Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/090202

SPET Algorithm: Stop and Proximity Episodes in Trajectories

Francisco Moreno[1,*], *Anderson Castaño*[1] *and Francisco Javier de Cos Juez*[2]

[1] Departamento de Ciencias de la Computación y la Decisión, Universidad Nacional de Colombia, Sede Medellín, Colombia
[2] Mining Exploitation and Prospecting Department, Universidad de Oviedo, 33004, Spain.

Received: 3 May 2014, Revised: 4 Aug. 2014, Accepted: 6 Aug. 2014
Published online: 1 Mar. 2015

Abstract: In this paper, we propose the SPET (Stop and Proximity Episodes in Trajectories) algorithm to identify stop and proximity episodes in trajectories. A trajectory is the record of the evolution of the position of an object that is moving in space during a given time interval in order to achieve a goal. A stop is an episode of a trajectory during which the object remained continuously inside a point of interest (POI) a minimum time (specified by the business analysts) and a proximity is an episode of a trajectory during which the object remained continuously near a POI a minimum time. These episodes may help to understand the behavior of moving objects in several domains. For example, proximities episodes can help in advertising, where agents can identify appropriate spots in order to try to increase the visibility of certains POIs. In order to prove the feasibility and expediency of our proposal, we conduct a series of experiments with real vehicle trajectories, in neighborhoods (the POIs) of Rio de Janeiro. Our results reveal information that can be useful for traffic analysis about the density of visits and proximities of vehicles to these neighborhoods.

Keywords: Trajectory, Moving object, episode, stop, move, proximity.

1 Introduction

Today, the collection of data about moving objects is a task that has been made easier thanks to technologies such as GPS [1], [2] and mobile devices [3], [4]. The analysis of such data can help to understand the behavior of almost any type of moving object, e.g., persons, animals, vehicles, among others.

In this paper, we propose the SPET (Stop and Proximity Episodes in Trajectories) algorithm to identify stop and proximity episodes in trajectories. A trajectory is the record of the evolution of the position of an object that is moving in space during a given time interval in order to achieve a goal [5]. On the other hand, according to the Merriam-Webster Dictionary, an episode is an event that is distinctive and separate although part of a larger series and according to Wordweb is a happening that is distinctive in a series of related events. The SMoT [6] algorithm enables the identification of stop and move episodes in the trajectory of an object. Informally, a stop is an episode of a trajectory during which the object remained continuously inside a point of interest (POI) a minimum time Δ (specified by the business analysts). On the other hand, a move is an episode during which an object was not inside any POI or if it was, it did not exceed the threshold Δ.

Unlike the SMoT algorithm, the SPET algorithm distinguishes between short and long stops, where a long stop is an episode during which the object remained continuously inside a POI a minimum time Δ and a short stop is an episode during which the object remained continuously inside a POI but did not exceed the threshold Δ.

Furthermore, the SPET algorithm identifies proximity episodes. Informally, a proximity is an episode of a trajectory during which the object remained continuously near a POI a minimum time Δ. Similarly to the stops, the proximities are also classified as short and long depending on their duration.

As for related works, we have identified the following. Cao [7] proposes collocation episodes, e.g., if a puma is moving near a deer, then a vulture is also going to move near to the deer with high probability within the next three minutes. In [8] the episodes are also classified in stops and moves. An episode is enriched with geographic data or with other interesting data for the

[*] Corresponding author e-mail: fjmoreno@unal.edu.co

application. For instance, information about the type of POI where a stop occurred may be included (e.g., if the POI is an office, a home, an entertainment site). In a subsequent work [9], a move is enriched with data about the way of displacement of the object (e.g., if the movement was done either by bus, subway, walking, etc.) In [10] the episodes are formed by a sequence of observations which have the same dynamic motion, i.e., they maintain the same speed or remain inside a specified speed range. In [11] the episodes correspond to segments of a trajectory, where each segment is defined according to the mean of transport used by the moving object (e.g., a tourist may move by bicycle, on foot, or by boat).

The rest of the paper is organized as follows. In Section 2, we present a motivating example. In Section 3, we present the SPET algorithm. Designed experiments are presented in Section 4 and in Section 5, we conclude the paper and discuss future works.

2 Motivating example

Consider a city where tourists can find a variety of POIs like hotels, shopping centers, restaurants, among others. Usually, tourists visit or pass near these POIs. Given the trajectory followed by a tourist in the city, we want to identify the POIs he visited or passed near to. The SMoT algorithm can identify the POIs visited by a trajectory: A POI is considered visited if the trajectory remained a minimum time inside it, e.g., thirty minutes in a restaurant. Therefore, the SMoT will not deem as visited the POIs where the trajectory entered but did not exceed such threshold, i.e., where it had a short stay. This algorithm will not identify either the places the trajectory passed near to, i.e., where it had proximities.

The identification of this type of episodes (short stops and proximities) may be useful to identify potential customers. For instance, if it is detected that a tourist entered a hotel H_1 (and he did not exceed the hotel time threshold) or passed near a hotel H_2 before visiting a hotel H_3 (and there he exceeded the hotel time threshold) the manager of hotel H_1 or hotel H_2 may try to determine the reasons why the tourist decided to enter hotel H_3 and not his. This information may help design strategies to attract more customers. Consider, e.g., Figure 1 where there are eight POIs for the application: three hotels (H_1, H_2, and H_3), two shopping centers (SC_1 and SC_2) and three restaurants (R_1, R_2, and R_3). The trajectory of a hypothetical tourist is also shown.

It is assumed that between each pair of temporally consecutive observations (points of the trajectory in the figure) fifteen minutes elapsed and that the first observation occurred at 10:00 am. For simplicity, we consider the same minimum time threshold in each hotel (four hours). According to the former, if the SMoT algorithm is applied, this will indicate that the tourist visited hotel H_1 since he remained there for more than four hours. However, the algorithm does not indicate that the tourist entered hotel H_2 (where he did not exceed the four hours threshold), or that he passed near to hotel H_3. If this information were available and the trajectories of different tourists were considered, we could identify, e.g., the number of tourists who entered a hotel but did not stay there (possibly they entered only to find out the hotel price and the amenities), we could also identify the number of tourists who passed near a hotel but did not enter there (maybe the hotel is not very visible from the spot the tourist went by and there are no signs indicating its presence).

Note that in the previous example it is necessary to calculate the distance from a point of the trajectory to a POI or determine whether a point is inside the POI. These calculations may be computationally expensive due to the irregular shape (geometry) that may represent the POI. However, in some situations just an estimate value could be sufficient. For this, the geometry representing a POI could be approximated by a buffer function, a MBR (minimum bounding rectangle), or a MBC (minimum bounding circle), among others [12]. For example, suppose we want to analyze the trajectory followed by an animal. If the animal visits the city or its vicinity during a time longer than a given threshold, it would be in imminent danger of being caught or be exposed to contaminants. Here, for practical reasons, the analysis of episodes could be developed by approximating the shape of the city by a MBR, see Figure 2. Thus, depending on the needs for analysis, we may identify the episodes with regard to the original geometry or an approximation of it.

3 Definitions and the proposed algorithm

Prior to the presentation of the SPET algorithm we introduce some preliminary concepts.

3.1 Trajectory

Let $\mathbb{R}$ be the set of real numbers. Let $\mathbb{R}^2 x \mathbb{R}$ be the space-time, where the first two dimensions represent the space (real plane) and the second one represents the time.

Definition 1. A trajectory of a moving object is a sequence of observations (points) $\langle (x_0, y_0, t_0), (x_1, y_1, t_1), \ldots, (x_n, y_n, t_n) \rangle$ in the space-time, where $x_i, y_i \in \mathbb{R}$, $t_i \in \mathbb{R}^+$, for $i = 0, \ldots, n$; and $t_0 < t_1 < \ldots < t_n$. x_i and y_i represent spatial coordinates and t_i represents a temporal coordinate (a time point). For simplicity, and to provide a finite representation, only rational coordinates are considered.

3.2 Trajectory episode

A trajectory may enter or pass near some POIs, e.g., hotels, shopping centers, cinemas, restaurants, parks,

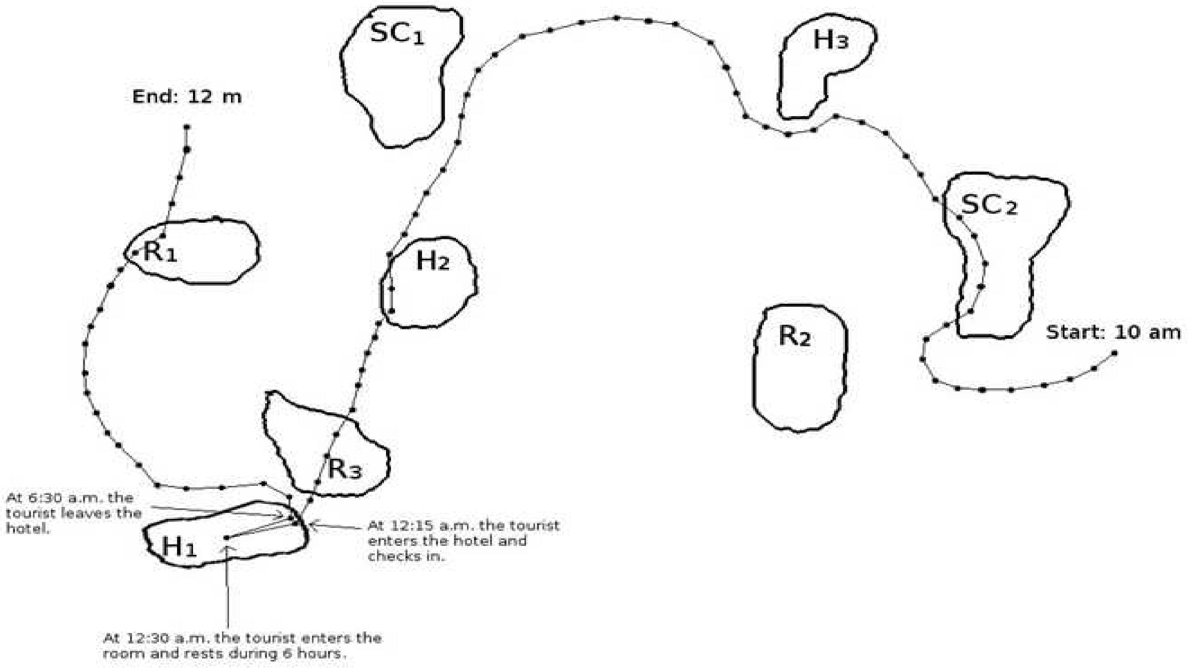
SC1
H3
End: 12 m
SC2
R1
H2
R2
Start: 10 am
R3
At 6:30 a.m. the tourist leaves the hotel.
At 12:15 a.m. the tourist enters the hotel and checks in.
H1
At 12:30 a.m. the tourist enters the room and rests during 6 hours.

Fig. 1: POIs and trajectory of a tourist.

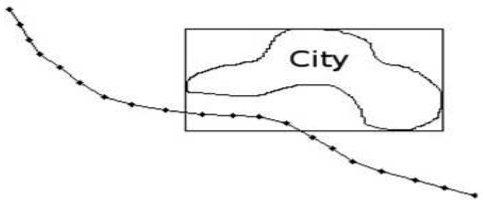

Fig. 2: Identification of episodes with regard to the MBR of the POI.

restrictive areas, among others. We consider that a trajectory episode occurs when a trajectory enters or passes near a POI. The trajectory episodes are classified into four types: short stop, long stop, short proximity, and long proximity, see Figure 3.

The short and long qualifiers are related to the duration of the trajectory episode. For instance, consider a cinema. Suppose we specify a minimum time that a person should stay in a cinema, e.g., one hour. Thus, if a person enters the cinema and exceeds this threshold (in practical terms this means that the person has seen a movie) then the episode is a long stop. On the other hand, if the person entered the cinema but did not exceed the threshold then the episode is a short stop.

Note that the value of the threshold that distinguish between short and long stops depends on the nature (class/role) of the POIs where the stop occurs. For a cinema one hour is a reasonable threshold, for a hotel it may be 6 or 8 hours, i.e., the average amount of time that a person sleeps per day. Therefore, classifying the POIs and their roles [8] is a key element to the business application in order to specify appropriate thresholds.

With regard to proximities episodes, we can specify a distance from which we consider that the object is near the POI and analogously to the stops, we can also specify a minimum time during which the object should remain near the POI to consider that there was a long or a short proximity.

3.2.1 Candidate stop

Definition 2. A candidate stop C is a tuple (R_C, Δ_{ST_C}), where R_C is a polygon (topologically closed) in $\mathbb{R}^2$ which represents the geometry of the POI and Δ_{ST_C} (staying time threshold) is a real number strictly positive that represents the minimum time period of continuous stay that an object must remain inside the

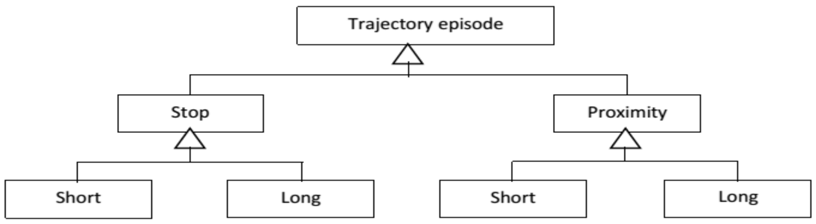

Fig. 3: Episode classification.

POI to consider that the object actually visited the POI.

3.2.2 Candidate proximity

Definition 3. A candidate proximity C is a tuple $(R_C, \Delta_{PD_C}, \Delta_{PT_C})$, where R_C is a polygon (topologically closed) in $\mathbb{R}^2$ that represents the geometry of the POI, Δ_{PD_C} (proximity distance threshold) and Δ_{PT_C} (proximity time threshold) are real numbers strictly positive. Δ_{PD_C} represents the distance from which it is considered that an object is near to R_C and Δ_{PT_C} represents the minimum time of continuous proximity that the object should remain near the POI to consider that the object was actually near the POI.

Definition 4. A candidate stop and proximity C is a tuple $(R_C, \Delta_{ST_C}, \Delta_{PD_C}, \Delta_{PT_C})$. This corresponds to the union of a candidate stop with a candidate proximity.

3.2.3 Application

An application $\mathscr{A}$ is a set $\{C_1 = (R_{C_1}, \Delta_{ST_{C_1}}, \Delta_{PD_{C_1}}, \Delta_{PT_{C_1}}), \ldots, C_n = (R_{C_n}, \Delta_{ST_{C_n}}, \Delta_{PD_{C_n}}, \Delta_{PT_{C_n}})\}$ of candidate stops and proximities; where $R_{C_j} \cap R_{C_k} = \varnothing, \forall 1 \leq j \leq n, 1 \leq k \leq n$, $j \neq k$. Informally, $\mathscr{A}$ represents a set of POIs each with its geometry and thresholds for identifying the stops and proximities of a trajectory.

3.2.4 Long and short stops, moves, long and short proximities.

Definition 5. Let T be a trajectory, $\mathscr{A}$ an application and a sub-trajectory $\langle (x_i, y_i, t_i), (x_{i+1}, y_{i+1}, t_{i+1}), \ldots, (x_{i+m}, y_{i+m}, t_{i+m}) \rangle$ of T. Let $C_k \in \mathscr{A}$ be such that $\forall j \in [i, i+m]$ the point (x_j, y_j) is inside R_{C_k} and the sub-trajectory be the maximum (with regard to this condition), then the tuple (R_{C_k}, t_i, t_{i+m}) is a *long stop* of T with regard to C_k if $|t_{i+m} - t_i| \geq \Delta_{ST_{C_k}}$ and it is, conversely, a *short stop*.

Definition 6. A move of T with regard to $\mathscr{A}$ corresponds to one of the following cases:
– The maximal contiguous subtrajectory of T between two temporally consecutive stops of T.
– The maximal contiguous subtrajectory of T between the first observation of T and the first stop of T.
– The maximal contiguous subtrajectory of T between the last stop of T and the last observation of T.
– T itself, if T has no stops.

Definition 7. Let T be a trajectory, $\mathscr{A}$ an application and a sub-trajectory $\langle (x_i, y_i, t_i), (x_{i+1}, y_{i+1}, t_{i+1}), \ldots, (x_{i+m}, y_{i+m}, t_{i+m}) \rangle$ of T. Let $C_k \in \mathscr{A}$ be such that $\forall j \in [i, i+m] : 0 < MinDistance((x_j, y_j), R_{C_k}) \leq \Delta_{PD_{C_k}}$ and the sub-trajectory be the maximum (with regard to this condition), then the tuple (R_{C_k}, t_i, t_{i+m}) is a *long proximity* of T with regard to C_k if $|t_{i+m} - t_i| \geqslant \Delta_{PT_{C_k}}$ and it is, conversely, a *short proximity*. The *MinDistance* function calculates the minimum distance between a point (x, y) and a polygon R. If the point is inside R, this function returns zero.

The SPET algorithm identifies the trajectory episodes of a set of trajectories with regard to an application $\mathscr{A}$, see Section 3.3.

3.3 SPET algorithm

In Stage 1, the Algorithm 1 identifies both short and long stops, and the trajectory moves. Then in Stage 2, the short and long proximities are identified, see Proximities function (Algorithm 2), as follows: the first observation of the proximity range is identified (line 4), then the other observations are overviewed (lines 7-9) until the last observation of the proximity range (line 10) is identified. Starting from the duration of the interval (line 12) it is determined if the episode is a short or a long proximity.

Note that the SPET algorithm identifies proximity episodes even if the object has entered the POI, as shown in Figure 4.

Our Proximities function can also identify all the proximities to all the POIs, even when a point of the

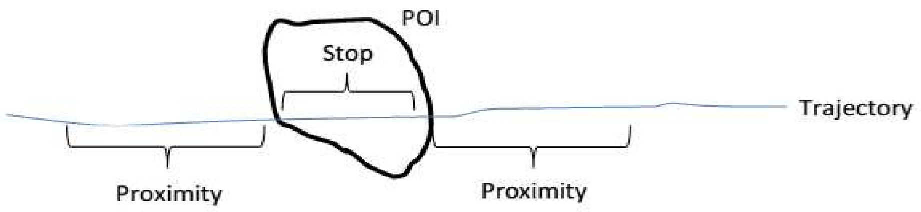

Fig. 4: A stop and two proximities.

Algorithm 1 SPET Algorithm // Short and Proximity Trajectory Episodes Algorithm

```
1: INPUT: $\mathscr{T}$ // Set of trajectories
2:         $\mathscr{A}$ /* Set of candidates $= \{C_1, C_2, \ldots, C_n\}$
               where $C_k = (R_{C_k}, \Delta_{ST_{C_k}}, \Delta_{PD_{C_k}}, \Delta_{PT_{C_k}})$*/
               approxGeo // Boolean variable to indicate if the geometry of the POIs must be approximated
3: OUTPUT: $\mathscr{M}$ // Set of moves
4:         $\mathscr{E}$ /* Set of episodes (Short Stops, Long Stops,
               Short Proximities, and Long Proximities)*/
5: BEGIN
6: if approxGeo then
7:     for each $R_{Ck}$ of a $C_k \in \mathscr{A}$ do
8:         Apply spatial approximation function to $R_{Ck}$ // Buffer, MBR, MBC, etc.
9:     end for
10: end if
11: // Stage 1: Find short and long stops, and moves
12: for each trajectory $T \in \mathscr{T}$ do
13:     Traverse $T$ from its first observation to its last one
        Find a maximal set of consecutive observations (MCO) of $T$ that intersects the $R_{Ck}$ of a $C_k \in \mathscr{A}$
        beginTime = Minimum time in MCO
        endTime = Maximum time in MCO
14:     if endTime – beginTime $\geq \Delta_{ST_{C_k}}$ then
15:         // A long stop of $T$ in $C_k$ was found
            $\mathscr{E} = \mathscr{E} \cup \{(T, R_{Ck}$, beginTime, endTime, ”Long Proximity” $)\}$
16:     else
17:         // A short stop of $T$ in $C_k$ was found
            $\mathscr{E} = \mathscr{E} \cup \{(T, R_{Ck}$, beginTime, endTime, ”Short Proximity” $)\}$
18:     end if
19:     end Traverse // Add moves
        Add to $\mathscr{M}$ all the sets of MCO of $T$ that do not belong to any short or long stop
20: end for
    // Stage 2: Find proximities, see Algorithm 2
    $\mathscr{E} = \mathscr{E} \cup$ Proximities $(\mathscr{A}, \mathscr{T})$; // Call Proximities function
    END
```

trajectory is near to several POIs at the same time, see Figure 5.

Note that when the geometry of a POI is not approximated to a regular shape, e.g., a rectangle (MBR)

Algorithm 2 Function Proximities

```
    FUNCTION Proximities $(\mathscr{A}, \mathscr{T})$
2:  for Each $C_k \in \mathscr{A}$ do
        Traverse $T$ from its first observation to its last one
4:      if $(0 <$ MinDistance(current observation of $T$, $R_{C_k}) \leq \Delta_{PD_{C_k}}$) then
            beginTime $=$ Time of the current observation of $T$
6:          Advance to the next observation of $T$
            while $(0 <$ MinDistance(current observation of $T$, $R_{C_k})$
            $\leq \Delta_{PD_{C_k}}$) do
8:              Advance to the next observation of $T$
            end while
10:         Go back to the previous observation of $T$
            endTime $=$ Time of the current observation of $T$
12:         if endTime – beginTime $\geq \Delta_{PT_{C_k}}$ then
                // A long proximity of $T$ in $C_k$ was found
                $\mathscr{E} = \mathscr{E} \cup \{(T, R_{C_k}$, beginTime, endTime, ”Long Proximity” $)\}$
14:         else
                // A short proximity of $T$ in $C_k$ was found
                $\mathscr{E} = \mathscr{E} \cup \{(T, R_{C_k}$, beginTime, endTime, ”Short Proximity” $)\}$
16:         end if
        end if
18:     end Traverse
    end for
20: return $\mathscr{E}$
```

or a circle (MBC), the process corresponding from lines 7 - 9 of the Proximities function may be computationally expensive because it involves calculating the distance of each observation trajectory to the geometry (possibly irregular) of each POI. This process is optimized by the OptimizedProximities function (Algorithm 3) which tries to avoid some of these calculations. The essential idea is this: let i be the observation of a trajectory and let d be the minimum distance between that observation and the R_{C_k} of a POI:

– If $d > \Delta_{PD_{C_k}}$ then the observation i does not meet the proximity distance threshold $(\Delta_{PD_{C_k}})$ with regard to R_{C_k}, we say that i is a *reference point*. Let be $diff = d - \Delta_{PD_{C_k}}$ and be d' the distance between observation i and observation $i+1$. If $d' < diff$ then observation $i+1$ neither meets the threshold $\Delta_{PD_{C_k}}$;

therefore, it is not necessary to calculate the distance between observation $i+1$ and R_{C_k}. This process is repeated until we find an observation j with $j > i$ such that the distance between i and j is greater than $diff$. If this happens, the distance between j and R_{C_k} is calculated and we analyze again if the threshold $\Delta_{PD_{C_k}}$ is met. If it is not met then j will be the new reference point.

– If $d \le \Delta_{PD_{C_k}}$, then the observation i does meet the proximity distance threshold $(\Delta_{PD_{C_k}})$ with regard to R_{C_k}, we say that i is a *reference point*. Let be $diff = \left|d - \Delta_{PD_{C_k}}\right|$ and be d' the distance between observation i and observation $i+1$. If $d' \leqslant diff$ then observation $i+1$ *also* meets the threshold $\Delta_{PD_{C_k}}$; therefore, it is not necessary to calculate the distance between observation $i+1$ and R_{C_k}. This process is repeated until we find an observation j with $j > i$ such that the distance between i and j is greater than $diff$. If this happens, the distance between j and R_{C_k} is calculated and we analyze again if the threshold $\Delta_{PD_{C_k}}$ is met. If it is met then j will be the new reference point.

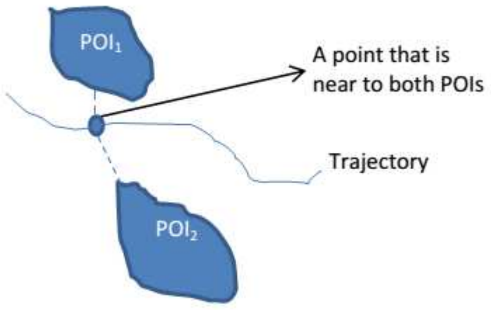

Fig. 5: A point near to two POIs.

Next, we illustrate the functioning of the Optimized Proximities function. Consider Figure 6. It shows a trajectory with 32 observations. It is assumed that between each pair of temporally consecutive observations one minute has elapsed. The figure shows the distance d $(43m)$ from observation 1 to POI C_k and it demarcates the circular zone coverage corresponding to the difference $diff = d - \Delta_{PD_{C_k}}$. Note that any observation of the trajectory that is in this region will not meet the $\Delta_{PD_{C_k}}$.

As the distance d $(43m)$ is greater than $\Delta_{PD_{C_k}}(10m)$ then observation 1 is a reference point. Then we calculate the difference $diff = 43m - 10m = 33m$ and the distance d' between observation 1 and subsequent observations, until we find an observation where the distance is greater than or equal to $33m$ (in Figure 6 the observation 8). We

Algorithm 3 Function OptimizedProximities

```
    FUNCTION OptimizedProximities ($\mathscr{A}$, $\mathscr{T}$)
    for each $C_k \in \mathscr{A}$ do
3:      $i = 0$
        while $i \leqslant size(T)$ do
            // Traverse $T$ from its first observation to its last one
6:          $d$ = MinDistance($T[i]$,$R_{C_k}$)
            reference_point = $i$ // Mark current observation of $T$ as a reference point
            $diff = d - \Delta_{PD_{C_k}}$
9:          if $diff > d$ then
                $diff = d$
            end if
12:         storeFlag = FALSE // Variable to indicate when a proximity is identified
            if $0 < d \leqslant \Delta_{PD_{C_k}}$ then
                // The current observation of $T$ meets $\Delta_{PD_{C_k}}$
15:             beginTime = Time of $T[i]$
                $i{+}{+}$
                while Distance($T$[reference_point],$T[i]$) $\le diff$ do
18:                 $i{+}{+}$
                end while
                if MinDistance($T[i]$,$R_{C_k}$) = 0 then
21:                 storeFlag = TRUE // Current observation of $T$ is inside $R_{C_k}$
                end if
                $i{-}{-}$
24:             endTime = Time of $T[i]$
            else
                if $T[i–1]$ AND MinDistance($T[i–1]$,$R_{C_k}$) $\le \Delta_{PD_{C_k}}$ then
27:                 // We have identified a proximity
                    storeFlag = TRUE
                end if
30:             $i{+}{+}$
                // The current observation of $T$ does not meet $\Delta_{PD_{C_k}}$
                while Distance($T$[reference_point],$T[i] \le diff$) do
33:                 $i{+}{+}$
                end while
                $i{-}{-}$
36:         end if
            if storeFlag = TRUE OR $T[i] \le$ size($T$) AND MinDistance($T[i–1]$,$R_{C_k}$) $\le \Delta_{PD_{C_k}}$ then
                if endTime – beginTime $\ge \Delta_{PT_{C_k}}$ then
39:                 // A long proximity of $T$ in $C_k$ was found
                    $\mathscr{E} = \mathscr{E} \cup \{(T, R_{C_k}$, beginTime, endTime, ”Long Proximity” $)\}$
                else
                    // A short proximity of $T$ in $C_k$ was found
                    $\mathscr{E} = \mathscr{E} \cup \{(T, R_{C_k}$, beginTime, endTime, ”Short Proximity” $)\}$
42:             end if
                storeFlag = FALSE
                $i{+}{+}$
45:         end if
        end while
    end for
48: return $\mathscr{E}$
```

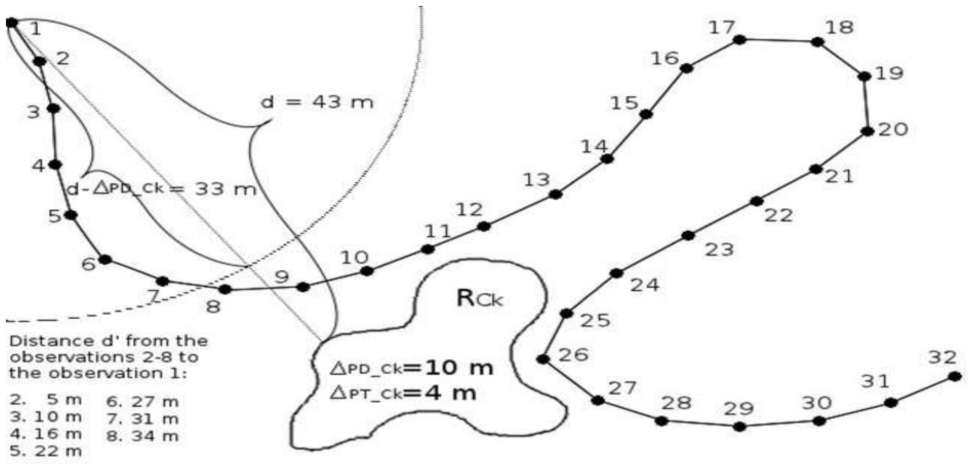

Fig. 6: Illustrating the OptimizedProximities function (Obs. 2-8).

Table 1: Application table $\mathscr{A}$.

POIId	POIName	POIType	Δ_{ST}(min)	Δ_{PD}(m)	Δ_{PT}(min)
1	San Diego	Shopping center	30	20	20
2	Media Naranja	Shopping center	30	20	20
3	Babylon	Nightclub	40	10	15
4	Plaza Mayor	Park	30	20	20
5	Parque Berrio	Park	30	20	20
6	Parque Explora	Park	30	20	20
7	Florida	Hotel	60	10	15
8	Alejandra	Hotel	60	10	15
9	Nutibara	Hotel	60	10	15
10	Alameda	Restaurant	30	10	15
11	Mar del plata	Restaurant	30	10	15
12	El Carboncito	Restaurant	30	10	15

calculate the distance d between observation 8 and C_k, see Figure 7, and we find that $d = 10m = \Delta_{PD_{C_k}}$. The observations from 9 to 13 also meet the threshold $\Delta_{PD_{C_k}}$. Since between observation 8 and observation 13 five minutes elapsed, a set of observations has been detected from the trajectory that meets the conditions for being a long proximity (since $\Delta_{PT_{C_k}} = 4min$). This process is repeated in an analogous way from observation 10 of the trajectory (new reference point).

Example. Table 1 shows a sample of data from an application $\mathscr{A}$ and Table 2 shows an example of trajectory episodes.

Next we present some queries that can be formulated from these tables.

1. Find the code of the POIs where trajectories had stops and tell if the stop was short or long.

SELECT POIId, TrajectoryId, EpisodeType
FROM Episode
WHERE EpisodeType LIKE ’%Stop’;

2. Which stops (long or short) had the trajectories that passed near the POI 3?

SELECT TrajectoryId, EpisodeType, POIId
FROM Episode
WHERE EpisodeType LIKE ’%Stop’ **AND** TrajectoryId IN
(**SELECT** TrajectoryId
FROM episode
WHERE EpisodeType LIKE ’%Proximity’ **AND** POIId = 3);

3. Which trajectories passed near or entered the POIs 2, 3, and 4?

SELECT TrajectoryId, EpisodeType, POIId
FROM Episode
WHERE POIId IN (2, 3, 4);

4. Which trajectories had long stops in hotels and passed near (long o short) or had short stops in hotels?

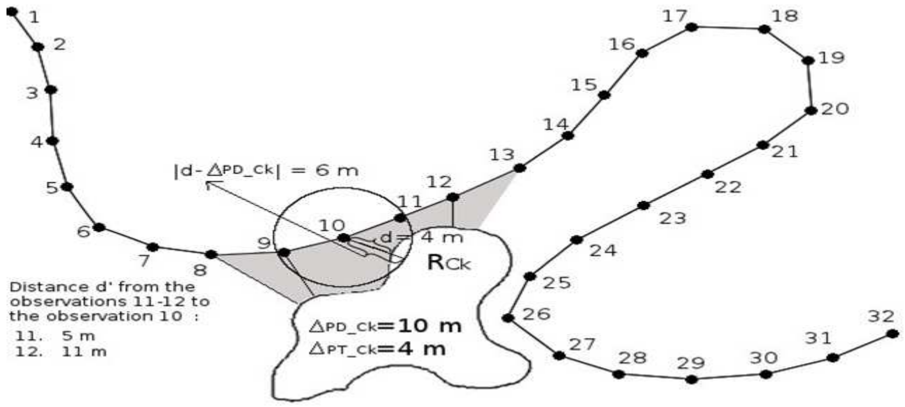
|d-ΔPD_Ck| = 6 m
d = 4 m
RCk
Distance d' from the observations 11-12 to the observation 10 :
11. 5 m
12. 11 m
ΔPD_Ck=10 m
ΔPT_Ck=4 m

Fig. 7: Illustrating the OptimizedProximities function (Obs. 8-13).

Table 2: Episode table.

EpisodeId	TrajectoryId	POIId	BeginTime (01/12/2012)	EndTime (01/12/2012)	EpisodeType
1	1	1	10:00	10:30	Long Stop
2	1	7	11:00	12:00	Long Stop
3	1	12	12:30	13:30	Long Stop
4	1	8	14:00	14:05	Short Stop
5	1	5	14:30	16:30	Long Stop
6	1	3	17:00	18:00	Long Proximity
7	1	11	18:30	18:40	Short Stop
8	1	2	19:30	19:35	Short Proximity
9	1	9	19:55	20:30	Long Proximity
10	2	4	08:55	10:30	Long Stop
11	2	8	10:55	11:00	Short Stop
12	2	12	12:00	13:00	Long Stop
13	2	10	13:20	13:30	Short Stop
14	2	1	13:45	13:55	Short Stop
15	2	6	14:15	14:20	Short Proximity
16	2	9	14:45	17:00	Long Stop
17	2	2	17:05	17:30	Long Proximity
18	2	11	19:50	20:30	Long Stop
19	2	5	21:00	21:05	Short Proximity

```
SELECT e1.TrajectoryId, a1.POIId, a1.POIName
FROM Episode AS e1, Application AS a1
WHERE e1.EpisodeType = 'Long Stop' AND
e1.POIId = a1.POIId AND a1.POIType = 'Hotel'
AND e1.TrajectoryId IN
        (SELECT e2.TrajectoryId
        FROM Episode AS e2, Application AS a2
        WHERE e2.EpisodeType <> 'Long Stop'
        AND e2.POIId = a2.POIId
        AND a2.POIType = 'Hotel');
```

5. During a time interval (e.g., 12:00 and 14:00, at lunchtime) on a specific day, which restaurants had more proximities (short or long) and short stops that long stops?

```
SELECT a.POIId, a.POIName
FROM Episode AS e, Application AS a
WHERE e.POIId = a.POIId AND a.POIType =
'Restaurant' AND e.BeginTime BETWEEN
TIMESTAMP '2012-10-06 12:00:00' AND
TIMESTAMP '2012-10-06 14:00:00'
GROUP BY e.POIId
HAVING COUNT(CASE WHEN e.EpisodeType
        <> 'Long Stop' THEN 1
        ELSE NULL END) >=
        COUNT(CASE WHEN e.EpisodeType =
        'Long Stop' THEN 1
        ELSE NULL END);
```

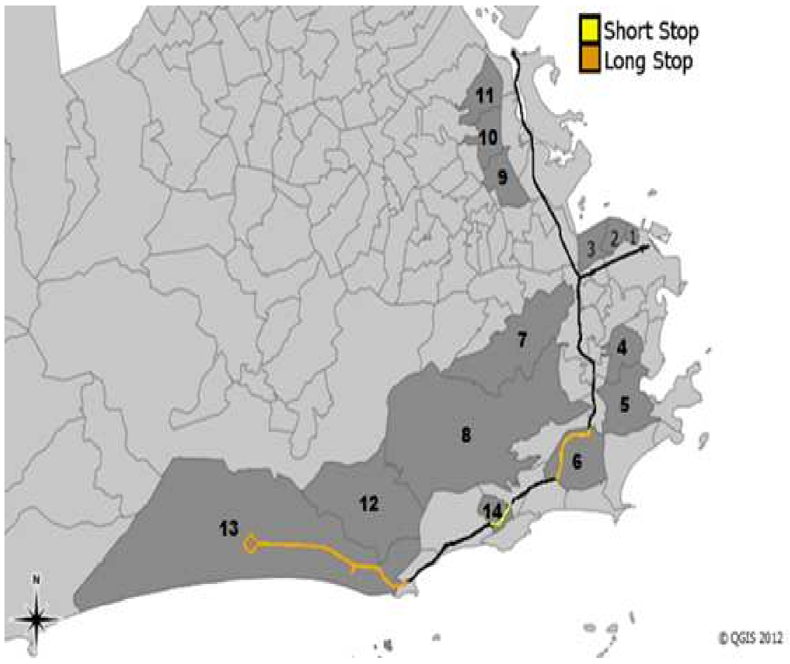

Fig. 8: Stops of a trajectory.

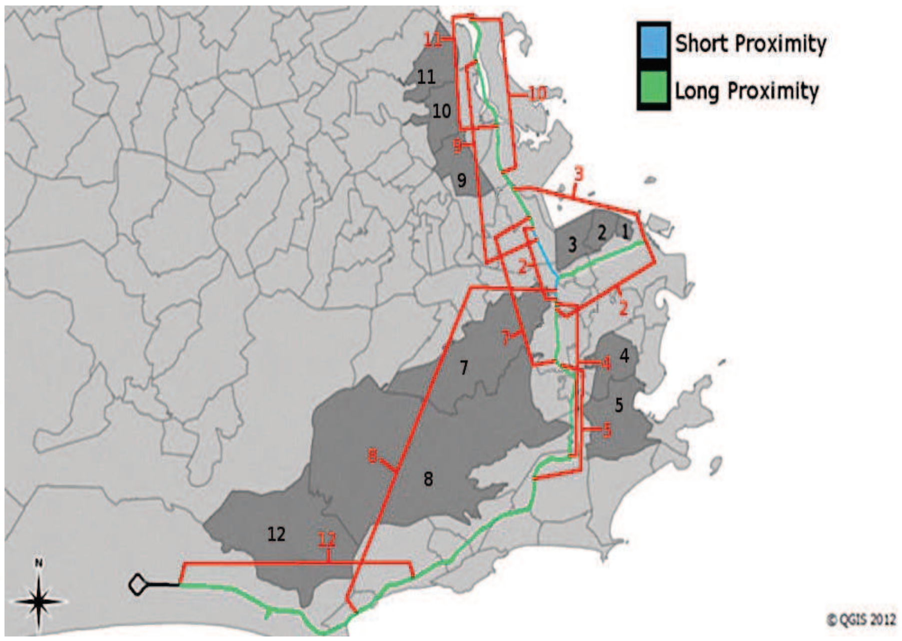

Fig. 9: Proximities of a trajectory.

4 EXPERIMENTS

Experiments were performed with 100 real vehicle trajectories, in 16 neighborhoods (the POIs) of Rio de Janeiro, see Table 3. The data were provided by Companhia de Engenharia de Tráfego do Rio de Janeiro (CET-Rio). The total number of observations on the 100 trajectories was 268900. We considered in all neighborhoods: $\Delta_{ST} = 10min$, $\Delta_{PD} = 2km$, and $\Delta_{PT} = 2min$. The SPET algorithm was implemented in PostgreSQL 9.1, a DBMS that offers a rich set of spatial features. Experiments were carried out on a laptop with a

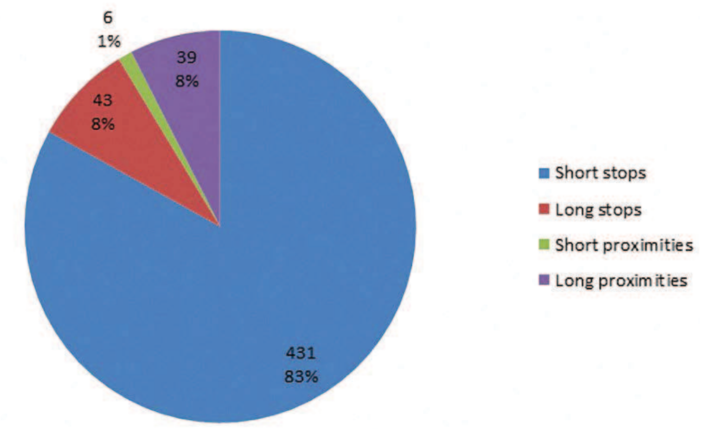

Fig. 10: Episodes identified.

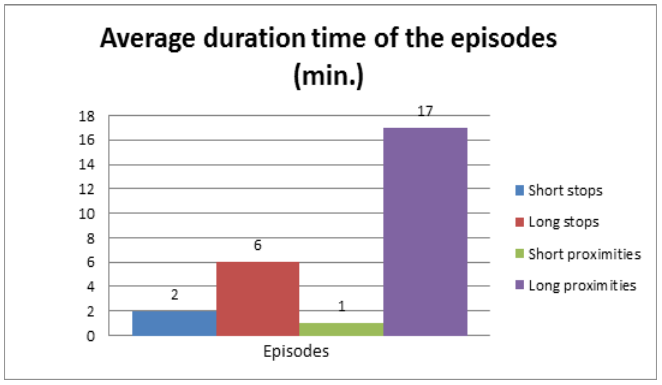

Fig. 11: Average duration time of the episodes.

2.13 GHz processor, 4 Gb RAM, and in a Linux Ubuntu 11.10 operating system. Figure 8 shows stop episodes generated by a specific trajectory. The numbers in the figure correspond to the neighborhoods in Table 3. We can see that this trajectory had two long stops (one in the neighborhood 6 and one in the neighborhood 13) and one short stop (in the neighborhood 14).

On the other hand, Figure 9 shows proximity episodes generated by the same trajectory. Each bracket represents a proximity episode. The number in red, attached to each bracket, indicates the neighborhood associated with the proximity episode. For example, we can see that this trajectory had one short proximity and one long proximity with regard to the neighborhood 2.

Table 4 shows a sample of the results and Table 5 shows consolidated results. The results were obtained in less than two minutes when the OptimizedProximities function was used and in less than four minutes when the non-optimized Proximities function was used. Considering the volume of data and the laptop specifications, these times are acceptable for interactive use. Figure 10 shows the total number of episodes identified for each type and in Figure 11 the average duration time of the episodes.

Table 3: Neighborhoods.

POIId	POIName
1	Sade
2	Gamboa
3	Santo Cristo
4	Laranjeiras
5	Botafogo
6	Lagoa
7	Tijuca
8	Alto da Boa Vista
9	Manguinhos
10	Bonsucesso
11	Ramos
12	Itanhang
13	Barra de Tijuca
14	Rocinha

Table 4: Episodes: sample of the results.

EpisodyType	TrajectoryId	POIId	BeginTime	EndTime
Short Stop	17	14	30/12/2004 12:24:19	30/12/2004 12:24:28
Short Stop	17	13	30/12/2004 12:28:50	30/12/2004 12:50:59
Short Proximity	18	4	27/12/2004 17:13:14	27/12/2004 17:13:14
Long Proximity	18	8	27/12/2004 17:12:14	27/12/2004 17:28:20
Long Stop	20	13	25/04/2005 08:20:59	25/04/2005 09:21:15
Long Proximity	20	12	25/04/2005 08:32:23	25/04/2005 09:25:41
Short Proximity	74	4	15/07/2005 11:17:14	15/07/2005 11:18:38

Table 5: Experiments: consolidated results.

Item	Value
Total number of episodes identified	519
Neighborhood with more short stops	Bonsucesso (131)
Neighborhood with more long stops	Santo Cristo (33)
Neighborhood with more short proximities	Alto da Boa Vista (4)
Neighborhood with more long proximities	Laranjeiras, Alto da Boa Vista (14)
Neighborhood with less short stops	Botafogo (1)
Neighborhood with less long stops	Botafogo, Tijuca, Barra da Tijuca (1)
Neighborhood with less short proximities	Laranjeiras (2)
Neighborhood with less long proximities	Itanhang (11)
Total number of neighborhoods without episodes	0
Total number of neighborhoods with short stops	11
Total number of neighborhoods with long stops	7
Total number of neighborhoods with short proximities	2
Total number of neighborhoods with long proximities	3

5 Conclusions and future work

In this paper, we proposed the SPET algorithm to enable the identification of trajectory episodes such as short and long stops, moves, and short and long proximities. These episodes may help to understand the behavior of moving objects in several domains. For example, proximities episodes can help in advertising, where agents can identify appropriate spots in order to try to increase the visibility of certains POIs. On the other hand, short stops episodes, another contribution of our paper, can help to identify potential customers. With regard to the work of Furtado [13], our work can be considered as an alternative and complementary way to find proximity episodes. His work is based on defining buffers around places in order to identify ”passBy episodes” whereas we consider a distance threshold to the places.

As future work we consider the following. i) to enrich the episodes with more information, e.g., weather conditions, customer satisfaction (e.g., how much enjoyable for a customer was his long stop in a hotel or a cinema), resources use (how much spent a tourist in his stops at a hotel? How much fuel a vehicle spent on its moves?), ii) to identify groups of typical episodes in a trajectory or in a set of trajectories, i.e., collective episodes, iii) to forecast the next episodes (and their expected duration) of an object that is moving in real time, considering its previous trajectories or the trajectories of similar moving objects, iv) to develop a specialized language to query episodes, and v) to identify other types of episodes, e.g., correlated episodes, i.e., episodes that usually involve the existence of other episodes, e.g., usually a tourist after a long trip (a move) has a stop at a hotel.

References

[1] M. Kennedy, The Global Positioning System and ArcGIS. CRC Press, (2009).

[2] P. Bolstad, GIS fundamentals. White Bear Lake, Mn: Eider Press, 395-463 (2005).

[3] K. D. Rogers, Mobile learning devices. Essentials for principals. Solution Tree, (2011).

[4] M. Saylor, The mobile wave: how mobile intelligence will change everything. Vanguard, (2012).

[5] S. Spaccapietra, C. Parent, M. L. Damiani, J. A. de Macedo, F. Porto, and C. Vangenot, A conceptual view on trajectories. Data & knowledge engineering, **65**, 126-146, (2008).

[6] L. O. Alvares, V. Bogorny, B. Kuijpers, J. A. de Macedo, B. Moelans, and A. Vaisman, A model for enriching trajectories with semantic geographical information. In Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems (p. 22). ACM, (November 2007).

[7] H. Cao, N. Mamoulis, and D. W. Cheung, Discovery of collocation episodes in spatiotemporal data. In Data Mining, 2006. ICDM'06. Sixth International Conference on (pp. 823-827), IEEE, (December 2006).
[8] Z. Yan, C. Parent, S. Spaccapietra, and D. Chakraborty, A hybrid model and computing platform for spatio-semantic trajectories. In The Semantic Web: Research and Applications (pp. 60-75), Springer Berlin Heidelberg, (2010).
[9] Z. Yan, N. Giatrakos, V. Katsikaros, N. Pelekis, and Y. Theodoridis, SeTraStream: semantic-aware trajectory construction over streaming movement data. In Advances in Spatial and Temporal Databases (pp. 367-385), Springer, (2011).
[10] S. Campora, J. A. de Macedo, and L. Spinsanti, St-Toolkit: A framework for trajectory data warehousing. AGILE, (2011).
[11] C. Parent, S. Spaccapietra, C. Renso, G. Andrienko, N. Andrienko, V. Bogorny, and Z. Yan, Semantic trajectories modeling and analysis.ACM Computing Surveys (CSUR), **45**, 42 (2013).
[12] K. Kemp, (Ed.), Encyclopedia of geographic information science. SAGE Publications, (2008).
[13] A. S. Furtado, R. Fileto, and C. Renso, M-Attract: assessing the attractiveness of places by using moving objects trajectories data. GeoInfo, MCT/INPE, (2012).

Francisco Moreno holds a PhD in Engineering Systems from Universidad Nacional de Colombia. He currently works an associate professor at Universidad Nacional de Colombia, sede Medelln. His research areas are datawarehouses and spatio-temporal databases.

Anderson Castaño holds a degree in Engineering Systems from Universidad Nacional de Colombia. He currently works as an independant consultant. His research areas are datawarehouses and spatio-temporal databases.

Francisco Javier de Cos Juez holds a PhD in Mining Exploitation and Prospecting from Universidad de Oviedo, Spain. He currently works as an associate profesor at Universidad de Oviedo. His research areas are engineering projects and mining analysis.