American Journal of Applied Sciences

Original Research Paper

Low-Sampling Trajectory Reconstruction using Criteria-Based Routing over a Graph

Edison Ospina, Francisco Moreno and Jaime Guzmán

Department of Computer Science and Decisions Making, National University of Colombia, Medellín, Colombia

Article history
Received: 05-03-2015
Reviewed: 13-08-2015
Approved: 02-02-2016

Corresponding Author:
Francisco Moreno,
National Department of Computer Science and Decisions Making, National University of Colombia,
Medellín, Colombia
E-mail: fjmoreno@unal.edu.co

Abstract: Location-based services mainly provide geo-location data. However, a moving object's detailed trajectory route is lost when there is low-sampling of these location data. Previous works have been developed in order to find the possible trajectories by using the location history logged by users. These methods can be considered as reconstruction or imputation processes. In this study, we reconstruct trajectories using personalization features of the routing theory based on evaluation criteria over a graph. In addition, this trajectory reconstruction has only been considered in a confined environment, i.e., a road network.

Keywords: Trajectory Reconstruction, Personalized Routing, Graph Theory, Imputation Process

Introduction

The fast development of technologies and mobile applications has arisen the need of analyzing the huge amount of geo-location data recorded regarding Moving Objects (MO). For example, users in mobile social networks such as Foursquare and Flickr use checking-in and sharing geo-tagged photos features to indicate their location.

However, usually it is difficult or impossible to get detailed data about the movement of a user due to privacy issues (Chow and Mokbel, 2011), energy saving or simply because people do not check-in in every place they visit. As a consequence, source (raw) trajectory data are not very accurate since there are missing data during the *silent durations*, i.e., the time durations of a trajectory when no data are available to describe the movement of an object (Hung *et al*., 2011). Thus, the trajectory between two consecutive data records is unknown. As a result, the following are some possible questions to be addressed: How does an object move during a silent duration? How well do the current methods describe the actual trajectory followed by a MO? Does an object move according to a certain criterion, e.g., trying to avoid traffic jams or slopes?

Previous works have focused on historical trajectory datasets of the same MO (Chang *et al*., 2011) or of similar MOs (Chen *et al*., 2011) as a way of inferring the routes or the movement patterns of a MO. For trajectory reconstruction (i.e., the imputation process for silent durations) some authors (Liu *et al*., 2011; Wei *et al*., 2012) use an uncertainty reinforcement approach (i.e., uncertain + uncertain → certain). However, these approaches may be inadequate if the silent durations in the trajectories followed by the same MO are long (they exceed an application threshold) and recurrent (i.e., there are recurrent trajectory segments where no trajectory data are available).

The problem of finding a route from one place to another, i.e., in the Route Finding Problem (RFP) (Da Silva *et al*., 2008; Schultes, 2008) is akin to the one of finding a trajectory between two consecutive low-sampled points. In recent years, several approaches (Hochmair, 2005; Da Silva *et al*., 2008; Schultes, 2008) have incorporated metrics other than distance (e.g., time) and user criteria (e.g., preference for the path with most touristic attractions) to the RFP in order to provide customized solutions.

Hochmair (2005) offers a brief taxonomy to build the "best" route based on criteria such as speed, safeness, attractiveness and simplicity for traversing a Road Network (RN). This same need is addressed by the route planning theory, i.e., the integration of user criteria to get "better routes" (Hochmair, 2005). A novel and relevant task is the reconstruction of low-sampling trajectories based on the movement patterns and the geographical space where it occurs, e.g., the RN of a city (i.e., the possible locations of the MO are constrained by the geometry of the RN (De Almeida and Güting, 2005; Trajcevski, 2011)).

Table 1. Research works including personalization and road network

Reference	Include criteria/ personalization	Consider RN	Perform trajectory reconstruction	Description
Orlando *et al.* (2007)	No	No	Yes	A simple interpolation is done for reconstructing trajectories as a previous step in a TDW proposal.
Marketos and Theodoridis (2009)	No	No	Yes	The trajectory reconstruction is included in a module of a TDW using parameters such as temporal and spatial gap between trajectories, maximum speed and tolerance distance.
Yuan *et al.* (2010; Chen *et al.*, 2011; Zheng *et al.*, 2012)	No	Yes	No	Uses a Pattern-based approach for the offline preprocessing of historical trajectory data for discovering mining patterns to infer routing information. However a route is inferred, not a trajectory.
Chen *et al.* (2010)	No	Yes	Yes	The K best connected trajectories are given when a set of locations (queried points) is the input.
Chang *et al.* (2011)	Yes	Yes	No	A familiar RN followed by a specific user is built using historical data. Routes are inferred from this familiar RN for the user. Chang *et al.* (2011) infers routes, not trajectories.
Hsieh and Li (2013)	Yes	Yes	No	Uses Greedy search approaches, i.e., optimal local choices at every decision stage providing an online recommendation based on the best immediate location to be visited for constructing the route. However a route is inferred, not a trajectory.

The route among check-in data of a low sampling trajectory is built (filled in) with additional geo-referenced data points and time-stamps. To help in this task, a graph that represents the RN is built where the vertices save geo-related information (longitude and latitude) and the edges describe the cost for reaching two vertices (Speičvcys *et al.*, 2003). The routing algorithms rely on this representation to build the trajectory between two location points (Dijkstra, 1959; Hart *et al.*, 1968).

Low-sampling data uncertainty management is a hard task to tackle. To facilitate this task, the trajectory reconstruction can rely on user preferences (a criterion), such as (minimize) distance or (visit) tourist attractions to try to fill in those silent durations. To the best of our knowledge, user preferences have not been considered in the low-sampling trajectory reconstruction problem. Our claim is that the movement of an object based on user preferences would generate some clues which may help in the trajectory reconstruction (Hung *et al.*, 2011). Moreover, this may help to analyze the movement from different perspectives, i.e., depending on the criterion used for the reconstruction process.

In order to clarify the contribution of our paper, in Table 1 we refer to some research works based on route finding (or reconstruction of trajectories) using personalization and RN. Chang *et al.* (2011) and Hsieh and Li (2013) are the only ones who find the routes based on the RN and the user preferences. However, they only infer the route and the trajectory is not reconstructed.

Representation of Trajectories

Several models for representing trajectories have been proposed in the literature (Orlando *et al.*, 2007; Spaccapietra *et al.*, 2008; Chang *et al.*, 2011). Most of them, except Spaccapietra *et al.* (2008), represent a trajectory as a sequence of geo-referenced points temporally ordered.

According to Orlando *et al.* (2007), a trajectory T_i = (ID_i, L_i) where ID_i is the unique identification of the MO and L_i is a sequence of M observations $=< L_i^1, L_i^2, ..., L_i^M >$. Each observation $L_i^j = \left(x_i^j, y_i^j, t_i^j\right)$ represents the MO at location $\left(x_i^j, y_i^j\right)$ where x_i^j , y_i^j $\in \mathbb{R}$ and at time $t_i^j \in \mathbb{T}$, where $\mathbb{T}$ is a set of time-points. $L_i \in 2^L$, where L is the set of all possible observations. L_i is temporally ordered, i.e., $t_i^j < t_i^{j+1}$, $\forall$ $1 \leq j < M$. TS = $\{T_i\}$ is a set of trajectories (possibly low-sampled).

Reconstruction of Low-Sampled Trajectories

Given a trajectory T_i of a MO where some pairs of observations j and j+1, $1 \leq j < M$, may be separated spatially and temporally in such a way that they exceed a spatial user threshold β and a temporary user threshold τ, i.e., they are considered as low-sampled, our goal is to fill in each of these pairs with imputed observations so that β and τ thresholds are met. Our reconstruction process is based on a set of criteria *Cset* from the personalized route planning theory (Hochmair, 2005; Da Silva *et al.*, 2008; Nadi and Delavar, 2011), such as time and distance.

We consider the network-constrained trajectories (*TS*, *Ga*), where *TS* is a set of trajectories and *Ga* is a *directed* and *labeled* graph representing the underlying constrained RN where the set of trajectories *TS* is constrained. The graph *Ga* is a two-tuple. *Ga* = (*V*, *E*), where *V* is a set of vertices $\{v_i\}$ (representing the intersections of the streets) and *E* is a set of edges $\{e_k\}$

(representing the segments of the streets). An edge e_k has a *source vertex* (the initial part of an edge) denoted by $v_{k,s}$, a *target vertex* (the end part of an edge), which is denoted by $v_{k,t}$ (the edge e_k is traversed from $v_{k,s}$ to $v_{k,t}$, but not the other way around) and an associated cost for traversing it denoted by $c_k \in \mathbb{R}$, i.e., an edge is a tuple $e_k = (v_{k,s}, v_{k,t}, c_k)$. Each vertex $v \in V$ can be described by a location *x, y* (longitude, latitude). We consider the graph *Ga*, which is derived from a RN, to be fully connected and without any isolated network segments. We consider the following functions, see also Fig. 1:

- *get_vertex_source*: $E \rightarrow V$. Function applied to an edge to get its source vertex
- *get_vertex_target*: $E \rightarrow V$. Function applied to an edge to get its target vertex
- *get_cost:* $E \rightarrow \mathbb{R}$. Function applied to an edge to get the cost of traversing the edge
- *get_x:* $V \rightarrow X$. Function applied to a vertex to get its longitude
- *get_y:* $V \rightarrow Y$. Function applied to a vertex to get its latitude

The function *road_distance:* L X L X *Cset* $\rightarrow \mathbb{R}$ receives a pair of consecutive observations and a criterion of movement *c* and generates the road distance between them according to the given criterion. The road distance refers to the distance of a particular path followed by a MO between the two observations. It depends on the underlying RN and on the criterion *c*. Figure 2 shows three possible roads (depicted in solid lines) between consecutive observations *A* and *B* according to some criteria. The criterion c_1 (distance) used in the road drawn in green has the shortest road distance, followed by the road distance of the road drawn in blue using the criterion c_2 (time). Finally, the road distance is the longest when the criterion c_3 (tourist attraction) is used, i.e., *road_distance*(A, B, c_1) $\leq$ *road_distance*(A, B, c_2) $\leq$ *road_distance*(A, B, c_3). Note how the distance between these observations changes according to the criterion of movement and the RN that were used. Note also that the Euclidean distance, depicted as a dashed line, does not correspond to the road distance in any of the three cases.

We regard the trajectory T_i as low-sampled if $\exists_j$, $1 \leq j \leq M$, (*road_distance* $(L_i^j, L_i^{j+1}, c) \geq \beta \wedge t_i^{j+1} - t_i^j \geq \tau$), i.e., the road distance according to a criterion *c* and a RN between two consecutive observations is longer than β (a user distance threshold) and their time difference $\left(t_i^{j+1} - t_i^j\right)$ is longer than τ (a user time threshold). We consider the function *traj*(L_i,*c*) where $L_i \in 2^L$ is the sequence of *M* observations of a trajectory T_i and $c \in$ *Cset*, is a criterion of reconstruction. The result of the *traj* function is a more detailed sequence of observations L'_i so that the thresholds β and τ are met for L_i^j and L_i^{j+1}, $\forall j$, $1 \leq j < M$.

The idea behind the trajectory reconstruction function is to fill in the trajectory with imputed observations between observations L_i^j and L_i^{j+1} ($\forall j$, $1 \leq j < M$, where both thresholds β and τ are not met) considering the criterion *c* and the RN. Next, we explain the effect of the *traj* function over a pair of observations L_i^j and L_i^{j+1} where the thresholds β and τ are not met in order to show how the sequence of low-sampling data are imputed.

As presented by Zhixian (2011) for the correct (cleaned) network-constrained trajectory datasets, given any of its spatio-temporal observations $\left(x_i^j, y_i^j, t_i^j\right)$, its location $\left(x_i^j, y_i^j\right)$ should be over a road edge $\in$E (set of edges of *Ga*). Consider two sampled observations L_i^j and L_i^{j+1} where the β and τ thresholds are not met. Each observation is associated with the nearest edge in a road map (represented by the graph *Ga*) using the *get_edge* function, i.e., *get_edge*(L_i^j,*Ga*) and *get_edge*(L_i^{j+1},*Ga*). The signature of the *get_edge* function is L X $G \rightarrow E$, where G is the set of the directed and labeled graphs representing RNs. Here, the nearest edge in the graph $Ga = (V, E)$ is the output of the *get_edge* function. Therefore, a point (x_i^j, y_i^j, t_i^j) that *is not over* an edge $\in E$ is replaced by a point ($x_i'^j$, $y_i'^j$, t_i^j) where ($x_i'^j$, $y_i'^j$) is *over* an edge of E, see Fig. 3. That is, when we consider raw trajectories with a RN, each point is mapped over a road segment by searching for its closest road segment. Because of this and following the approach of Zhixian (2011), the minimum distance between L_i^j and a road segment e_k is computed by Equation 1:

$$d\left(\mathrm{L}_{\mathrm{i}}^{\mathrm{j}}, e_k\right) = \begin{cases} d\left(\mathrm{L}_{\mathrm{i}}^{\mathrm{j}}, e_k\right) \mathit{if}\ \mathrm{L}_{\mathrm{i}}'^{\mathrm{j}} \in e_k \\ \min \begin{Bmatrix} d\left(\mathrm{L}_{\mathrm{i}}^{\mathrm{j}}, get\_vertex\_source\left(e_k\right)\right), \\ d\left(\mathrm{L}_{\mathrm{i}}^{\mathrm{j}}, get\_vertex\_target\left(e_k\right)\right) \end{Bmatrix} \mathit{otherwise} \end{cases} \quad (1)$$

where, L'_i is the projection of L_i^j over e_k, d(L_i^j,e_k) is the perpendicular distance between L_i^j and e_k *and d*(L_i^j, *get_vertex_source*(e_k)) and d(L_i^j, *get_vertex_target*(e_k)) is the Euclidean distance between L_i^j and the *source/target* vertex of e_k. Note that the *d* function is overloaded with the signatures L X $E \rightarrow \mathbb{R}$ and L X $V \rightarrow \mathbb{R}$. The e_k segment, which has the minimum distance

$d(L_i^j, e_k)$ among all the RN segments is where the point L_i^j is mapped. i.e. $get\_edge(L_i^j, Ga) = e_k$.

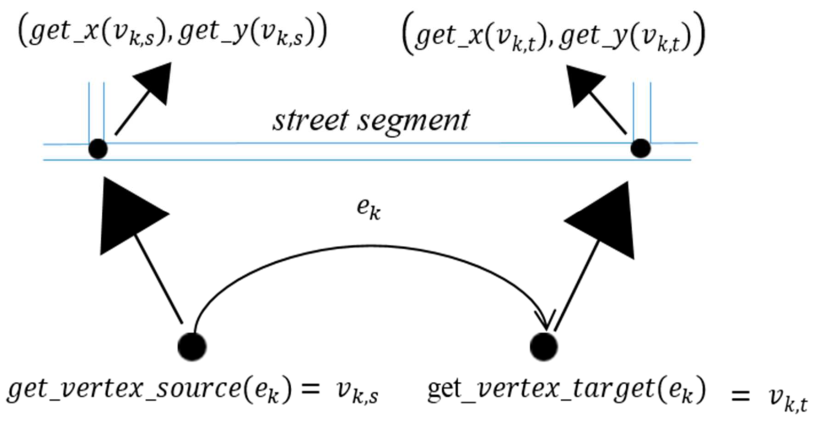

$(get\_x(v_{k,s}), get\_y(v_{k,s}))$
$(get\_x(v_{k,t}), get\_y(v_{k,t}))$
street segment
e_k
$get\_vertex\_source(e_k) = v_{k,s}$
$get\_vertex\_target(e_k) = v_{k,t}$

Fig. 1. A street segment and its corresponding edge e_k

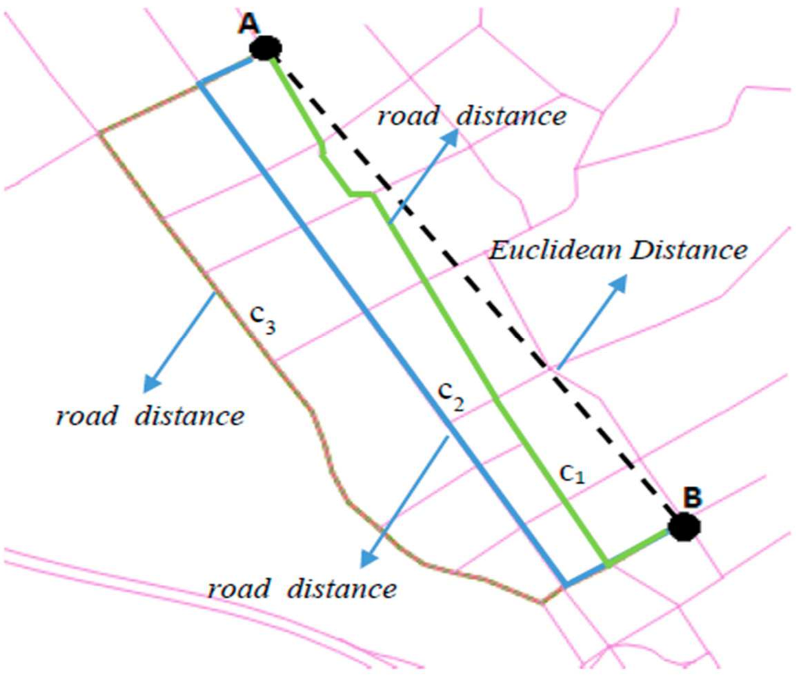


Fig. 2. Road distance according to three criteria Vs the Euclidean distance between two observations *A* and *B*

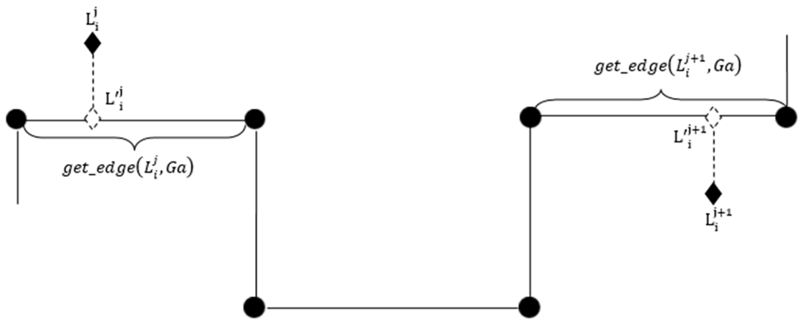


Fig. 3. Edges of the graph *Ga* where L_i^j and L_i^{j+1} fit better

Getting the Location Point from Routing Algorithms

Let a and b be observations where a = ($get\_x(get\_vertex\_target(get\_edge(L_i^j, Ga)))$, $get\_y(get\_vertex\_target(get\_edge(L_i^j, Ga)))$, $set\_time(L_i^j)$) and b = ($get\_x(get\_vertex\_target(get\_edge(L_i^{j+1}, Ga)))$, $get\_y(get\_vertex\_target(get\_edge(L_i^{j+1}, Ga)))$, $set\_time(L_i^{j+1})$).

We use the *set_time* function: $V \rightarrow \mathbb{T}$ to assign a time-stamp to observations a and b. This function is explained in the next subsection.

Then the function $traj(\{L_i^j, L_i^{j+1}\}, c)$ returns a sequence of observations $\{a, o_1, o_2, ..., o_p, b\}$ describing the route between L_i^j and L_i^{j+1} according to a criterion c and a RN, see Fig. 4. Note that the sequence of observations is imputed from the application of a routing algorithm over the graph Ga between its edges $get\_edge(L_i^j, Ga)$ and $get\_edge(L_i^{j+1}, Ga)$. In this way, the (sub) trajectory obtained between L_i^j and L_i^{j+1} according to a criterion c can be described by the Equation 2:

$$traj(L_i^j, L_i^{j+1}, c) = \left\{ \begin{array}{l} L_i^j, \\ (get\_x(get\_vertex\_target(e_1)), \\ get\_y(get\_vertex\_target(e_1)), \\ set\_time(get\_vertex\_target(e_1))), \\ ...., \\ (get\_x(get\_vertex\_target(e_k)), \\ get\_y(get\_vertex\_target(e_k)), \\ set\_time(get\_vertex\_target(e_k))), \\ ..., \\ (get\_x(get\_vertex\_target(e_{p-1})), \\ get\_y(get\_vertex\_target(e_{p-1})), \\ set\_time(get\_vertex\_target(e_{p-1}))), \\ L_i^{j+1} \end{array} \right\} \quad (2)$$

According to the RN mapping defined by Speičvcys *et al.* (2003) the end vertex of an edge e_k is the initial vertex of the edge e_{k+1}, see Fig. 5. Therefore, $get\_x(get\_vertex\_target(e_k))$ = $get\_x(get\_vertex\_source(e_{k+1}))$ and $get\_y(get\_vertex\_target(e_k))$ = $get\_y(get\_vertex\_source(e_{k+1}))$. Note that $get\_edge(L_i^j, Ga) = e_1$ and $get\_edge(L_i^{j+1}, Ga) = e_p$.

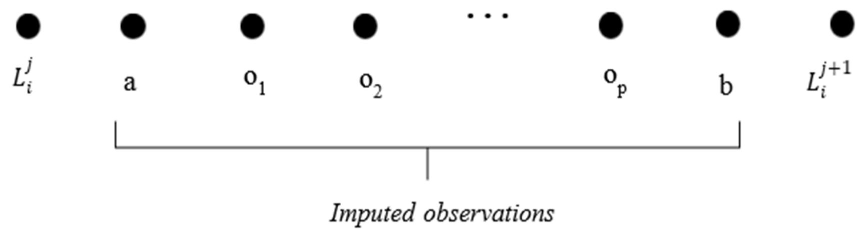


Fig. 4. Imputed observations between the observations a and b

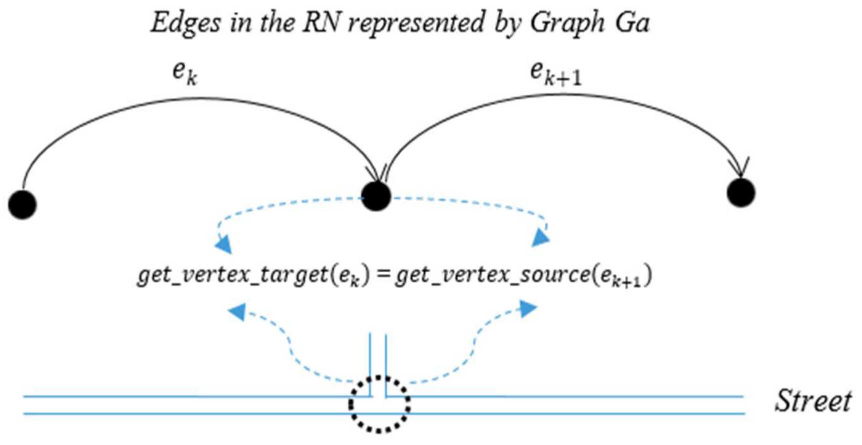

Edges in the RN represented by Graph Ga
e_k
e_{k+1}
$get\_vertex\_target(e_k) = get\_vertex\_source(e_{k+1})$
Street

Fig. 5. The end vertex of an edge e_k is the initial vertex of the edge e_{k+1}.

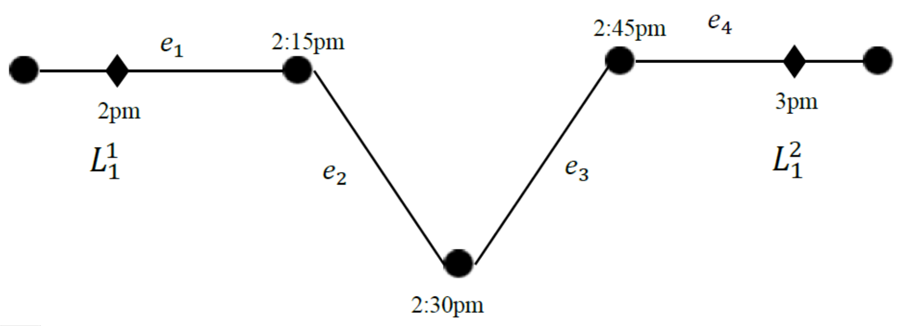


Fig. 6. Example of time assignment to a reconstructed (sub)trajectory

For each imputed observation used for the reconstruction of the trajectory between L_i^j and L_i^{j+1} a time-stamp must be set. For this purpose, we define the *set_time* function.

Getting the Time-Stamps from Routing Algorithms

To compute the time-stamp of each imputed observation of the reconstructed trajectory segment, the difference *get_time*(L_i^{j+1})-*get_time*(L_i^j) is proportionally assigned to each of them. Then, the time-stamp of an imputed point can be computed as follows. Let:

- *get_distance: E* → ℝ. Function applied to an edge to get the distance of the edge
- $D_{L_i^j}$: The distance from the observation L_i^j to *get_vertex_source*(*get_edge*(L_i^j ,*Ga*))
- $D_{L_i^{j+1}}$: The distance from L_i^{j+1} to *get_vertex_source*(*get_edge*(L_i^{j+1} ,*Ga*))

Then, see Equation 3:

$$\begin{aligned} &total\_distance(L_i^j, L_i^{j+1}) = \\ &get\_distance(get\_edge(L_i^j, Ga)) - D_{L_i^j} \\ &+\sum_{k=2}^{p-1} get\_distance(e_k) + D_{L_i^{j+1}} \end{aligned} \tag{3}$$

Note that the addition begins at $k = 2$ since we suppose that *get_edge*(L_i^j ,*Ga*) = e_1 and ends at *p-1* because *get_edge*(L_i^{j+1} ,*Ga*) = e_p. That is, both e_1 and e_p are part of the resulting sequence. Then, the time-stamp of a *get_vertex_target*(e_k) vertex is computed by Equation 4:

$$\begin{aligned} &set\_time\left(get\_vertex\_target\left(e_k\right)\right) = get\_time\left(\mathrm{L}_i^j\right) + \\ &\frac{\left(\sum_{k=1}^{p-1} get\_distance\left(e_k\right)\right) - D_{L_i^j}}{total\_distance\left(\mathrm{L}_i^j, \mathrm{L}_i^{j+1}\right)} * \begin{pmatrix} get\_time\left(\mathrm{L}_i^{j+1}\right) \\ -get\_time\left(\mathrm{L}_i^j\right) \end{pmatrix} \end{aligned} \tag{4}$$

For example. Let us consider the reconstructed trajectory between the observations L_1^1 and L_1^2 shown in Fig. 6 where we get the edges e_1, e_2, e_3, e_4. The *get_time*(L_1^1) = 2 pm and *get_time*(L_1^2) = 3pm, then *get_time*(L_1^2)- *get_time*(L_1^1) = 1 hour, i.e., an hour must be proportionally divided among the edges. Then *get_distance*(e_1) = 12, *get_distance*(e_2) = 10, *get_distance*(e_3) = 10, *get_distance*(e_4) = 10, $D_{L_1^1}$ = 2, $D_{L_1^2}$ = 2. Note that *get_edge*(L_1^1 ,G) = e_1 and *total_distance*(L_1^1 , L_1^2) = 40.

For $k = 1$

set_time(*get_vertex_target*(e_1)) = *get_time*(L_1^1) + ((*get_distance*(e_1)- $D_{L_1^1}$)/*total_distance*(L_1^1 , L_1^2)) * *get_time*(L_1^2)- *get_time*(L_1^1) = 2 pm + 1/4 = 2:15 pm

For $k = 2$

set_time(*get_vertex_target*(e_2)) = *get_time*(L_1^1) +((*get_distance*(e_1) + *get_distance*(e_2)- $D_{L_1^1}$)/*total_distance*(L_1^1 , L_1^2)) * *get_time*(L_1^2)- *get_time*(L_1^1) = 2 pm + 2/4 = 2:30 pm

For $k = 3$

set_time(*get_vertex_target*(e_3)) = *get_time*(L_1^1) +((*get_distance*(e_1) + *get_distance*(e_2) + *get_distance*(e_3)- $D_{L_1^1}$)/*total_distance*(L_1^1 , L_1^2)) * *get_time*(L_1^2)- *get_time*(L_1^1) = 2 pm + 3/4 = 2:45 pm

Note that, after the reconstruction, it is possible that the imputed data points do not meet the β and τ thresholds. In this case, the longitude of the street segments is longer than the β threshold because this imputation stage only gets location points based on the edges of a graph *Ga* that represents the segments of a RN where a MO moves. Additional imputed data points can be obtained using interpolation methods between the imputed points, i.e., the start and the end vertex of an edge. The following equations find additional data points over a segment e_k based on the line equation, see Equation 5:

$$y = \frac{get\_y(get\_vertex\_target(e_k)) - get\_y(get\_vertex\_source(e_k))}{get\_x(get\_vertex\_target(e_k)) - get\_x(get\_vertex\_source(e_k))} * \left(x - get\_x(get\_vertex\_target(e_k))\right) + get\_y(get\_vertex\_target(e_k)) \quad (5)$$

where, *get_x* and *get_y* are found slicing the segment e_k in such a way that $A \leq \beta \land road\_distance(L_i^j, L_i^{j+1}, c) \leq A*\beta$. Where A is the amplitude of the sub segments of e_k, see Equation 6 and 7.

$$x_i = get\_x(get\_vertex\_source(e_k)) + \frac{d_i}{road\_distance(L_i^j, L_i^{j+1}, c)} * \begin{pmatrix} get\_x(get\_vertex\_target(e_k)) \\ -get\_x(get\_vertex\_source(e_k)) \end{pmatrix} \quad (6)$$

$$y_i = get\_y(get\_vertex\_source(e_k)) + \frac{d_i}{road\_distance(L_i^j, L_i^{j+1}, c)} * \begin{pmatrix} get\_y(get\_vertex\_target(e_k)) \\ -get\_y(get\_vertex\_source(e_k)) \end{pmatrix} \quad (7)$$

where, $d_i = A * i$, $1 \leq i \leq N$-1. N is the number of intervals so that $road\_distance(L_i^j, L_i^{j+1}, c) = N * A$.

In Fig. 7 we show an example for finding additional data points for a segment e_k, where $get\_x(get\_vertex\_source(e_k)) = 3$, $get\_y(get\_vertex\_source(e_k)) = 1$, $get\_x(get\_vertex\_target(e_k)) = 6$, $get\_y(get\_vertex\_target(e_k)) = 5$. Let $\beta = 1.25$, $road\_distance(L_i^j, L_i^{j+1}, c) = 5$, then we choose $A = 1.25$ and $N = 4$:

- $d_1 = 1.25$, then $x_1 = 3.75$, $y_1 = 2$
- $d_2 = 2.5$, then $x_2 = 4.5$, $y_1 = 3$
- $d_3 = 3.75$, then $x_3 = 5.25$, $y_1 = 4$

Thus, the set of additional data points between (3, 1) and (6, 5) is {(3.75, 2), (4.5, 3), (5.25, 4)}. The time-stamps for each of these points can be found by the proportional assignment of the time difference between observations. The results are also shown in Fig. 8, where we suppose that *set_time(get_vertex_source(e*$_k$*))* = 12 pm and *set_time(get_vertex_target(e*$_k$*))* = 4pm.

Implementation of the *traj* Function

Given (a) users check-in records describing a set of trajectories $TS = \{T_i\}$ from a certain location-based service and (b) a user criterion *c;* we claim that a "good" route should (a) meet the user criterion and (b) returns a more detailed trajectory T'_I (as long as T_i has at least a pair of low-sampled observations). Algorithm 1 calls the Function 1 (*traj*) for each pair of consecutive low-sampled observations of a trajectory T_i.

```
INPUT: { $L_i^j$, $L_i^{j+1}$ |$road\_distance(L_i^j, L_i^{j+1}, c) \geq \beta \land t_i^{j+1} - t_i^j \geq \tau$}
        c: criterion of movement

OUTPUT: { $L_i^j$, $L_i^{j+1}$ |$road\_distance(L_i^j, L_i^{j+1}, c) \geq \beta' \land t_i^{j+1} - t_i^j \geq \tau' \land \beta' < \beta \land \tau' < \tau$}

BEGIN
1.   // To apply a routing algorithm according to criterion c between get_edge( $L_i^j$, Ga) and get_edge( $L_i^{j+1}$, Ga)

2.   FOR EACH $e_k$

3.       // Use set_time function for setting the time to each vertex resulting from the routing algorithm

4.       $O_k$ ← (get_x(get_vertex_target($e_k$)), get_y(get_vertex_target($e_k$)), set_time(get_vertex_target($e_k$)))

5.       IF road_distance($O_k$, $O_{k+1}$, c) ≥ $\beta \land$ get_time($O_{k+1}$)- get_time($O_k$) ≥ $\tau$ THEN

6.       // Interpolate between $O_k$ and $O_{k+1}$
         Use Equation 6 and 7

7.       Trajectory ← { $L_i^j$,(get_x(get_vertex_target($e_1$)),
         get_y(get_vertex_target($e_1$)),
         set_time(get_vertex_target($e_1$))), …,
         (get_x(get_vertex_target($e_k$)),
         get_y(get_vertex_target($e_k$)),
         set_time(get_vertex_target($e_k$))), …,
         (get_x(get_vertex_target($e_{p-1}$)),
         get_y(get_vertex_target($e_{p-1}$)),
         set_time(get_vertex_target($e_{p-1}$))),
         $L_i^{j+1}$ }
8.   END
9.   RETURN Trajectory
END
```

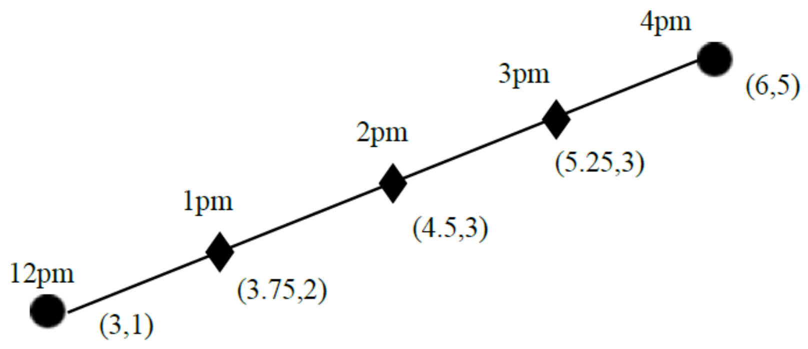


Fig. 7. Additional imputed data points and time-stamps data points for the start and end vertex of a same edge

Reconstruction of a Trajectory Algorithm

```
INPUT: TS: set of sampled trajectories
       c: criterion of movement

OUTPUT: {TS'|∀$T_i$ ∈ TS' |road_distance($L_i^j$, $L_i^{j+1}$, c) ≥ β' $t_i^{j+1}$ - $t_i^j$ ≥ τ' β' < β τ' < τ}
BEGIN
1. TS' ← ø, $T'_i$ ← ø

2. FOR EACH $T_i$ IN TS
3.     FOR EACH $L_i^j$ in $T_i$
4.     IF road_distance($L_i^j$, $L_i^{j+1}$, c) ≥ β $t_i^{j+1}$ - $t_i^j$ ≥ τ
THEN

5.             Trajectory ← traj({$L_i^j$, $L_i^{j+1}$}, c)
6.             APPEND Trajectory TO $T'_i$

7.     ELSE

8.             APPEND {$L_i^j$, $L_i^{j+1}$} TO $T'_i$
9.             NEXT $L_i^j$
10.    END
11.    END
12. END
13. RETURN TS'
END
```

How the Traj Function Works: An Example

To explain how the *traj* function works, let us consider a set of check-in data describing a trajectory of a particular user as shown in Table 2 and the RN of the city of Medellín, Colombia.

We get the nearest edges *get_edge(Check-in A, Ga)*, *get_edge(Check-in B, Ga)* and *get_edge(Check-in C, Ga)* for each check-in observation. Next, the change of the imputed data of the reconstructed trajectories is shown as the criterion *c* changes. Let *β* be less than the actual road distance between each pair of check-in and *c* be less than the difference between time check-ins. Distance (Fig. 8), time (Fig. 9) and tourist attraction (Fig. 10) criteria were used. We also show the original trajectory, see Fig. 11.

Measuring and Comparing the Resulting Reconstructed Trajectories using Different Criteria with the real Ones

There are many approaches for measuring the similarity between trajectories in the literature review (Zhao *et al*., 2009; Tiakas *et al*., 2009; Hung *et al*., 2011). A similar approach proposed by Zhao *et al.* (2009) is followed:

Two trajectories T_1 and T_2 are spatio-temporally similar, iff (a) Trajectories T_1 and T_2 have the same temporal granularity and the trajectories are spatially similar, i.e., $SIM_{POI}(T_1,T_2,\theta) < \theta$, where $SIM_{POI}(T_1,T_2,\theta)$ is a spatial similarity measure, see Equation 8, θ is a threshold to consider that two trajectories are spatially similar and that the Point Of Interest (POI) represents an interesting place.

$$SIM_{POI}(T_1,T_2,\theta) = \frac{POI_{T_1} \cap POI_{T_2}}{POI_{T_1} \cup POI_{T_2}} \tag{8}$$

The reconstructed trajectories have the same temporal granularity according to Zhao *et al.* (2009) because they have similar time-stamp assignment according to the method proposed here, in which the time-stamps are assigned proportionally. We consider the POIs as the road segments that a trajectory traverses. Next, we compute the SIM_{POI} measured for 80 high- sampled trajectories in Medellín. The check-in data were simulated (time and location data were deleted) for those trajectories to get low-sampled trajectories and the (sub)trajectories were computed based on some criteria using the *traj* function between the simulated check-ins, see Fig. 12.

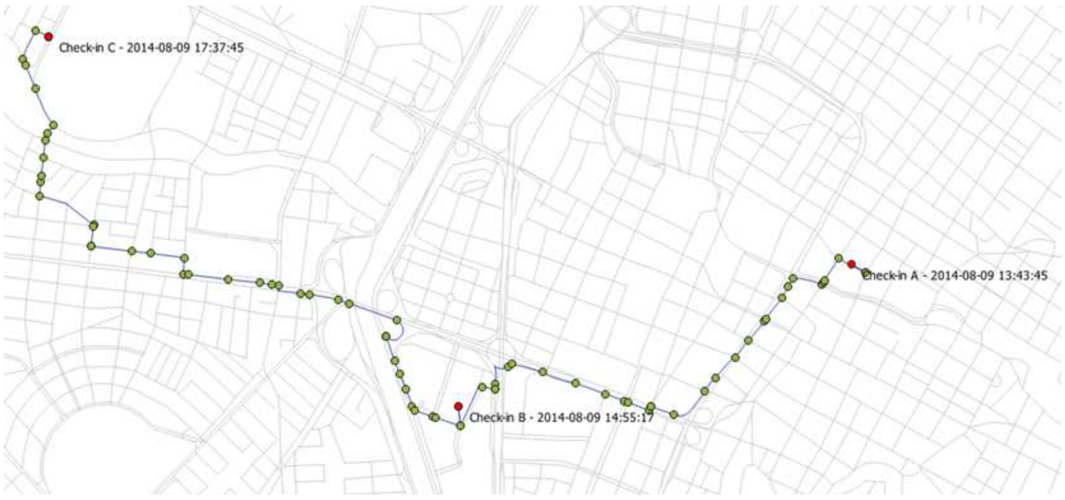

Check-in C - 2014-08-09 17:37:45
Check-in A - 2014-08-09 13:43:45
Check-in B - 2014-08-09 14:55:17

Fig. 8. Reconstructed trajectory using distance criterion

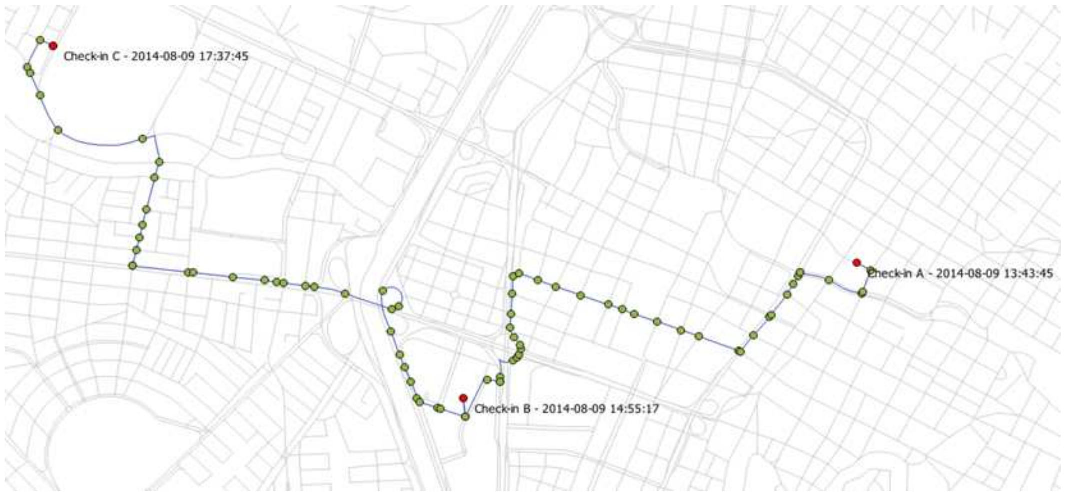

Check-in C - 2014-08-09 17:37:45
Check-in A - 2014-08-09 13:43:45
Check-in B - 2014-08-09 14:55:17

Fig. 9. Reconstructed trajectory using time criterion

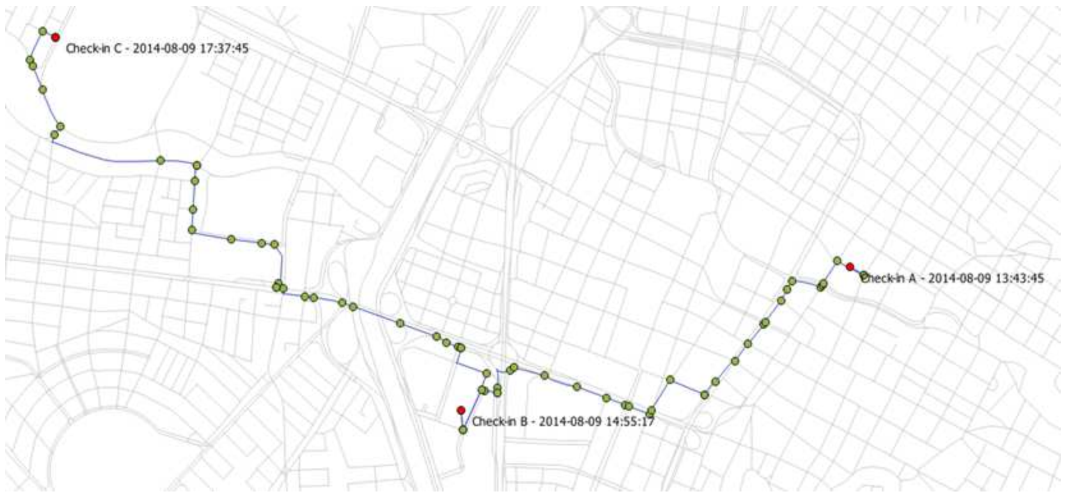

Check-in C - 2014-08-09 17:37:45
Check-in A - 2014-08-09 13:43:45
Check-in B - 2014-08-09 14:55:17

Fig. 10. Reconstructed trajectory using tourist attraction criterion

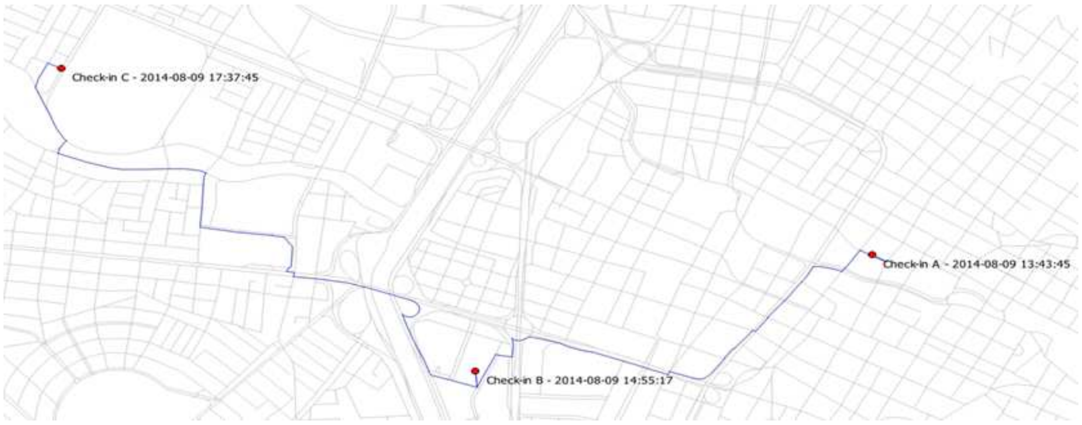

Check-in C - 2014-08-09 17:37:45
Check-in A - 2014-08-09 13:43:45
Check-in B - 2014-08-09 14:55:17

Fig. 11. Original trajectory

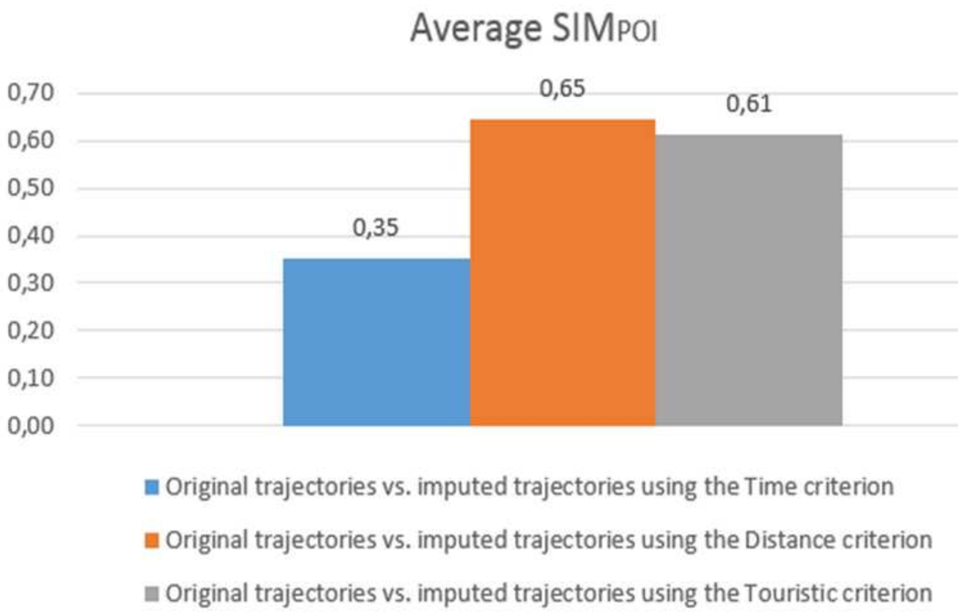


Fig. 12. The average similarity measure between the reconstructed trajectories and the original ones for a set of 80 users

Table 2. Check- in data of a particular user

User	Data Point	POI Name	(x,y,t)
15307763	Check-in A	Shop	(-75.562555, 6.249437, 20140809134345)
15307763	Check-in B	Restaurant	(-75.576790, 6.244406, 20140809145517)
15307763	Check-in C	Shop	(-75.591672, 6.257514, 20140809173745)

Note how the average SIM_{POI} is higher when the distance criterion was used followed by the tourist attraction criterion, i.e., the best imputation process for this 80 trajectories could be achieved when some of these criteria were used. However, remember that the purpose of the trajectory reconstruction proposed here is to discover the new possibilities of reconstruction as an imputation process of the actual trajectories. The trajectory reconstruction procedure takes place in order to transform low-sampled location data into trajectories with a better sampling so that we can acquire some useful knowledge.

Technical Details

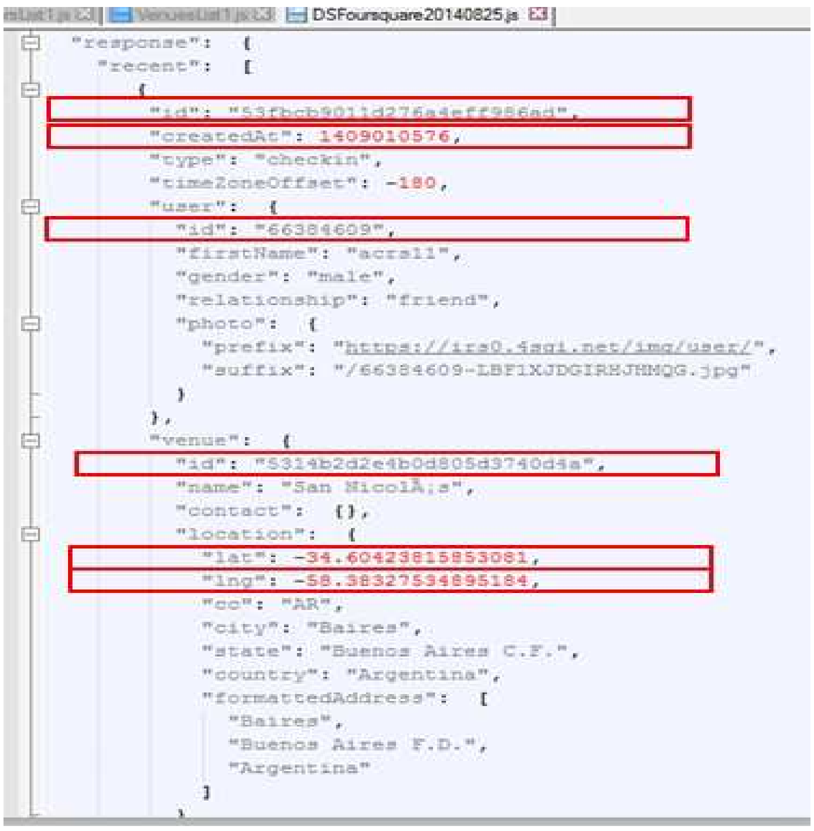
This technical details are intended to offer a more comprehensive understanding of the solution and it serves as a reference for future implementation of the system. It also pretends to provide the technical details to replicate the previously executed experiments. We used the following software tools:

- *Apigee*. An infrastructure for creating and operating APIs and apps
- *Foursquare API*. Foursquare for developers. Access to world-class places database of Foursquare
- *Openstreetmap* is a map of the world free to use under an open license
- *Osm2po-4.8.8*. Routing On OpenStreetMap is both, a converter and a routing engine, the converter parses OpenStreetMap's XML-Data and makes it routable
- *Pentaho Data Integration 5.0.1*. Delivers extraction, transformation and loading capabilities
- *Postgress 9.2*. An Object-Relational Database Management System (ORDBMS)
- *PgRouting*. Extends the PostGIS/PostgreSQL geospatial database to provide geospatial routing functionality
- *Qgis Desktop 2.0.1*. A Free and Open Source Geographic Information System. Create, edit, visualise, analyse and publish geospatial information

Next, we detail our sources. The source data can be extracted from multiple location-based devices and applications. For this technical proposal, JSON files were generated using Foursquare API and then read using Pentaho Data Integration. The Foursquare API was accessed using Apigee. We got details of the users from Foursquare (https://developer.foursquare.com/docs/users/users).

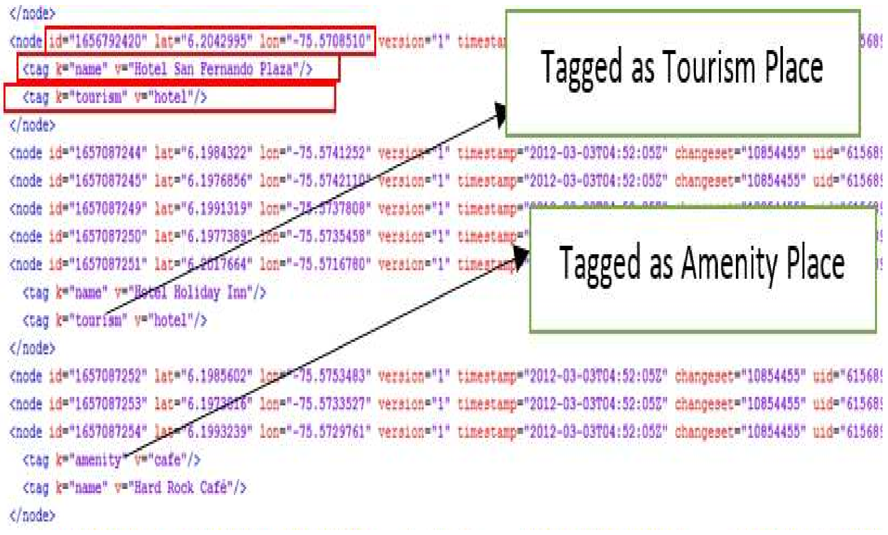
Data of the venues (POIs) registered in Foursquare in the city of Medellín, Colombia were also collected. For the points where the people make check-in, data of 80 active random users living in the Medellin, Colombia city were collected. Figure 13 shows an instance of the file gotten with this response.

Data of a list of check-in of the users described above were gathered during a week. A list of touristic points of Medellín, Colombia city were defined. Those were extracted from OpenStreetMap were people can tagged those places as touristic. See an example of this file in Fig. 14. The location for each one was also included. The idea behind this definition is to assign a lower cost to segments of the streets near to those touristic points. The Graph Map was gotten using osm2po-4.8.8. The *traj* function was implemented and carries out the reconstruction task proposed in this study. The implementation of the *traj* function, additional documents and all the software can be found at https://www.dropbox.com/sh/3mlfrveicpwjrgp/AADzZQ8jneo9jpBFlofFkGSba?dl=0.



```
"response":  {
  "recent":  [
     {
       "id": "53fbcb9011d276a4eff986ad",
       "createdAt": 1409010576,
       "type": "checkin",
       "timeZoneOffset": -180,
       "user":  {
         "id": "66384609",
         "firstName": "acrall",
         "gender": "male",
         "relationship": "friend",
         "photo":  {
           "prefix": "https://irs0.4sqi.net/img/user/",
           "suffix": "/66384609-LBF1XJDGIRHJHMQG.jpg"
         }
       },
       "venue":  {
         "id": "5314b2d2e4b0d805d3740d4a",
         "name": "San NicolÃ;s",
         "contact":  {},
         "location":  {
           "lat": -34.60423815853081,
           "lng": -58.38327534895184,
           "cc": "AR",
           "city": "Baires",
           "state": "Buenos Aires C.F.",
           "country": "Argentina",
           "formattedAddress":  [
             "Baires",
             "Buenos Aires F.D.",
             "Argentina"
           ]
         },
```

Fig. 13. Instance of the file of check-in data



```
</node>
<node id="1656792420" lat="6.2042995" lon="-75.5708510" version="1" timesta
  <tag k="name" v="Hotel San Fernando Plaza"/>
  <tag k="tourism" v="hotel"/>
</node>
<node id="1657087244" lat="6.1984322" lon="-75.5741252" version="1" timestamp="2012-03-03T04:52:05Z" changeset="10854455" uid="61568
<node id="1657087245" lat="6.1976856" lon="-75.5742110" version="1" timestamp="2012-03-03T04:52:05Z" changeset="10854455" uid="61568
<node id="1657087249" lat="6.1991319" lon="-75.5737808" version="1" timestamp=
<node id="1657087250" lat="6.1977389" lon="-75.5735458" version="1" timestamp="
<node id="1657087251" lat="6.2017664" lon="-75.5716780" version="1" timestamp
  <tag k="name" v="Hotel Holiday Inn"/>
  <tag k="tourism" v="hotel"/>
</node>
<node id="1657087252" lat="6.1985602" lon="-75.5753483" version="1" timestamp="2012-03-03T04:52:05Z" changeset="10854455" uid="61568
<node id="1657087253" lat="6.1973816" lon="-75.5733527" version="1" timestamp="2012-03-03T04:52:05Z" changeset="10854455" uid="61568
<node id="1657087254" lat="6.1993239" lon="-75.5729761" version="1" timestamp="2012-03-03T04:52:05Z" changeset="10854455" uid="61568
  <tag k="amenity" v="cafe"/>
  <tag k="name" v="Hard Rock Café"/>
</node>
```

Fig. 14. Instance of the file of points of interest

Conclusion

Valuable information can be extracted from trajectories. It can be useful for location-based services applications including trip planning, personalized navigation routing services, mobile commerce and location-based recommendation services. In this study, we reconstructed low-sampling trajectories using the personalization features of the routing theory based on a criterion evaluation over a graph. The *traj* function with different criteria can be used as an input for different mining algorithms over trajectories as a way to deal with analytics using uncertain trajectories. Here, we claim that analytics over reconstructed trajectories can change depending on the criterion used for their reconstruction. Moreover, this criteria-based reconstruction can be used to perform analytical tasks and to offer the possibility of formulating questions based on user criteria, such as:

- How do *regions of interest* (Cao *et al*., 2005) change according to a chosen criterion of reconstruction during a determined time?
- What are the main bottlenecks in the city in a determined period according to a certain movement reconstruction criterion?
- What would be the fuel consumption if the vehicles moved according to a certain criterion in a determined period?

Author's Contributions

All the authors contributed equally to the writing of the manuscript. All the authors discussed and conceptualized the idea, contributed to analyses and interpretation of the results and to the preparation of the final manuscript.

Ethics

All the authors believe that there are no ethical issues that may arise after the publication of this manuscript.

References

Cao, H., N. Mamoulis and D.W. Cheung, 2005. Mining frequent spatio-temporal sequential patterns. Proceedings of the 5th IEEE International Conference on Data Mining, Nov. 27-30, IEEE Xplore Press, Houston, Texas. DOI: 10.1109/ICDM.2005.95

Chang, K.P., L.Y. Wei, M.Y. Yeh and W.C. Peng, 2011. Discovering personalized routes from trajectories. Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks, Nov. 1-4, ACM, Chicago, Illinois, pp: 33-40. DOI: 10.1145/2063212.2063218

Chen, Z., H. Shen, X. Zhou, Y. Zheng and X. Xie, 2010. Searching trajectories by locations: An efficiency study. Proceedings of the ACM SIGMOD International Conference on Management of data, Jun. 6-11, ACM, Indianapolis, Indiana, USA, pp: 255-266. DOI: 10.1145/1807167.1807197

Chen, Z., H.T. Shen and X. Zhou, 2011. Discovering popular routes from trajectories. Proceedings of the IEEE 27th International Conference on Data Engineering, Apr. 11-16, IEEE Xplore Press, Hannover, Germany, pp: 900-911. DOI: 10.1109/ICDE.2011.5767890

Chow, C.Y. and M.F. Mokbel, 2011. Privacy of Spatial Trajectories. In: Computing with Spatial Trajectories, Zheng, Y. and X. Zhou (Eds.), Springer, New York, ISBN-10: 978-1-4614-1628-9, pp: 109-141.

Da Silva, E.R., C. de Baptista, L. Menezes and A. Paiva, 2008. Personalized path finding in road networks. Proceedings of the 4th International Conference on Networked Computing and Advanced Information Management, Sep. 2-4, IEEE Xplore Press, Gyeongju, Korea, pp: 586-591. DOI: 10.1109/NCM.2008.211

De Almeida, V.T. and R.H. Güting, 2005. Supporting uncertainty in moving objects in network databases. Proceedings of the 13th Annual ACM International Workshop on Geographic Information Systems, (GIS' 05), ACM, pp: 31-40. DOI: 10.1145/1097064.1097070

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numerische Mathematik, 1: 269-271. DOI: 10.1007/BF01386390

Hart, P.E., N.J. Nilsson and B. Raphael, 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybernet., 4: 100-107. DOI: 10.1109/TSSC.1968.300136

Hochmair, H., 2005. Towards a Classification of Route Selection Criteria for Route Planning Tools. In: Developments in Spatial Data Handling, Hochmair, H. (Ed.), Springer Berlin Heidelberg, ISBN-13: 978-3-540-22610-9, pp: 481-492.

Hsieh, H. and C. Li, 2013. Constructing trip routes with user preference from location check-in data. Proceedings of the ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication, Sep. 8-12, ACM, Zurich, Switzerland, pp: 195-198. DOI: 10.1145/2494091.2494155

Hung, C.C., W.C. Peng and W.C. Lee, 2011. Clustering and aggregating clues of trajectories for mining trajectory patterns and routes. VLDB J., 24: 1-24. DOI: 10.1007/s00778-011-0262-6

Liu, H., L.Y. Wei, Y. Zheng, M. Schneider and W.C. Peng, 2011. Route discovery from mining uncertain trajectories. Proceedings of the IEEE 11th International Conference on Data Mining Workshops, Dec. 11-11, IEEE Xplore Press, Vancouver, Canada, pp: 1239-1242. DOI: 10.1109/ICDMW.2011.149

Marketos, G. and Y. Theodoridis, 2009. Mobility Data Warehousing and Mining. VLDB PhD Workshop, Philippe Rigaux and Pierre Senellart, (Eds.), Aug. 24, Lyon, France

Nadi, S. and M.R. Delavar, 2011. Multi-criteria, personalized route planning using quantifier-guided ordered weighted averaging operators. Int. J. Applied Earth Observ. Geoinform., 13: 322-335. DOI: 10.1016/j.jag.2011.01.003

Orlando, S., R. Orsini, A. Raffaetà, A. Roncato and C. Silvestri, 2007. Trajectory data warehouses: Design and implementation issues. J. Comput. Sci. Eng., 1: 211-232.

Schultes, D., 2008. Route planning in road networks. Doctoral dissertation, Universität Fridericiana zu Karlsruhe. Karlsruhe, Germany.

Spaccapietra, S., C. Parent, M.L. Damiani, J.A. de Macedo and F. Porto *et al*., 2008. A conceptual view on trajectories. Data Knowledge Eng., 65: 126-146. DOI: 10.1016/S0169-023X(08)00013-X

Speičvcys, L., C.S. Jensen and A. Kligys, 2003. Computational data modeling for network-constrained moving objects. Proceedings of the 11th ACM International Symposium on Advances in Geographic Information Systems, (GIS' 03), ACM, pp: 118-125. DOI: 10.1145/956676.956692

Tiakas, E., A.N. Papadopoulos, A. Nanopoulos, Y. Manolopoulos and D. Stojanovic *et al*., 2009. Searching for similar trajectories in spatial networks. J. Syst. Software, 82: 772-788. DOI: 10.1016/j.jss.2008.11.832

Trajcevski, G., 2011. Uncertainty in Spatial Trajectories. In: Computing with Spatial Trajectories, Zheng, Y. and X. Zhou (Eds.), Springer, New York, ISBN-13: 978-1-4614-1628-9, pp: 63-107.

Wei, L.Y., Y. Zheng and W.C. Peng, 2012. Constructing popular routes from uncertain trajectories. Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Aug. 12-16, ACM, Beijing, China, pp: 195-203. DOI: 10.1145/2339530.2339562

Yuan, J., Y. Zheng, C. Zhang and W. Xie, 2010. T-drive: Driving directions based on taxi trajectories. Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, Nov. 2-5, ACM, San Jose, California, pp: 99-108. DOI: 10.1145/1869790.1869807

Zhao, H., Q. Han, H. Pan and G. Yin, 2009. Spatio-temporal similarity measure for trajectories on road networks. Proceedings of the 4th International Conference on Internet Computing for Science and Engineering, Dec. 21-22, IEEE Xplore Press, Harbin, China, pp: 189-193. DOI: 10.1109/ICICSE.2009.18

Zheng, K., Y. Zheng, X. Xie and X. Zhou, 2012. Reducing uncertainty of low-sampling-rate trajectories. Proceedings of the IEEE 28th International Conference on Data Engineering, Apr. 1-5, IEEE Xplore Press, Washington, DC., pp: 1144-1155. DOI: 10.1109/ICDE.2012.42

Zhixian, Y.A.N., 2011. Semantic trajectories: Computing and understanding mobility data. Doctoral dissertation, École Polytechnique Fédérale De Lausanne, Lausanne, Switzerland.