# Towards Semantic Trajectory Outlier Detection

**Artur Ribeiro de Aquino**[1], **Luis Otavio Alvares** [1], **Chiara Renso** [2],**Vania Bogorny**[1]

[1]Departamento de Informatica e Estatistica – UFSC

[2]KDD Lab - Pisa, Italy

artur.aquino@posgrad.ufsc.br, alvares@inf.ufsc.br

renso.chiara@isti.cnr.it,vania.bogorny@ufsc.br

***Abstract.*** *Only a few works in trajectory data mining have focused on outlier detection, and to the best of our knowledge, no works so far have made a deeper analysis to either understand or to give a meaning to the outliers. In this paper we present an algorithm to add meaning to trajectory outliers considering three main possible reasons for a detour: stops outside the standard route, events, and traffic jams in the standard path. We show with experiments on real data that the method correctly finds the different types of outliers.*

## 1. Introduction and Motivation

The current advances in mobile technology made mobility traces more present. As a consequence, there is an increasing need for developing new methods for interpreting these traces to provide more information to the decision maker.

Mobility traces, well known as trajectories of moving objects, are collected as raw data, with the position of the object in space and time. Several works have already been developed for trajectory data analysis. Different types of patterns can be extracted from mobility data, but only a few attempts have been made to either discover the meaning of a pattern or the reason of certain behaviors of moving objects. This is specially true for trajectory outliers. Existing works for trajectory outlier detection as ([Lee et al. 2008], [Yuan et al. 2011], [Chen et al. 2011] look for trajectories that simply behave differently from the majority of the trajectories in a dataset, but no further analysis is performed to discover when the outliers occur or which are the reasons for a different behavior. Trajectory outliers can be interesting to discover suspicious behaviors in a group of people, to find alternative routes in traffic analysis or to reveal the best or worse paths that connect regions of interest. The interpretation of outliers can provide more information to the decision maker and help to answer questions like: has a driver deviate from the main group because he needed to pickup up someone nearby? Was he avoiding a police check? Is this an alternate route to reach a specific place? Is that a suspicious driver? More information about an outlier can be useful to answer these questions in different application domains. However, to discover why and object followed a different route is very complex, since normally only the raw trajectory data are available and no information is given about the moving object and his intents.

In this paper we try to go one step further in trajectory outlier detection, aiming to infer the possible reason why an object made a different movement. More specifically, we extend the work of Fontes [Fontes et al. 2013], which finds outliers between regions of interest. Figure 1 shows an example of trajectories moving from region $R_1$ to region $R_2$, where most of the trajectories follow the same path and two objects deviate from this path, $c_1$ and $c_5$, which are the *outliers*.

In this paper we consider three main aspects which could be the reason of an outlier: (i) an event on the standard path, (ii) a traffic jam in the standard path, and (iii) a stop in the outlier. In summary, we make the following contributions to the state of the art in trajectory outlier detection: try to interpret the reason of an outlier, define different types of trajectory outliers and propose an algorithm to classify the outliers.

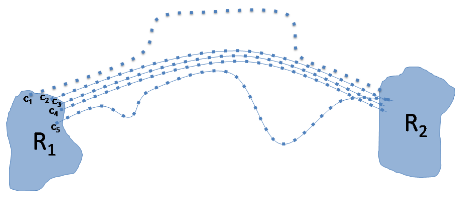

**Figure 1. Example of outliers.**

The rest of the paper is organized as follows: Section 2 presents the related works, Section 3 presents the main definitions used as base to the new concepts and algorithm presented in Section 4. Section 5 presents experiments on real trajectory data. Finally, Section 6 concludes the paper and suggests directions of future research.

## 2. Related Works

Only a few works are specifically developed for finding outliers in trajectories [Lee et al. 2008], [Yuan et al. 2011], and [Chen et al. 2011]. The first two approaches split trajectories in subtrajectories to find outliers. Outliers are the trajectories with a fraction of partitions distant from other partitions and must have a certain length. The distance function considers position and direction. In both approaches time is not taken into account and no standard path must exist for trajectories to be classified as outliers. None of these algorithms consider any semantic information or give more meaning to the outliers, their concern is more geometrical, while our work focuses on the meaning.

The last work [Chen et al. 2011] is the closest to the work of this paper. It proposes an algorithm to identify the most popular routes in a trajectory dataset. They build a graph where the nodes are the starting and ending points of the trajectories and the places where the trajectories cross each other, and the arrows are the possible paths from one node to another. For each node the probability is calculated. The standard paths and outliers can be easily derived from this probability. Its purpose is very similar to the work of Fontes [Fontes et al. 2013], but no further analysis is performed over the outliers.

The approaches of [Alvares et al. 2011] can also be used to find outliers. The algorithm proposed by [Alvares et al. 2011] finds trajectories that avoid or deviate from target objects as surveillance cameras, traffic jams, or other pre-defined static objects. It verifies for each trajectory if it avoids static objects and if there is a valid path that crosses the region of the avoided object. This work looks for outliers between trajectories and static objects, while we deal with outliers among trajectories. Indeed, no further interpretation is made on the outliers.

Gupta in [Gupta et al. 2013] presents a survey on the most recent approaches for outlier detection in temporal and spatio-temporal data.

[Fontes et al. 2013] finds outliers between regions of interest. In a first step the algorithm extracts the candidates, that are the subtrajectories moving between pairs of regions. In a second step the standards and the outliers are discovered. In this paper we extend the work of [Fontes et al. 2013] to find the specific standard path that the outlier deviated and propose to give a meaning to the outliers looking for episodes (i.e. events and traffic jam) in the standard path.

## 3. Basic Concepts

In this section we introduce some basic concepts about trajectories that will help to understand our approach. Most of these concepts are based on Fontes [Fontes et al. 2013], which is the work we extended here. We start the definitions with the very basic concept of point.

**Definition 1** (Point)**.** A point $p$ is a tuple $(x, y, t)$, where $x$ and $y$ are spatial coordinates and $t$ is the time instant in which the coordinates were collected.

A list of points ordered in time is a trajectory.

**Definition 2** (Trajectory)**.** A trajectory $T$ is a list of points $\langle p_1, p_2, p_3, ..., p_n\rangle$, where $p_i = (x_i, y_i, t_i)$ and $t_1 < t_2 < t_3 < \ldots < t_n$.

In general, trajectory patterns do not exist in the whole trajectory or during the complete trajectory life. Trajectory patterns occur in parts of the trajectories, therefore, we make use of subtrajectories, that is a well known concept in trajectory research.

**Definition 3** (Subtrajectory)**.** A subtrajectory $S$ of $T$ is a list of consecutive points $\langle p_k, p_{k+1}, ..., p_{k+l}\rangle$, where $p_i \in T, k \geq 1$, and $k + l \leq n$.

In this work we try to understand the reason why an object made a detour in relation to the path followed by the majority of the trajectories to move between regions of interest. The trajectories that intersect a pair of regions are the candidates [Fontes et al. 2013], as shown in Figure 1(five candidates).

A candidate is the smallest subtrajectory that moves between two regions. It corresponds to the last point of the subtrajectory that intersects the first region and the first point that intersects the final region, as shown in Figure 2 (left).

**Definition 4** (Candidate)**.** Let $R_1$ and $R_2$ be two regions such that $R_1 \cap R_2 = \emptyset$ and $T$ a trajectory. A candidate from $R_1$ to $R_2$ is the subtrajectory $S = \langle p_i, p_{i+1}, ..., p_m\rangle$ of $T$, where $(S \cap R_1) = \{p_i\}$ and $(S \cap R_2) = \{p_m\}$.

A candidate that moves from a region R1 to a region R2 is different from a candidate that moves in the opposite direction. In the example of Figure 2 (left) the movement is from $R_1$ to $R_2$, thus the candidate has the points from $p_i$ to $p_m$.

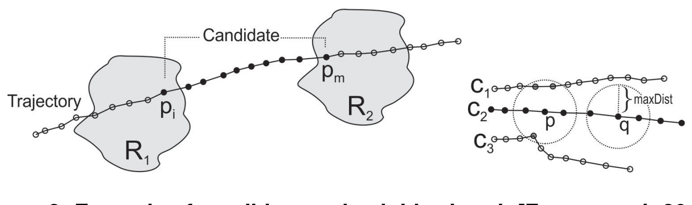

**Figure 2. Example of candidate and neighborhood. [Fontes et al. 2013]**

According to [Fontes et al. 2013], outliers are discovered among candidates that move in the same direction. The first characteristic that a candidate should have to be an outlier is to move apart from other candidates. Therefore, for each point of a candidate its neighbors are computed. A candidate is a neighbor of a point if it is close to the point. If a point has a few candidates in its neighborhood, then at that time the moving object was following a path different from the majority of the candidates. The maximal distance (Euclidean distance) for a candidate to be a neighbor of a point is called $maxDist$.

**Definition 5** (Neighborhood)**.** Let $p$ be a point. The neighborhood of a point $p$ wrt to $maxDist$ is
$N(p, maxDist) = \{c_i | c_i$ is a candidate and $\exists q \in c_i, dist(p, q) \leq maxDist\}$.

Here we consider spatial neighborhood, because we are interested in all candidates that move together in space. In order to find *traffic avoiding outliers* (as can be seen in Section 4) we analyze the duration of both synchronized and non-synchronized candidates. Figure 2 (right) shows an example of neighborhood. The neighborhood of point $p$ are the candidates $c_1$ and $c_3$, since these two candidates have at least one point inside the radius of size $maxDist$ around $p$. Point $q$ has no candidates inside its radius of size $maxDist$, so its neighborhood is empty. We can conclude that at point $p$, $c_2$ was moving with $c_1$ and $c_3$ (same path), but at point $q$, $c_2$ was moving far from $c_1$ and $c_3$ (different path).

When each point of a candidate has a number of candidates in its neighborhood that is higher than a threshold for minimal support, called *minSup*, then this candidate is called a *standard*.

**Definition 6** (Standard)**.** Let $c = \langle p_1, p_2, ..., p_n \rangle$ be a candidate, $c$ is a standard candidate wrt $maxDist$ and $minSup$ if and only if $\forall p_i \in c$,
$|N(p_i, maxDist)| \geq minSup$, where $|X|$ means the cardinality of $X$.

Figure 1 shows an example of standard candidates, where c2, c3 and c4 are always moving together. So in this example there are three standard candidates considering *minSup*=2. In the following section we present the new concepts, including a new definition for outlier and an algorithm to add meaning to the outliers.

## 4. Adding Meaning to Outliers

The main goal of this paper is to add meaning to the outliers. The reasons for an outlier are broad, and can be related to several things. In this paper we consider three main cases that can be the reason of an outlier: (a) stops in the outlier trajectories, where the moving object had the intent to stop somewhere else out of the standard path, (b) there is an event in the standard path that could block it or cause traffic jams in the area of the standard path, (c) a traffic jam in the standard path. We name these three types of outliers as *stop outlier*, *event outlier* and *traffic outlier*, respectively. Each case is detailed in the following subsections.

Before going into detail of the three cases of outliers, we define the concept of *standard path*, redefine the concept of outlier, and define an outlier segment. These definitions are needed because it is necessary to know which standards move in the same path that the outlier deviated. Two standards can be directly connected or reachable from another standard.

**Definition 7** (Directly Connected)**.** A standard $d$ is directly connected to a standard $e$ wrt maxDist if $\forall p_i \in d,\ e \in N(p_i, maxDist)$.

**Definition 8** (Reachable)**.** A standard $d$ is reachable from a standard $e$ wrt maxDist if there is a chain of standards $d_1, d_2, ..., d_n$ where $d_1 = e, d_n = d$ such that $d_{i+1}$ is directly connected from $d_i$.

When two or more standards are reachable from any other standard we can define a standard path.

**Definition 9** (Standard Path)**.** Let $D$ be a set of standards moving from region $R_1$ to region $R_2$. A standard path $H$ wrt maxDist is a non-empty subset of $D$ satisfying the following conditions:

- $\forall d, e \in D$: if $d \in H$ and $e$ is reachable from $d$ wrt maxDist, then $e \in H$.
- $\forall d, e \in H$: $d$ is reachable from $e$ wrt maxDist.

In this work we do not consider an outlier when only a few points of a trajectory deviated the standard path. Therefore, we define the concept of outlier such that it should have a minimal deviation length, called *minLenght*.

**Definition 10** (Outlier)**.** Let $R_1$ and $R_2$ be two regions and $C$ the set of candidates from $R_1$ to $R_2$. A candidate $o \in C$ is an outlier wrt $maxDist$, $minSup$ and $minLenght$ if $\exists c \in C$ . $c$ is a standard $\wedge$ $\exists s$ .$s$ is a subtrajectory of $o$, $s = \langle p_i, p_{i+1}, \ldots, p_n \rangle$. $\forall p_k \in s$, $|N(P_k, maxDist)| < minSup \wedge \sum_{j=i}^{n-1} dist(p_j, p_{j+1}) > minLenght$.

Figure 1 shows an example of outlier, considering $minSup = 60\%$. In this example there are 5 candidates that move from $R_1$ to $R_2$, where $c_1$ and $c_5$ are the outliers and $c_2$, $c_3$ and $c_4$ are the standard path from $R_1$ to $R_2$.

In order to interpret the meaning of an outlier, we want to analyze only the part of the outlier trajectory that made the detour. Figure 3 (left) shows an outlier example where the subtrajectories from $p_1$ to $p_7$ and from $q_1$ to $q_14$ will be analyzed. These subtrajectories are called *outlier segments*.

**Definition 11** (Outlier Segment)**.** Let $o$ be an outlier. Let $s = \langle p_i, p_{i+1}, ..., p_n \rangle$ be a subtrajectory of $o$. $s$ is an outlier segment wrt $maxDist$, $minSup$, and $minLenght$ if $\forall p_k \in s$, $\mid N(p_k, maxDist) \mid < minSup$ and $\mid N(p_{i-1}, maxDist) \mid \geq minSup$ and $\mid N(p_{n+1}, maxDist) \mid \geq minSup$ and $\sum_{j=i}^{n-1} dist(p_j, p_{j+1}) > minLenght$.

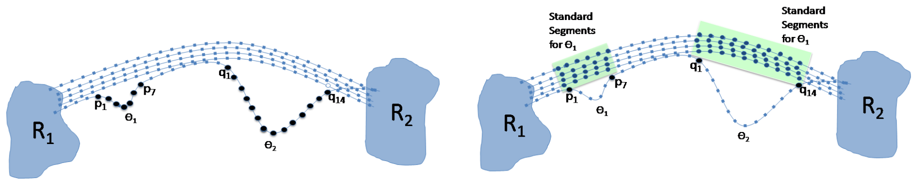

**Figure 3. Example of outlier segments and standard segments.**

In the example of Figure 3 (left), if *minLenght* is defined as 10%, both $\theta_1$ and $\theta_2$ are outlier segments. If *minLenght*, for instance, is defined as 30%, only $\theta_2$ is an outlier segment.

Having defined the outlier segments we try to interpret them looking for stops, traffic jams and events around these segments, as detailed in the following subsections.

## 4.1. Stop Outliers

A stop outlier occurs when the moving object made a stop for some time during the deviation. She/he had an appointment, a meeting, or something to do somewhere else that was not in the standard path. This is an intentional detour with a reason. Very common examples can be go shopping after work, pick up the children at school, go to a happy hour with friends, pass by a bakery, visit a friend or relative, or having a work meeting. When these objectives are out of the standard path, they can be the reasons for the outliers.

To discover if an outlier has a stop we need to look for stops not in the complete outlier trajectory, but only in the subtrajectory that corresponds to the outlier (deviation), i.e., the outlier segment. We consider as a stop a subtrajectory that had speed close to zero for a minimal amount of time (*minTime*). We consider a $maxSpeed$ threshold and not the value zero because, in reality, due to GPS imprecision, it is not common to have points with exact the same coordinates when the object has stopped.

**Definition 12** (Stop)**.** Let $\theta$ be an outlier segment. A subtrajectory $s \subseteq \theta$, $s = \langle p_1, p_2, ..., p_n \rangle$ is a stop of $\theta$ wrt $minTime$ and $maxSpeed$ if $t_n - t_1 \geq minTime$ and $\frac{\sum dist(p_i, p_{i+1})}{t_n - t_1} \leq maxSpeed$

**Definition 13** (Stop Outlier)**.** An outlier segment $\theta$ is a stop outlier iff it made a stop.

In the following section we detail the event outliers.

## 4.2. Event Outliers

The stop outlier is the most simple case of an outlier. The second case is more complex. An *event avoiding outlier* is a detour from the standard path because an event is going on close to the standard path. The complexity starts because there might be more than one standard path connecting two regions in the same direction, as shown in Figure 4 (left), or two standard paths may start together and split later, as shown in Figure 4 (right).

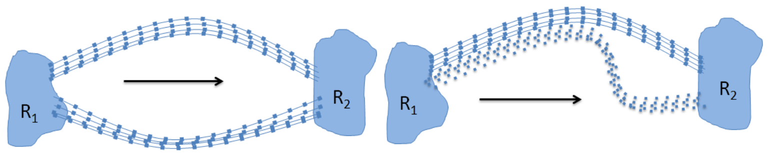

**Figure 4. Examples of standard paths from $R_1$ to $R_2$.**

An outlier is an event outlier when the standard path has an event nearby, i.e., in the area where the outlier avoided the standard path. However, to discover if the standard path has an event nearby is not a trivial task. We cannot simply look if there is an event at any part of the standard path. For an outlier to avoid an event in the standard path it is necessary to analyze only the part of the standard path that was deviated by the outlier. Therefore, we first need to discover which standard path an outlier segment has avoided. Then we need to find the corresponding *segments* of the standard path that the outlier deviated, since the event should be around these segments.

As there might be more than one standard path connecting the regions, we look for the standard path closest to the outlier, in order to analyze this area. Figure3 (right) shows two examples where the outlier segments $\theta_1$ and $\theta_2$ deviated from a different set of standard segments in the same standard path. The segments in a standard path that the outlier deviated are called *standard segments*.

**Definition 14** (Standard Segment)**.** Let $o$ be an outlier, $\theta = \langle p_1, p_2, \ldots, p_n \rangle$ an outlier segment of $o$, $p_0$ the point of $o$ immediately before $p_1$, $p_{n+1}$ the point of $o$ immediately after $p_n$ and $d$ a standard in the same path as $p_0$ and $p_{n+1}$. A subtrajectory $s = \langle p_k, \ldots, p_l \rangle$

of $d$ is a standard segment of $\theta$ if and only if $s \subset d$ and $p_k = closest(p_0, d) \wedge |s| < |d| \wedge p_l = closest(p_{n+1}, d)$, where $closest(p, c)$ is a function that returns the closest point of a candidate $c$ from a point $p$. If $p_0$ does not exist then $p_k$ is the first point of $d$. If $p_{n+1}$ does not exist then $p_l$ is the last point of $d$.

An event is represented by its region of effect and the times in which the effect starts and ends. The effect of an event is its influence on the movement around.

**Definition 15** (Event)**.** An event $e$ is a triple $\langle R, t_0, t_f \rangle$, where $R$ is a region, and $t_0$ and $t_f$ are the starting and ending time, respectively.

In an event outlier, the outlier segment should not intersect the event, that must happen at the same time of the deviation, but the event should intersect the standard segment (Figure 5 (right) shows an example). Moreover, there must be a standard segment that is synchronized with the outlier to be sure that the standard path was free during the deviation. All standard segments that are close to the outlier in the time that the outlier started, are called *synchronized standard segments*. Figure 5 (left) shows an example of three synchronized standard segments with respect to outlier $\theta_2$ and one non synchronized.

**Definition 16** (Synchronized Standard Segment)**.** Let $D$ be the set of standard segments of outlier segment $\theta$ moving in the same standard path. A subtrajectory $s$ is a synchronized standard segment with respect to $\theta$ when $s \in D$ and $abs(s.t - \theta.t) \leq timeTol$, where $s.t$ and $\theta.t$ represent the time of the first point of $s$ and $\theta$, respectively.

When there is at least one standard segment that intersects an event but the outlier segment does not, and there is an overlap between the outlier segment and the event times, then the outlier segment is called *event avoiding outlier*.

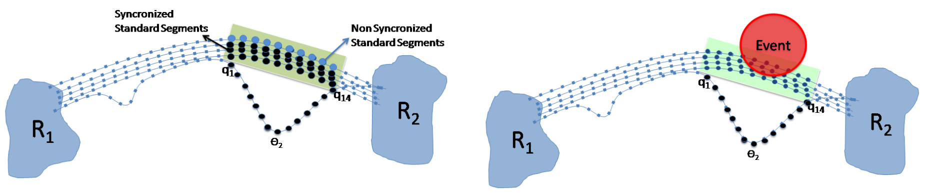
Syncronized Standard Segments
Non Syncronized Standard Segments
q1
q14
θ2
R1
R2
Event
q1
q14
θ2
R1
R2

**Figure 5. Example of traffic avoiding outlier and event avoiding outier.**

**Definition 17** (Event Avoiding Outlier)**.** Let $e$ be an event, $R_e$ its region, $t_{e0}$ its start time and $t_{e_f}$ its end time. Let $d$ be a standard segment of an outlier segment $\theta$, $t_{\theta_0}$ its start time and $t_{\theta_f}$ its end time. The outlier segment $\theta$ is an event avoiding outlier if and only if it is not a stop outlier and $R_e \cap d \neq \emptyset \wedge R_e \cap \theta = \emptyset \ \wedge \ (t_{e_0} < t_{\theta_0} < t_{e_f})$.

In the following section we explain a traffic avoiding outlier.

## 4.3. Traffic Avoiding Outlier

The last possibility treated so far is that an outlier may be deviating from a traffic jam in the standard path. Normally, traffic jams at the main paths or roads in a city are well known by most people, mainly at rush hours. However, some objects may follow the standard path anyway, while others that know alternative routes leading to the destination may prefer them. Therefore, in this section we look for slow traffic in the standard path at the time of the outlier. Here we assume that the outlier has no stop in its subtrajectory and there is no event around the standard path.

To measure the time on the standard path at the same moment that the outlier deviated from it we need to look only at the synchronized standard segments. Figure 5 (left) shows an example of outlier ($\theta_2$) and the respective synchronized and non-synchronized standard segments. For each outlier segment, the average speed of all synchronized standard segments in the same standard path are compared to the speed of the non-synchronized standard segments in the same path. This way it is possible to know if at the moment of the deviation the traffic was slower than normal traffic (fastest segments). We consider that there is a traffic jam when the average speed of the synchronized standard segments is less than half of the average speed of the non-synchronized standard segments. If this is the case, then we define the outlier as a traffic avoiding outlier.

**Definition 18** (Traffic Avoiding Outlier)**.** Let $\theta$ be an outlier segment, $M$ a set of synchronized standard segments with $\theta$ and $N$ a set of non-synchronized standard segments of $\theta$. $\theta$ is a traffic avoiding outlier if and only if it is neither a stop outlier nor an event avoiding outlier and $avgSpeed(M)/avgSpeed(N) \leq 0.5$.

A trajectory can have many outlier segments, and each one is interpreted independently. This means that a trajectory that has 3 outlier segments may have, for instance, 3 types of outliers. After defining the three types of semantic outliers we can finally present an algorithm to automatically interpret the outliers.

## 4.4. Algorithm

In this section we present an algorithm to interpret the outliers. Algorithm 1 shows the pseudo-code of the main algorithm. The input of the algorithm are thresholds detailed in the definitions. The output is a set of classified outliers. The first step is to find the outlier segments and the standard segments (lines 16 and 17), according to Definition 11 and Definition 14, respectively.

**Algorithm 1. Main algorithm pseudocode**

```
1  INPUT:
2    C // set of candidates between 2 regions
3    E // set of Events
4    maxDist // for neighborhood
5    minLenght  // for the outlier segments
6    minSup // for a standard path
7    minTime // for a stop
8    timeTol // for synchronized  standard segments
9
10 OUTPUT:
11   SO  // set of stop outliers
12   EAO // set of event avoiding outliers
13   TAO // set of traffic avoiding outliers
14
15 METHOD:
16 OutSegs = findOutlierSegments(C, maxDist, minSup);
17 StandardSegs = findStandardSegments(C,OutSegs,maxDist,minSup);
18 SO = findStopOutlier(OutSegs,mintime);
19 EAO = findEventAvoidingOutlier(OutSegs,StandardSegs, E, SO);
20 TAO = findTrafficAvoidingOutlier(OutSegs,StandardSeg, SO,EAO,timeTol);
21 RETURN SO, EAO, TAO
```

The function findStopOutlier() (line 18) computes the stop outliers checking if each outlier segment has a stop for at least $minTime$, according to Definition 12 (Stop). If the outlier segment has a stop, it is a stop outlier. The maxSpeed threshold used in Definition 12 is computed as 5% of the average speed of the candidate.

The functions findEventAvoidingOutlier() (line 19) and findTrafficAvoidingOutier() (line 20) are detailed in algorithm 2 and algorithm 3, respectively. In order to find event avoiding outliers, the first step of algorithm 2 is to find all the outlier segments without stops (line 11). Then, for each outlier segment, the algorithm gets its standard segments (line 13) according to Definition 14 and checks the intersection with the set of events (line 14). If there is an event which intersects any of these standard segments without intersecting the outlier segment and at least part of the event happens during the detour, then it is added to the event avoiding outlier list (line 16).

**Algorithm 2. findEventAvoidingOutlier pseudocode**

```
1  INPUT:
2    OutSegs // set of outlier segments
3    StandardSegs // set of standard segments
4    E // set of events
5    SO // set of stop outliers
6
7  OUTPUT:
8    EAO // set of event avoiding outliers
9
10 METHOD:
11 O = OutSegs − SO;
12 FOR EACH(o in O)
13   S = StandardSegs.getStandardSegments(o);
14   e = getIntersection(S, E);
15   IF(e != NULL && !hasIntersection(o, e) && timeOverlaps(o, e))
16     EAO.add(o);
17   ENDIF
18 ENDFOR
19 RETURN EAO;
```

The function findTrafficAvoidingOutlier() is shown in algorithm 3. The first step is to remove from the set of outlier segments the ones that are stop outliers and event outliers (line 12). Then, the algorithm computes the standard segments that are synchronized (according to Definition 16) and the ones not synchronized (lines 14 and 15). For the non-synchronized the algorithm considers 5% of the fastest standard segments. This way we obtain the speed of the path when there is no traffic. Then the algorithm is able to compare the average speed of both synchronized and non synchronized segments, and infer if there was a traffic jam at that moment (line 16). When the average time of the synchronized standard segments is half of the average of the set of non-synchronized ones, we say that there was a traffic jam at that moment in the standard path, thus the outlier is classified as a traffic avoiding outlier.

**Algorithm 3. findTrafficAvoidingOutlier pseudocode**

```
1
2  INPUT:
3    OutSegs // set of Outlier Segments
4    StandardSegs // set of Standard Segments
5    SO // set of stop outliers
6    EAO // set of event avoiding outliers
7
8  OUTPUT:
9    TAO // set of traffic avoiding outliers
10
11 METHOD:
12 O = OutSegs − SO − EAO;
13 FOR EACH(o in O)
14   sync = StandardSegs.getSyncStandardSegments(o);
15   notSync = StandardSegs.getNotSyncStandardSegments(o);
16   IF(avgtime(sync.time) < avgtime(notSync.time)/2)
17     TAO.add(o);
18   ENDIF
19 ENDFOR
20 RETURN TAO;
```

## 5. Experimental Results

In this section we present the results of preliminary experiments on the taxi trajectory dataset collected in San Francisco, California, in May and June 2008 [Crawdad 2013]. This dataset is interesting for analyzing outliers because taxi drivers, in general, know several paths to reach the same place. Therefore, we want to find the alternative routes (outliers) in relation to the standard path and to discover if the alternative route was made to avoid an event in the standard path, to avoid low traffic in the standard path or if the taxi driver had a stop in the detour.

This dataset contains trajectories of taxi drivers during one month, with over 11 million points. One trajectory corresponds to the movement of one taxi driver during several days (with a trajectory identifier for each driver, and not each trajectory), having very long trajectories with sampling rate around 1 minute. Even with such a large time interval between every two points the method was able to find semantic outliers. In order to analyze the behavior of the driver between regions of interest we split the trajectories using the occupation attribute (which states if the taxi has passengers or not), so instead of having only one trajectory of a driver, we have several trajectories of the same object.

When analyzing trajectory patterns we should consider the time periods, since the movement patterns can be different during the day and night, and during weekdays and weekends. In this experiment we separated trajectories of weekdays and weekends, and because of space limitations, we analyzed only the trajectories from Monday to Friday. After this preprocessing step a set of 537.098 trajectories with 6.314.120 points was obtained.

We selected as regions of interest the Airport of San Francisco and a downtown area where most hotels are located, because there is a high taxi flow between these regions. To find the standard path and the outliers we considered 120 meters as *maxDist*, 5% as $minSup$ (number of candidates in the neighborhood for discovering a standard path between the regions) to find the standard path, and $minLenght$ as 10%, i.e., at least 10% of a candidate should be moving far from the standard path in order to be considered an outlier. Figure 6 (left) shows the standard path from the Airport to the central area.

In order to evaluate the method proposed in this paper, which is to give a meaning to the outliers, we simulated an event at Bayshore Freeway (US 101) with start time 17:30 and end time at 21:30. For discovering stop outliers we considered 15 minutes ($minTime$) as the minimal time for an outlier to be a stop outlier and 15 minutes as $timeTol$ for synchronized trajectories.

The method discovered 73 stop outliers (for $minTime$ 15 minutes), 6 traffic avoiding outliers and one event avoiding outlier. Because of space limitations we show one example of each type of outlier. Figure 6 (right) shows an event avoiding outlier, where the method correctly detected the outlier which deviated the standard path with an event. The outlier segment is represented in the figure by the triangles, the event by a circle in the standard path, and the standard segments are represented in black.

Figure 7 (left) shows a stop outlier, where the stop in the outlier segment had a duration of 44:13 minutes. The stop in the outlier segment is zoomed in the figure. Figure 7 (right) shows an example of traffic avoiding outlier, highlighting the outlier segment (triangles) and the corresponding standard segments (black). In this case, the average time of the synchronized standard segments was 7:05 minutes while the average duration of the non-synchronized segments was 3:40 minutes, characterizing a small traffic jam in the standard path at the moment of the outlier. Most detected traffic jams were near the downtown area, and not in the main road from the airport. Therefore, for this execution we reduced the parameter $minLenght$ in order to detect short detours as the one shown in Figure 7 (right).

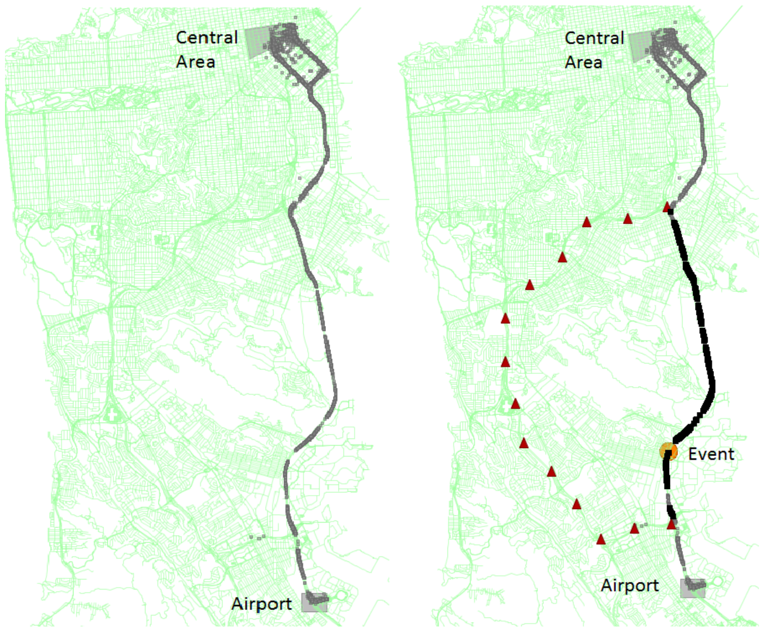

**Figure 6. (left) Standard path from Airport to Central area and (right) event outlier.**

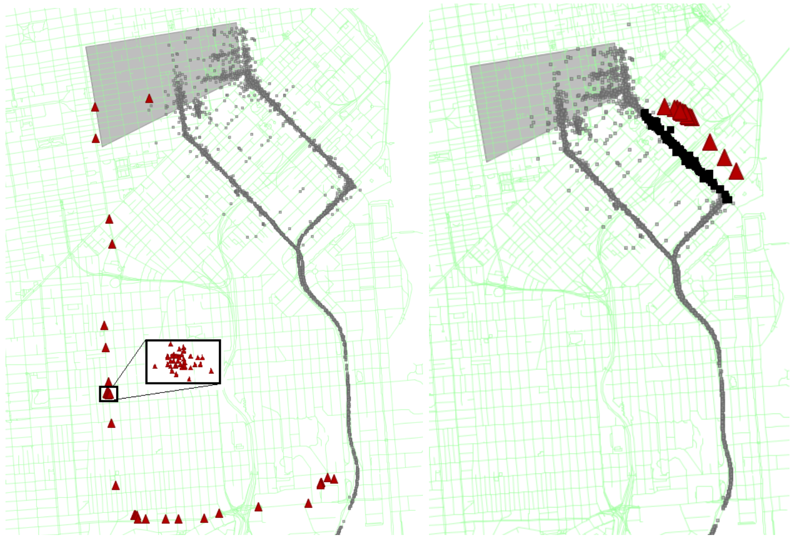

**Figure 7. (left) Stop outlier and (right)Traffic avoiding outlier**

## 6. Conclusion and Future Works

Several algorithms have been proposed for trajectory data mining, but only a few consider trajectory outlier detection. Existing approaches for trajectory outliers do not make deeper analysis of the discovered patterns to give more meaning or semantics. In this paper we look for outliers among trajectories that move between the same regions and try to interpret them.

For all outliers the proposed method finds and interprets each outlier segment, which is the part of the outlier that corresponds to the detour itself. So far the interpretation is separated in three cases which represent some possibilities of deviations: *stop outliers*, *event avoiding outliers*, and *traffic avoiding outliers*.

Future work includes outlier classification according to the type of detour made for each segment, and to give weights for each outlier, depending on the types of its segments. Indeed, more experiments have to be performed to better evaluate the method and the parameters need to be better studied and reduced.

## 7. Acknowledgments

This work has been partially supported by EU project FP7-PEOPLE SEEK (N. 295179 http://www.seek-project.eu) and the Brazilian agencies CAPES and CNPq. Authors would like to thank also for the partial support from CNR-CNPQ Joint project 2012, DataSIM FP7-ICT-270833, and UFSC.

## References

Alvares, L. O., Loy, A. M., Renso, C., and Bogorny, V. (2011). An algorithm to identify avoidance behavior in moving object trajectories. *J. Braz. Comp. Soc.*, 17(3):193–203.

Chen, Z., Shen, H. T., and Zhou, X. (2011). Discovering popular routes from trajectories. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 900–911. IEEE.

Crawdad (2013). Crawdad a community resource for archiving wireless data at dartmouth. `http://crawdad.cs.dartmouth.edu/meta.php?name=epfl/mobility`. Accessed: 2013-05-10.

Fontes, V. C., de Alencar, L. A., Renso, C., and Bogorny, V. (2013). Discovering trajectory outliers between regions of interest. In *GeoInfo*.

Gupta, M., Gao, J., Aggarwal, C. C., and Han, J. (2013). Outlier detection for temporal data: A survey. *TKDE*, 25.

Lee, J.-G., Han, J., and Li, X. (2008). Trajectory outlier detection: A partition-and-detect framework. In Alonso, G., Blakeley, J. A., and Chen, A. L. P., editors, *ICDE*, pages 140–149. IEEE.

Yuan, G., Xia, S., Zhang, L., Zhou, Y., and Ji, C. (2011). Trajectory outlier detection algorithm based on structural features. *Journal of Computational Information Systems*, 7(11):4137–4144.