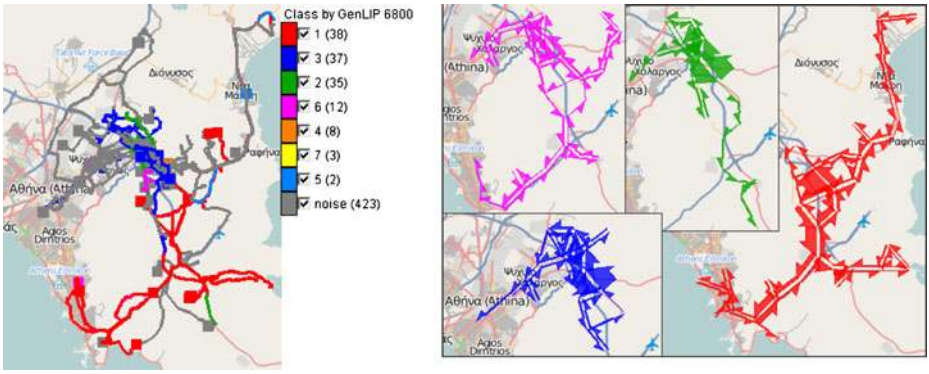

Fig. 13 Clusters produced with the use of GenLIP

movement. Hence, an analyst can use GenLIP in order to see the different areas attended by the trucks and use DDIST to get an idea about the main directions of the trips.

In order to investigate the temporal and dynamic characteristics of the truck movement in different traffic zones and/or on different routes, the analyst applies the versions of the distance functions taking into account the temporal characteristics of the trajectories. In particular, the functions GenSTLIP and TDDIST are used to refine the results of clustering by GenLIP and DDIST, respectively. Thus, the attempts to refine the clusters presented in Fig. 13 through further clustering with the use of GenSTLIP result in finding in each cluster a subset of trajectories with close temporal characteristics. The sizes of the subsets vary depending on the distance threshold specified for the clustering tool. For example, Fig. 15 portrays a subset of trajectories from cluster 2. The trajectories are shown in a 3-dimensional view, known as space-time cube, where the vertical dimension represents the time. The temporal axis is directed upwards. The time references in the trajectories have been shifted to synchronize the starting moments of the trajectories.

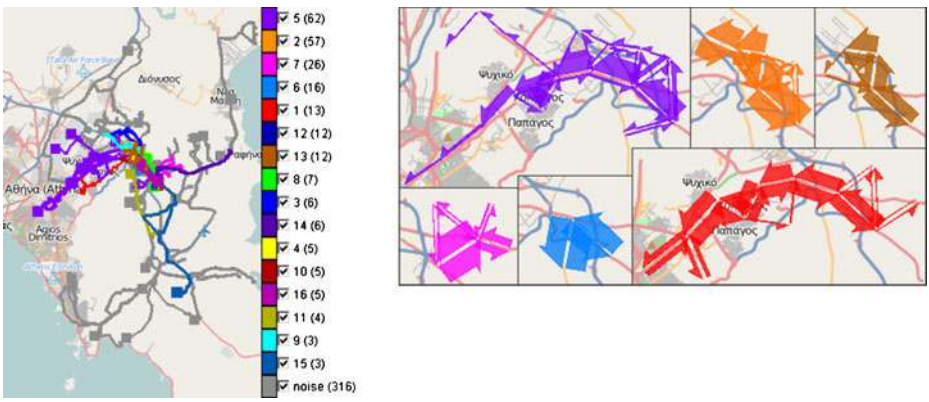

Fig. 14 Clusters produced with the use of DDIST

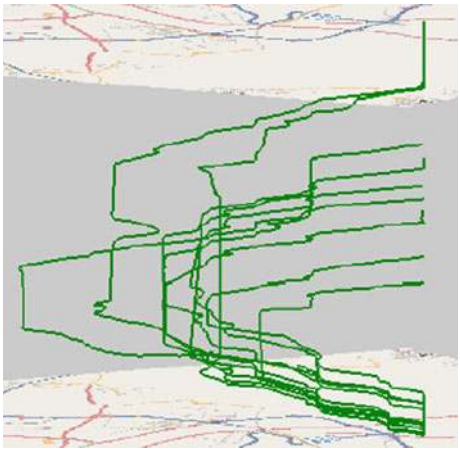

Fig. 15 A cluster of similar trajectories according to GenSTLIP

For selected clusters produced with DDIST, further clustering with the use of TDDIST gives in some cases more than one sub-cluster. For example, Fig. 16 presents the outcomes of sub-clustering of the largest cluster (violet) shown in Fig. 14. Four sub-clusters have been detected; their trajectories are coloured in blue, yellow, red, and green, respectively.

Clearly, the blue lines are longer than the others; apparently, the dynamic characteristics of the corresponding trajectories differ from the rest mainly because of the difference in the routes and trip destinations. However, one of the green lines has the same length as the blue lines, which means that not only the length played a role. The shape of the green line significantly deviates from those of the blue lines in the lower part of the space-time cube, which means different dynamics in the initial period of the movement. Note that a gentle slope of a line indicates movement with a high speed while a steep slope means slow movement; vertical line segments correspond to absence of movement. Thus, the shape of the green line indicates a delay at the beginning (vertical line), which was followed by movement with a higher speed than in the trajectories represented by the blue lines. By the speed pattern after the initial delay this trajectory is similar to the other trajectories of the "green" group. The trajectories represented by the red lines are characterized by

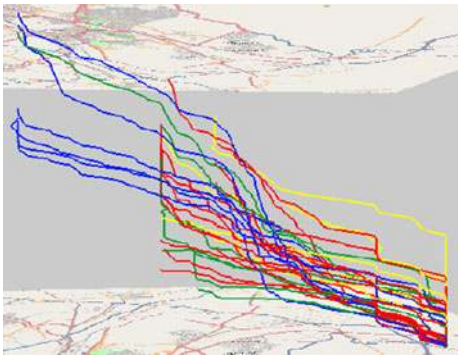

Fig. 16 Four sub-clusters of the violet cluster shown in Fig. 14 have been obtained by means of clustering with the use of TDDIST

high speeds at the beginning of the movement followed by slowing down. This is similar to the blue lines, but there the speeds in the middle of the trips are yet lower than in the "red" cluster. The yellow lines represent the trips where the speeds were high all the time except for some initial delays in the place of the trip start. These observations are actually confirmed by clustering the trajectories of this subset by using the GenSPSTLIP operator.

The detailed example we provided demonstrates that (1) the distance functions we have defined and implemented are effective for analysing real data, and (2) several different distance functions are required for a comprehensive analysis. The reason is that trajectories are complex spatio-temporal objects with heterogeneous properties, which cannot be handled uniformly. Using different distance functions for different properties of trajectories allows the analyst to gradually build a comprehensive understanding of all relevant relationships between them.

5.3 A deeper look at local similarities

An interesting property of *LIPgrams* is that they allow us to define and study local similarities. This is very useful in real world cases in which discovering the similarity among full trajectories could be meaningless. However, discovering the similarity in particular spatial regions and/or temporal periods in an unsupervised way could be really important, as it identifies a similarity pattern between two trajectories. Of course, as the number of *LIPgrams* increases, their storage cost becomes higher, while some of them may carry little information, whereas some other may exhibit a repetitive pattern. As already discussed, in order to tackle this issue we have introduced the p parameter. In addition, we may aggregate *LIPgrams* having as goal not to spoil the distance pattern introduced. For doing so we may apply any well known binning method, which can be easily incorporated in the algorithms of the operators. For instance, we can merge $LIPgram_i$ with $LIPgram_{i+1}$ to produce a new one $LIPgram_{\text{new}} = (ip_{\text{i}}, ip_{\text{i+2}}, AVG(LIP_{\text{i}}, LIP_{\text{i+1}})$, by concatenating the areas and setting the new distance to be the average distance of their parents. This merging may occur if the combined locality does not correspond over a certain fraction f of the lengths of Q and S, and the absolute difference of the individual distances is less than a matching threshold e (i.e., $|LIP_{\text{i}} - LIP_{\text{i+1}}| < e$).

We demonstrate the usage of the above discussion using the 'trucks' dataset. Instead of trying to compare trajectories in their full extent we try to study local similarities applying a similar strategy as the one described earlier. We apply the *GenSPSTLIP* operator by binning on local spatial regions of a regular grid produced over the metropolitan area of Athens. Figure 17 presents the produced grid, where each cell is visualized as a 3-dimensional cube, whose height corresponds to the number of pairs of trajectories that exhibit similar successive *GenSPSTLIPgrams* during the movement of the trajectories inside the cell. The height is normalized in the range from 1 km to 10 km with respect to sea level, so as for the upper side of each cell to be visible. In other words, the height of each cell is analogous to the number of pairs that present a stable speed pattern inside the cell, irrespectively of the absolute value of the *GenSPSTLIPgrams*.

Obviously, low height cells mean either low frequency of concurrent movements in the corresponding area, or high diversity in the speed patterns. Such diversity could be useful for an analyst as an indication that either the road network or the

Fig. 17 Exploiting LIPgrams

street lights allow such a driving behavior. Examination of sudden changes between adjacent cells may further be an issue that deserves further investigation by a traffic engineers.

6 Performance study

In this section, we present a performance evaluation of our proposal. The goal is twofold; on the one hand we evaluate the scalability of the proposed algorithms, while on the other hand, we assess the quality of the distance operators taking advantage of ground truth provided by synthetic datasets.

6.1 Experimental settings

We have used a synthetic movement dataset generated by Brinkhoff's Generator (Brinkhoff 2002). In particular, we used Oldenburg road network (Brinkhoff 2011) in order to generate the dataset. The network consists of 6,106 nodes and 7,036 edges (density $D = 0.00038$). In Table 4 we list the details of the dataset used in our experiments; a screenshot of the dataset appears in Fig. 18a.

We further produced five variants of the above synthetic dataset, where each new dataset is a compressed version of the original. More specifically, the trajectories of the synthetic dataset were compressed using the TD–TR state-of-the-art trajectory

Table 4 Synthetic dataset generated on Oldenburg road network

# of Trajectories	100
# of 3DLS	163 … 166
Trajectory length (m)	2516 ... 41312 (avg. 7427)
Trajectory speed (m/s)	1.0 ... 16.9 (avg. 3.1)

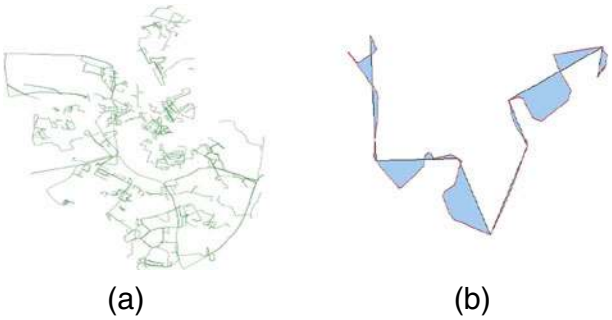

Fig. 18 **a** Synthetic dataset S_0 generated on Oldenburg's road network, **b** the route of a trajectory (*black polyline*) and its compressed counterpart (*red polyline*)

compression algorithm (Meratnia and de By 2004). The TD–TR algorithm is based on the well-known Douglas-Peckeur line simplification algorithm (Douglas and Peucker 1973), which follows a divide-and-conquer approach to keep only the most important points of a polyline, i.e., the ones that lie far from the line that would result if these points were removed. TD–TR extends this algorithm by further taking the parameter of time into account. In particular, it replaces the Euclidean distance used in Douglas-Peckeur by a time-aware one, called Synchronous Euclidean Distance (SED). We applied the TD–TR compression technique with parameter values of p_{TD-TR} in the set $\{0.05\%, 0.1\%, 0.2\%, 1\%, 5\%\}$ of the length of each trajectory. As such, five datasets of 100 trajectories each were constructed (denoted as $S_{0.05}$, $S_{0.1}$, $S_{0.2}$, S_1, S_5, while by S_0 we denote the original dataset). We should note that even for small TD–TR parameter values the effect of the compression is significant. For instance, for $p_{TD-TR} = 0.1\%$ the compressed trajectory has less than half 3DLS in comparison with the original trajectory, while for $p_{TD-TR} = 5\%$ the corresponding dataset has almost 5% of the initial number of 3DLS. In other words, increasing p_{TD-TR} parameter produces compressed trajectories with much fewer sampled points, while the general sketch of the trajectory remains unaffected (see Fig. 18b). The purpose for constructing these compressed versions of the synthetic dataset is that it allows us to consider as a cluster each trajectory from S_0 together with all of its compressed counterparts. The nice property of such clusters, which to the best of our knowledge is not available in real datasets, is that they contain trajectories with exactly the same lifespans, while concurrently having similar routes (the higher the compression degree, the lower the route similaity) and directional patterns, while at the same time appearing different sampling rates and lengths.

Note that compressing trajectories produces clusters with the above mentioned properties, but also has the effect of shortening the length of the routes as compression vanishes all the topical detailed movements. To experiment also with trajectories of roughly the same length but with different directional patterns we have produced five more datasets by adding different percentages of noise at every point of the S_5 dataset, after having normalized it (we denote this dataset as N_0). Normalization is recommended so that the distance between two trajectories is invariant to spatial scaling (Chen et al. 2005). We chose S_5 since this is the dataset that includes trajectories having the most abstract sketch. The noise was added using $x_{noise} = x + randn * rangeValues$ and $y_{noise} = y + randn * rangeValues$ formulas, where *randn* produces a random number chosen from a normal distribution with mean 0 and

variance (0.001, 0.002, 0.05, 0.1, 0.2), and *rangeValues* is the range of values on X and Y axis, respectively. Obviously, the idea of these datasets (denoted as $N_{0.001}$, $N_{0.002}$, $N_{0.05}$, $N_{0.1}$, $N_{0.2}$) is to produce 'shaken' variants of the initial dataset.

The experiments were run on a PC with Intel Pentium at 3.4 GHz, 2 GB RAM and 100 GB hard disk. All datasets can be downloaded from: http://infolab.cs.unipi.gr/pubs/jiis2011/.

6.2 Experimenting with GenLIP processing time

Our experimental study starts with results on the cost of computing the GenLIP distance function as it is the one that prescribes the computational cost of the proposed operators. We utilize the S_0 dataset and for each route, we compute the GenLIP distance with each route of the other (hence we measure the total time to perform 100 $\times$ 100 calculations). The experiment is performed five times, for different values of $p = 0, 2, 4, 6, 8$; recall that p is a look-ahead parameter of GenLip algorithm (see Fig. 8). This set of experiments is repeated five times, using the $S_{0.05}$, $S_{0.1}$, $S_{0.2}$, S_1, S_5 datasets, respectively. Figure 19 illustrates the average processing time between a pair of routes.

As can be observed from the experimental results, the running times follow a curve that proves the complexity analysis of the GenLIP distance function. Obviously, increasing the $p_{\text{TD-TR}}$ parameter (i.e., increasing the sampling rate) we get significantly lower processing time and this is rational as the number of 3DLS is decreasing. We also notice that enlarging the values of p has no effect on the performance of the GenLIP operator that remains stable (i.e., differs only a few seconds for 10,000 experiments). This implies that checking the LIP criterion comes almost with no cost.

6.3 Experimenting with GenLIP quality

Turning to the quality of GenLIP, we experiment on its effectiveness when used for classification tasks. We classified routes according to their GenLIP distance following a methodology initially introduced by Keogh and Kasetty (2002). In this technique, each route is assigned a class label. Then the *leave-one-out* prediction mechanism is applied to each route in turn. That is, the class label of the chosen route is predicted to be the class label of its nearest neighbor, based on the given distance. If the prediction

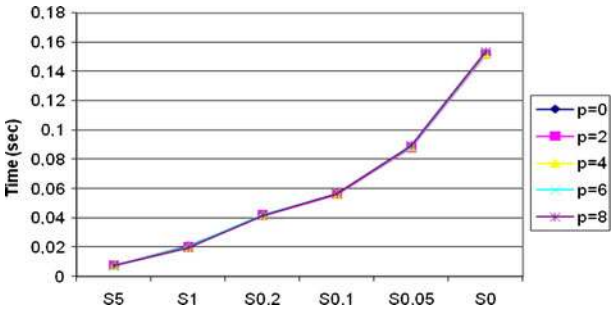

Fig. 19 Average processing times for GenLIP function

is correct, then it is a *hit*; otherwise, it is a *miss*. The *classification success* is defined as the ratio of the number of hits to the total number of routes. We applied the *leave-one-out* prediction mechanism based on the GenLIP distance operator for each one of the 100 routes of S_0 dataset, against each of the five compressed variants (i.e., $S_{0.05}$, $S_{0.1}$, $S_{0.2}$, S_1, S_5), succesively. These five experiments are performed five times each, for different values of $p = 0, 2, 4, 6, 8$. Note that in each experiment, every trajectory of S_0 is assigned a different class label, and we count a hit when compared to all trajectories of the compressed variant (e.g. $S_{0.05}$) the nearest trajectory is the counterpart compressed. So, by increasing the p_{TD-TR} parameter we harden the classification problem. Obviously, this one-hundred-class classification problem is a hard problem, and the increase of the compression rationally results in few misses when strictly counting hits only if the 1-NN has the same label. Relaxing this, we may consider as a hit if the compressed counterpart is one of the k-NN. Applying this procedure for $k = 1, 2, \ldots, 10$ and $p = 0$, we get the results illustrated in Fig. 20.

The main conclusion that can be drawn from this chart is that the GenLIP distance operator turns out to be robust since it presents up to 90% accuracy for low compression rates. The accuracy lowers as the compression increases (which is a straightforward behavior) but even for very high compression it remains at high levels with the reported accuracy being even absolutely correct for small values of k. The same conclusions are present in the charts when varying the number of p (they are omitted as they are almost identical to the above one). This is because parameter p is a simple way to produce bigger and more intuitive *LIPgrams*, or in other words, a mean to overlook jerky movements (or noise).

6.4 Experiments on spatiotemporal similarity

Although starting from different baselines, it is only GenSTLIP among the proposed operators that can be compared with related work. The main conclusion from the study of Ding et al. (2008) is that it is not straightforward to come up with an

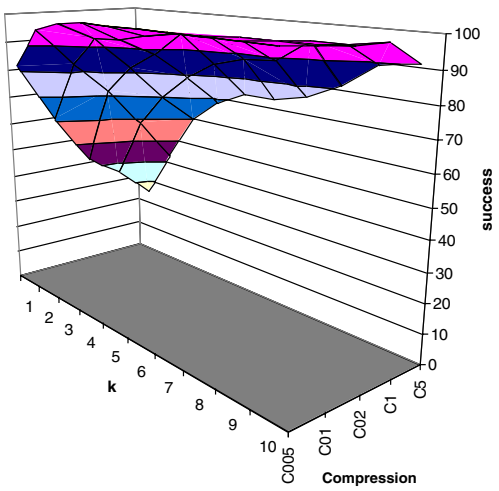

Fig. 20 GenLIP leave-one-out classification success against compression $p_{\mathbf{TD-TR}}$ and k

absolute winner similarity measure, while the results vary among different datasets and depend on the experimental setting. For instance, in (Chen et al. 2005) the authors' experiments show that EDR clearly outperform LCSS (Vlachos et al. 2002b) and DTW (Berndt and Clifford 1996). However, in Ding et al. (2008) the authors show that EDR is only slightly better than DTW. We would like to note that the purpose of this section is not to exhaustively compare GenSTLIP with all other proposals. As exemplified in Ding et al. (2008) this would require a comprehensive setting with many datasets and by carefully designing experiments in a way that they will reveal the advantages as well as the disadvantages of each similarity measure against various types of mobility data (e.g. animals, cars, ships, cyclists, walkers etc.). We consider that the main contribution of the current approach relies on the integration of various types of novel distance operators under a flexible framework, and a systematic comparison of all proposals against mobility data, although very interesting is beyond the scope of the current work. As such, here we demonstrate the applicability of GenSTLIP by comparing it with EDR (Chen et al. 2005), which seems to have even slight better behavior than others, and we leave as future work the design of a more analytical benchmark.

More specifically, following (Chen et al. 2005) and in order to get the best clustering results we normalized the dataset. For a fair benchmark, we also improved EDR by interleaving samples in the compressed trajectory with linear interpolation at the timestamps the checked dataset trajectory was sampled. Although not important in the experiment setting, we run GenSTLIP giving penalty $k_t = 1$, while we set the temporal interval δ to 10 s. Note that, as parameter δ is similar to the parameter ε of EDR and LCSS, a general guideline for setting parameter δ is to be a quarter of the maximum standard deviation of trajectories for which the best clustering results are reported (Chen et al. 2005).

We conducted experiments using the synthetic dataset S_0 consisting of 100 trajectories, which was randomly split to 10 tens and, for each ten, the experiment introduced in (Vlachos et al. 2002b) and (Chen et al. 2005) was performed. More specifically, for each ten we formed 10 datasets of 10 clusters each, one for each trajectory, where one dataset is different from the other only in the number of trajectories per cluster. For example, clusters of the first dataset (i.e., $p_{\mathrm{TD-TR}} = 0.05\%$) consist of the original trajectory and the corresponding compressed with the lowest $p_{\mathrm{TD-TR}}$ value (i.e., $p_{\mathrm{TD-TR}} = 0.05\%$). Clusters of the second dataset consist of the original trajectory and the two corresponding compressed with the two lowest $p_{\mathrm{TD-TR}}$ values, and so on. For each dataset, we got all possible pairs of clusters (i.e., 45 cluster pairs) and we partitioned them into two clusters applying an agglomerative hierarchical clustering algorithm with the complete linkage criterion found in (CLUTO 2011). We sketched the dendrogram of each clustered result to evaluate whether it correctly partitions the trajectories.

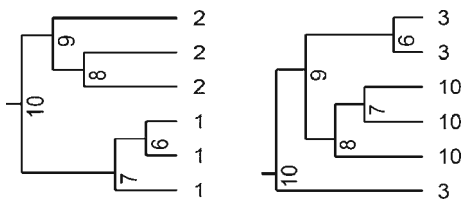

Fig. 21 Examples of correct (*left*) and erroneous (*right*) clusterings

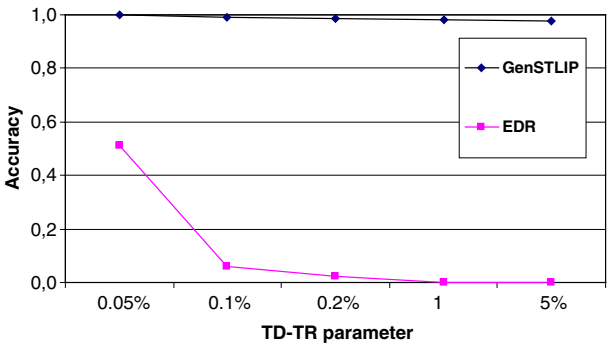

Fig. 22 Accuracy against clusters with more dissimilar trajectories

Figure 21 depicts a correct (left) and an erroneous (right) clustering for the second dataset. A dendrogram is considered erroneous if there is even one node in the hierarchy, except the root, that joins items belonging to different classes (i.e., the right dendrogram is characterized as erroneous due to node 9 that joins nodes from different classes).

The average (from the 10 tens, i.e. from $45 \times 10 = 450$ experiments) results of the experiments are illustrated in Fig. 22. Obviously, STLIP outperforms EDR. Actually, STLIP correctly partitions the dataset into two clusters even we add trajectories that were compressed with high values of $p_{\mathrm{TD-TR}}$ of their length. A straightforward conclusion that can be made by observing the dendrograms produced by EDR is that they are able to correctly identify the NN of the query trajectory (i.e., the case of node 6 in the right dendrogram of Fig. 21); or even to temporarily/initially identify the correct cluster at the lower levels of the dendrogram (i.e., the case of node 8 in the right dendrogram of Fig. 21); however, at the end it fails in detecting similar trajectories of almost the same length which have been sampled differently. This is because it attempts to match the sampled positions one by one, which is not the usual case in real world case studies.

Normalization does not affect the outcome of this experimentation. We repeated the experiment with dataset N_0 and the results turned out to be the same (therefore, they are omitted).

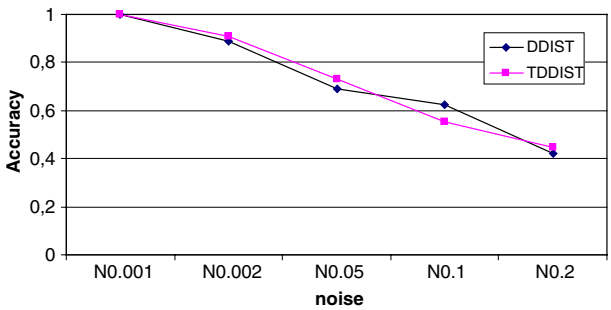

Fig. 23 Demonstrating TDDIST and DDIST

6.5 Experiments on directional similarity

The purpose of this section is to evaluate the operators focusing on the directional similarity between trajectories, namely the TDDIST and the time-relaxed version DDIST. We perform the same experiment as in the case of the previous subsection. Intuitively, compressed versions of a trajectory follow similar direction patterns, and, as such, they form a cluster that, both TDDIST and DDIST will be able to identify against a corresponding set of compressed versions of another trajectory.

The results of the experiments having as base the S_0 dataset are 100% success for $p_{\mathrm{TD-TR}} = 0.05\%$ and 0% for all other compression values. The reason of this result is clearly due to the fact that compression significantly reduces the length of the trajectories, while DDIST and TDDIST are scale variant. The results having as base the N_0 dataset, depicted in Fig. 23, confirm our intuition, also illustrating the efficiency of the operators. As trajectories in the N_i datasets share the same lifespans, it is rational that DDIST and TDDIST present more or less the same behaviour.

7 Related work

Most of related work in trajectory similarity search is inspired by the time series analysis domain, based on mapping a trajectory into a vector in a feature space and using an L_p-norm distance function. The advantage of this approach is that it allows the similarity between the trajectory vectors in the time domain to be approximately equal to the similarity between the two points in the feature domain. The first proposal following this paradigm was by Agrawal et al. (1993) who adopt the Discrete Fourier Transformation (DFT) to be the feature extraction technique since DFT preserves the Euclidean distance and, furthermore, only the first few frequencies are important. Rafiei and Mendelzon (2002) use Fourier descriptors to represent shapes boundaries, compute a fingerprint for each shape, and build a multidimensional index (R-tree) on fingerprints. The distance between two shapes is approximated by the distance of the corresponding fingerprints; this distance is not affected by variations in location, size, rotation and starting point. Although robust in the timeseries domain, these approaches do not take into account the distinctive properties of MOD.

Although Euclidean measures are easy to compute, they do not allow for different baselines or different scales. The main drawback of these methods is that their performance degrades in the presence of noise and outliers since all elements should be matched. To address the disadvantages of the L_{p}-norm, Goldin and Kanellakis (1995) use normalization transformations and compute the similarity between normalized sequences. Although this method solves some problems like the different baselines, it is still sensitive to phase shifts in time and does not allow for acceleration along the time axis. (Lee et al. 2000) compute the distance between two multidimensional sequences by finding the distance between minimum bounding rectangles. Although this speeds up calculations, it leads to coarse distance approximation. Two approaches, which also use Euclidean distances, include the lower bound techniques (Cai and Ng 2004) and the shape-based similarity query (Yanagisawa et al. 2003). Both approaches can be applied only on trajectories with same lengths. In the literature there are proposals for generalizing a dis-

tance function to non-uniform sized points (Tiakas et al. 2009). The lack of such a methodology makes these approaches not directly applicable to a real-world MOD.

Another approach is based on the Dynamic Time Warping (DTW) technique that allows stretching in time in order to get a better distance (Berndt and Clifford 1996). DTW has been adopted in order to measure distances between two trajectories that have been represented as path and speed curves (Little and Gu 2001) or as sequences of angle and arc-length pairs (Vlachos et al. 2002a). Sakurai et al. (2005) present the fast search method for Dynamic Time Warping, utilizing a new lower bounding distance measure that approximates the time warping distance. (Fu et al. 2008) combine DTW and uniform scaling to achieve meaningful results in domains where natural variability of human actions must be taken into account. Lin and Su (2005) introduce the "One Way Distance" (OWD) function, which is shown to outperform DTW regarding precision and performance. In general, DTW suffers from shortcomings such as sensitivity to different baselines and scales of the sequences that can be reduced by first normalizing the sequences.

Another approach uses the Longest Common Sub Sequence (LCSS) similarity measure (Bollobas et al. 2001). The basic idea is to match some sequences by allowing some elements to be unmatched. The advantage of the LCSS method is that it allows outliers, different scaling functions, and variable sampling rates. Vlachos et al. (2002b) use the LCSS model to define similarity measures for trajectories. The LCSS model is extended by considering a set of translations and finding the translation that yields the optimal solution to the LCSS problem.

In order to overcome the inefficiency of the previously described methods in the presence of noise or obstacles, Chen et al. (2005) proposed a new distance function called Edit Distance on Real sequences (EDR), based on the Edit Distance (ED) widely used in bio-informatics and speech recognition to measure the similarity between two strings. Their experimental evaluation shows that the proposed distance function is more robust than Euclidean distance, DTW and ERP (Chen and Ng 2004) and more accurate than LCSS, especially when dealing with trajectories having Gaussian noise.

As already mentioned, the previous works focus on the movement shape of the trajectories, considering them as 2D or 3D time series data. In other words, only the sequence of the sampled positions is taken into account, while the temporal dimension is ignored. Obviously, leaving the absolute temporal information out of the definition of the distance functions, omits crucial semantics regarding the temporal distribution of relationships between trajectories. This is a key distinctive characteristic of our approach w.r.t. above cited works.

In (Frentzos et al. 2007) the authors proposed a similarity metric (and an approximation method to reduce its calculation cost) in order to support similarity search by utilizing R-tree-based trajectory indexing structures. Furthermore, assuming constant speed during individual segments of trajectories, Trajcevski et al. (2007) proposed an optimal and an approximate matching algorithm between trajectories, both under translations and rotations, where the approximate algorithm guarantees a bounded error with respect to the optimal one. The idea behind the approach is to balance the lack of temporal-awareness of the Hausdorff distance with the generality of the Frechet distance. These works in comparison to our approach propose only a

spatiotemporal distance function, while we focus on all interesting features of the trajectories (i.e. locality, temporality, directionality, rate of change) and propose different similarity functions to especially handle each property. This allows us to formulate a flexible framework for the comparison of trajectories based on the above properties, which may be used in diverse application scenarios, enhanced by visual analytics functionality. Moreover, the design of the proposed distance functions, allow us to compare trajectories locally and as such to expose their local relationships. This is in contrast to all previous approaches that only define global functions for the whole lifespan of the trajectories.

Another work that follows a similar tactic with our approach, as it takes into account inherent motion parameters, is the work of Lee et al. (2007) that defines a distance function on directed segments, based on their perpendicular, parallel and angle distance, which is used in a variant of the DBSCAN clustering algorithm aiming at the discovery of sub-trajectories clusters. The shortcomings of this work are that temporal information is not taken into account, while this approach is only applicable to simple linear sub-trajectories (i.e. segments) and not to whole trajectories. As such, our work may be considered as a generalization of this work to whole trajectories that further utilizes the temporal dynamics of complex movement data.

Recently, Ding et al. (2008) provided a comprehensive validation of most of the above discussed similarity measures by conducting an extensive set of experiments on 38 time series data from various application domains. The results of the experiments of this insightful work validated some of the existing achievements, while in some cases the study suggests that certain claims in the literature may be unduly optimistic. The datasets used in this study are not movement data (i.e. GPS recordings) and omit the time stamps at which the recordings occur, assuming that the sampling rates of the time series are the same. Furthermore, in this way absolute time semantics are left out of the evaluation benchmark. We believe that the reproduction of the same kind of experiments on many different real mobility datasets is an extremely interesting work that would provide insightful knowledge to the mobility data mining community (Giannotti and Pedreschi 2008; Giannotti et al. 2007; Nanni and Pedreschi 2006; Pelekis et al. 2009).

In the visual analytics domain, the most common approach to dealing with large collections of movement data is aggregation. A survey of the aggregation methods used for movement data is done in (Andrienko and Andrienko 2010). Another approach is based on filtering, i.e. visualization is applied to a data subset selected according to a user-specified query. Researchers pursuing this approach focus mainly on advancing query and search techniques (Kwan and Lee 2004; Yu 2006). Laube et al. (2005) combine visualization with data mining techniques searching for specific kinds of patterns in movement data. Combination of visualization and clustering is also a frequently used approach in visual analytics; however, clustering is typically applied to relational data (i.e. tables) rather than spatio-temporal data. The visual analytics framework for movement data proposed in (Andrienko et al. 2007) combines visualization with database processing and clustering techniques. The framework essentially relies upon the use of clustering with diverse similarity measures, which is consistent with the work reported in this paper. More sophisticated distance functions are introduced in this paper, which can be used for more comprehensive analysis of movement data, as illustrated in Section 6.

8 Conclusions and future work

In this paper, we proposed novel distance operators, to address different versions of the so-called trajectory similarity search problem that may straightforwardly support knowledge discovery in MOD. In particular, we proposed measures for *spatiotemporal* and (temporally-relaxed) *spatial* similarity, as well as useful derivations, including *speed-pattern based*, *acceleration-pattern based* and *directional* similarity

To the best of our knowledge, this is the first work that decomposes the problem into different types of similarity queries based on various motion parameters of the trajectories. The synthesis of the operators under a unified trajectory management, knowledge discovery and visual analytics framework provides functionality so far unmatched in the literature. The efficiency and robustness of the operators have been proved experimentally by performing clustering and classification tasks to both real and synthetic trajectory datasets.

From a trajectory data management and mining perspective, the experiments proved the intuition that using various distance functions which take into account the motion parameters of the trajectories and their local properties provides added value when querying and analyzing complex datasets. From the visual analytics point of view, the analysis with real data showed the importance of interactive visual interfaces giving the user a high flexibility in applying diverse similarity measures for trajectories. Trajectories are complex spatio-temporal objects with multiple heterogeneous characteristics, which cannot be comprehended all at once. The analyst needs to consider these objects from several complementary perspectives, which are enabled by an array of distance functions dealing with different aspects and traits of trajectories. This approach can be generalized to other kinds of complex objects, e.g., graphs, images, etc. (of course, a different set of distance functions would be needed for a different kind of objects).

Clear future work objectives arise from this work. As already discussed, being non-metric implies that GenLIP and its variations are non-indexable by traditional distance-based indexing methods; therefore, we plan to investigate whether and how the proposed distance functions could be made to obey near triangle inequality (Chen et al. 2005), which would allow us to devise appropriate indexing structures in order to improve the overall performance of the operators. Independently from the above, we intend to design a large-scale benchmark of as many as possible proposed similarity functions against various real mobility datasets. Additionally, we will study thoroughly the quality of *LIPgrams* and utilize these similarity meta-data patterns so as to perform other mining tasks, like finding most frequent motion patterns. A different perspective is to extend our techniques in order to address the problem of similarity search for trajectories restricted in spatial networks (Tiakas et al. 2009). Finally, another interesting problem for future work is how to relate the results of several independent runs of clustering with different distance functions and how to visualize the relationships in order to help the user to compose a comprehensive view of a set of trajectories from several partial views resulting from each individual run.

Acknowledgements Research partially supported by the FP7 ICT/FET Project MODAP (Mobility, Data Mining, and Privacy) and the ESF/COST Action IC0903 MOVE (Knowledge Discovery from Moving Objects), both funded by the European Union. More information about these activities is available at www.modap.org and www.move-cost.info, respectively.

References

Agrawal, R., Faloutsos, C., & Swami, A. (1993). Efficient similarity search in sequence databases. In *Proceedings of fourth internationall conference foundations of data organization and algorithms*.

Andrienko, G., & Andrienko, N. (2010). A general framework for using aggregation in visual exploration of movement data. *The Cartographic Journal, 47*(1), 22–40.

Andrienko, G., Andrienko, N., & Wrobel, S. (2007). Visual analytics tools for analysis of movement data. *ACM SIGKDD Explorations, 9*(2), 38–46.

Ankerst, M., Breunig, M. M., Kriegel, H.-P., & Sander J. (1999). OPTICS: Ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD international conference on management of data*.

Berndt, J., & Clifford, J. (1996). Finding patterns in time series: A dynamic programming approach. In *Advances in knowledge discovery and data mining*. Menlo Park: AAAI/MIT Press.

Bollobas, B., Das, G., Gunopulos, D., & Mannila, H. (2001). Time-series similarity problems and well-separated geometric sets. *Nordic Journal of Computing, 8*, 409–423.

Brinkhoff, T. (2002). A Framework for generating network-based moving objects. *Geoinformatica, 6*(2), 153–180.

Brinkhoff, T. (2011). Network-based generator of moving objects. IAPG, Jade University Oldenburg, Germany. http://www.fh-oow.de/institute/iapg/personen/brinkhoff/generator/. Accessed 1 Feb 2011.

Cai, Y., & Ng, R. (2004). Indexing spatio-temporal trajectories with Chebyshev polynomials. In *Proceedings of the ACM SIGMOD international conference on management of data*.

Chan, K. P., & Fu, A. W.-C. (1999). Efficient time series matching by Wavelets. In *Proceedings of international conference on data engineering*.

Chan, T. M. (1994). A simple trapezoid sweep algorithm for reporting red/blue segment intersections. In *Proceedings of Canadian conference on computational geometry*.

Chazelle, B., & Edelsbrunner, H. (2002). An optimal algorithm for intersecting line segments in the plane. *Journal of the ACM, 39*(1), 1–54.

Chen, L., & Ng, R. (2004). On the marriage of edit distance and Lp norms. *International Journal on Very Large Data Bases, 11*, 28–46.

Chen, L., Tamer Özsu, M., & Oria, V. (2005). Robust and fast similarity search for moving object trajectories. In *Proceedings of the ACM SIGMOD international conference on management of data*.

CLUTO (2011). Karypis Lab, University of Minnesota, USA. http://glaros.dtc.umn.edu/gkhome/views/cluto/. Accessed 1 Feb 2011.

Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., & Keogh, E. (2008). Querying and mining of time series data: Experimental comparison of representations and distance measures. In *International conference on very large data bases*.

Douglas, D., & Peucker, T. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer, 10*(2), 112–122.

Frentzos, E., Gratsias, K., & Theodoridis, Y. (2007). Indexed-based most similar trajectory search. In *Proceedings of international conference on data engineering*.

Fu, A. W.-C., Keogh, E., Lau, L. Y. H., Ratanamahatana, C. A., Wong, R. C.-W. (2008). Scaling and time warping in time series querying. *The VLDB Journal, 17*, 899–921.

Giannotti, F., Nanni, M., Pinelli, F., & Pedreschi, D. (2007). Trajectory pattern mining. In *Proceedings of conference of knowledge discovery and data mining*.

Giannotti, F., & Pedreschi, D. (2008). *Mobility, data mining and privacy, geographic knowledge discovery*. New York: Springer.

Goh, K. S., Li, B., & Chang, T. (Eds.) (2002). *Dyndex: A dynamic and non-metric space indexer. Proceedings of International Conference of SIGMM*.

Goldin, Q., & Kanellakis, C. (1995). On similarity queries for time-series data: Constraint specification and implementation. *Lecture Notes in Computer Science, 976*, 137–153.

Huttenlocher, D. P., Klanderman, G. A., & Rucklidge, W. A. (1993). Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 15*(9), 850–863.

Keim, D. A. (2005). Scaling visual analytics to very large data sets. Presentation at visual analytics workshop. Available from http://infovis.uni-konstanz.de/events/VisAnalyticsWs05/index.php. Accessed 1 Feb 2011.

Keogh, E., & Kasetty, S. (2002). On the need for time series data mining benchmarks: A survey and empirical demonstration. In *Proceedings of conference of knowledge discovery and data mining*.

Korn, F., Jagadish, H., & Faloutsos, C. (1997). Efficiently supporting ad hoc queries in large datasets of time sequences. In *Proceedings of the ACM SIGMOD international conference on management of data*.

Kwan, M.-P., & Lee, J. (2004). Geovisualization of human activity patterns using 3-D GIS: A time-geographic approach. In M. F. Goodchild & D. G. Janelle (Eds.), *Spatially integrated social science*. New York: Oxford University Press.

Laube, P., Imfeld, S., & Weibel, R. (2005). Discovering relative motion patterns in groups of moving point objects. *International Journal of Geographical Information Science, 19*(6), 639–668.

Lee, J.-G., Han, J., & Whang, K.-Y. (2007). Trajectory clustering: A partition-and-group framework. In *Proceedings of the ACM SIGMOD international conference on management of data*.

Lee, S.-L., Chun, S.-J., Kim, D.-H., Lee, J.-H., & Chung, C.-W. (2000). Similarity search for multidimensional data sequences. In *Proceedings of international conference on data engineering*.

Lin, B., & Su, J. (2005). Shapes based trajectory queries for moving objects. In *Proceedings of the ACM annual international workshop on geographic information*.

Little, J. L., & Gu, Z. (2001). Video retrieval by spatial and temporal structure of trajectories. *Proceedings of SPIE, 4315*, 545–552.

Marketos, G., Frentzos, E., Ntoutsi, I., Pelekis, N., Raffaeta, A., & Theodoridis, Y. (2008). Building real-world trajectory warehouses. In *Proceedings of the seventh ACM international workshop on data engineering for wireless and mobile access*.

Meratnia, N., & de By, R. A. (2004). Spatiotemporal compression techniques for moving point objects. In *Proceedings of the international conference on extending data base technology*.

Nanni, M., & Pedreschi, D. (2006). Time-focused clustering of trajectories of moving objects. *Journal of Intelligent Information Systems, 27*(3), 267–289.

Pelekis, N., Frentzos, E., Giatrakos, N., & Theodoridis, Y. (2008). HERMES: Aggregative LBS via a trajectory DB engine. In *Proceedings of the ACM SIGMOD international conference on management of data*.

Pelekis, N., Frentzos, E., Giatrakos, N., & Theodoridis, Y. (2011). HERMES: A trajectory DB engine for mobility-centric applications. *International Journal of Knowledge-based Organizations*, in press.

Pelekis, N., Kopanakis, I., Kotsifakos, E., Frentzos, E., & Theodoridis, Y. (2009). Clustering trajectories of moving objects in an uncertain world. In *Proceedings of international conference on data mining*.

Pelekis, N., Kopanakis, I., Ntoutsi, I., Marketos, G., Andrienko, G., & Theodoridis, Y. (2007). Similarity search in trajectory databases. In *Proceedings of the international symposium on temporal representation and reasoning*.

Pelekis, N., & Theodoridis, Y. (2006). Boosting location-based services with a moving object database engine. In *Proceedings of the international workshop on data engineering for wireless and mobile access*.

Pelekis, N., Theodoridis, Y., Vosinakis, S., & Panayiotopoulos, T. (2006). Hermes—A framework for location-based data management. In *Proceedings of international conference on extending database technology*.

R-tree Portal (2011). InfoLab, University of Piraeus, Greece. http://www.rtreeportal.org. Accessed 1 Feb 2011.

Rafiei, D., & Mendelzon, A. O. (2002). Efficient retrieval of similar shapes. *The VLDB Journal, 11*(1), 17–27.

Rinzivillo, S., Pedreschi, D., Nanni, M., Giannotti, F., Andrienko, N., & Andrienko, G. (2008). Visually–driven analysis of movement data by progressive clustering. *Information Visualization, 7*(3/4), 225–239.

Sakurai, Y., Yoshikawa, M., & Faloutsos, C. (2005). FTW: Fast similarity search under the time warping distance. In *Proceedings of the twenty-fourth acm sigmod-sigact-sigart symposium on principles of database systems*.

Thomas, J. J., & Cook, K. A. (Eds.) (2005). Illuminating the path. The research and development agenda for visual analytics. Washington, DC: IEEE Computer Society.

Tiakas, E., Papadopoulos, A. N., Nanopoulos, A., Manolopoulos, Y., Stojanovic, D., & Djordjevic-Kajan, S. (2009). Searching for similar trajectories in spatial networks. *Journal of Systems and Software, 82*(5), 772–788.

Trajcevski, G., Ding, H., Scheuermann, P., Tamassia, R., & Vaccaro, D. (2007). Dynamics-aware similarity of moving objects trajectories. In *Proceedings of ACM international conference on geographic information systems*.

Vlachos, M., Gunopulos, D., & Das, G. (2002a). Rotation invariant distance measures for trajectories. In *Proceedings of conference of knowledge discovery and data mining*.

Vlachos, M., Kollios, G., & Gunopulos, D. (2002b). Discovering similar multidimensional trajectories. In *Proceedings of international conference on data engineering*.

Yanagisawa, Y., Akahani, J., & Satoh, T. (2003). Shape-based similarity query for trajectory of mobile objects. In *Proceedings of the international conference on mobile data management*.

Yi, B.-K., Jagadish, H., & Faloutsos, C. (1998). Efficient retrieval of similar time sequences under time warping. In *Proceedings of international conference on data engineering*.

Yu, H. (2006). Spatial-temporal GIS design for exploring interactions of human activities. *Cartography and Geographic Information Science, 33*(1), 3–19.