Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 9178539, 15 pages
http://dx.doi.org/10.1155/2016/9178539

Research Article

Human Mobility Modelling Based on Dense Transit Areas Detection with Opportunistic Sensing

Fernando Terroso-Sáenz, Mercedes Valdes-Vela, Aurora González-Vidal, and Antonio F. Skarmeta

Department of Information and Communications Engineering, Computer Science Faculty, University of Murcia, Murcia, Spain

Correspondence should be addressed to Fernando Terroso-Sáenz; fterroso@um.es

Received 20 May 2016; Accepted 28 June 2016

Academic Editor: Celimuge Wu

With the advent of smartphones, opportunistic mobile crowdsensing has become an instrumental approach to perceive large-scale urban dynamics. In this context, the present work presents a novel approach based on such a sensing paradigm to automatically identify and monitor the areas of a city comprising most of the human transit. Unlike previous approaches, the system performs such detection in real time at the same time the opportunistic sensing is carried out. Furthermore, a novel multilayered grill partitioning to represent such areas is stated. Finally, the proposal is evaluated by means of a real-world dataset.

1. Introduction

For the last years, smartphones have been the center of most of the technological advances due to their growing popularity. As a result of these improvements, they are now equipped with several sensors like GPS, accelerometer, microphone, and so forth.

This palette of sensors allows capturing a large amount of contextual information related to the phone's holders and their surrounding environment [1]. This has eased the development of the *mobile crowdsensing* (MCS) or *human/people sensing* paradigm so as to perceive large-scale phenomena that can not be detected at an individual level [2].

One of the most useful phenomena to be perceived is human dynamics. Due to the steady improvement of the positioning sensors installed in mobile devices or vehicles and the fact that location is the most critical element in reflecting users' movement, the *mobility mining* discipline is one of the domains where MCS has been most widely applied so as to uncover different human mobility aspects [3]. In turn, this eases the deployment of innovative location-based services like predictive queries for moving object databases [4] or pervasive navigation systems [5].

Although several studies already exist, mobility mining solutions based on MCS still face the following challenges.

(i) The intensive usage of the positioning and communication capabilities of mobile devices required by this type of solution is rather battery draining. This is an important barrier for users to contribute to MCS-based solutions for mobility detection. Nonetheless, when it comes to composing detailed mobility models, most works assume that a large set of users are always available to report their location traces.

(ii) Existing mechanisms usually follow an offline mining process of a previously gathered mobility dataset. As a result, the extracted knowledge is fixed and can not smoothly adapt to changes of human dynamics.

(iii) Last but not least, location data is quite sensitive in terms of privacy for many people. Therefore, mobility models should be designed so that they can not be used to uncover meaningful places or routes of particular users.

In this context, the present work proposes an innovative framework for human mobility modelling with opportunistic

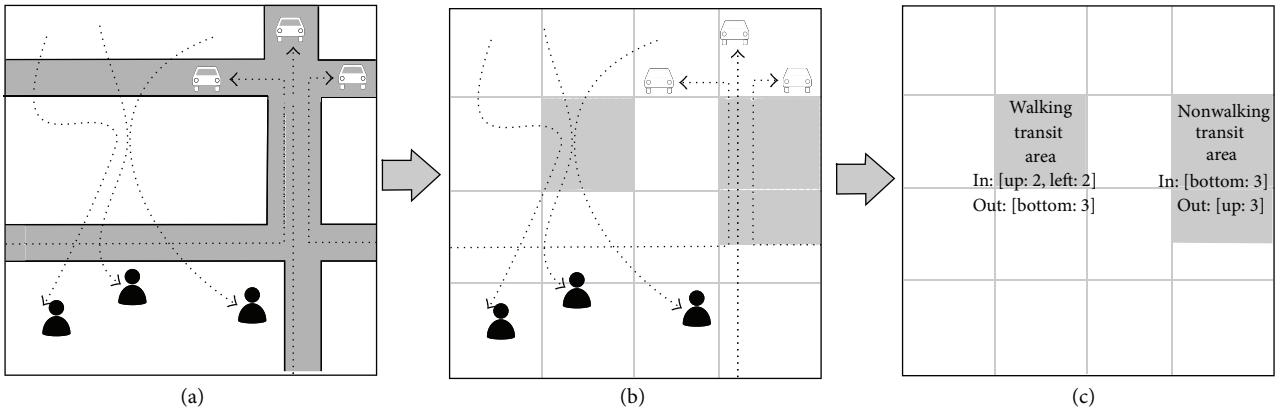


Figure 1: Overview of transit areas detection in the 2D space. (a) Users' raw trajectories, (b) detection high transit areas, and (c) transit areas parameters extraction.

MCS that considers the aforementioned open challenges. The key goal of the proposal is to detect regions with a high density of human transit that capture most of the mobility of a city.

In large city deployments, the number of these dense transit areas is usually very high. Hence, in order to ease their storage and management, a novel region abstraction for mobility mining is introduced. As Figure 1 shows, the idea is to represent such regions of interest with basic geometrical forms and define the incoming and outgoing flow of people inside each region with respect to their (left, right, up, and bottom) sides.

To do so, an online aggregation of the spatiotemporal traces from a set of contributing users is proposed. Unlike previous offline proposals, the introduced mechanism allows discovering the target regions at the same time the trajectories are being received. Once a stable set of transit regions have been discovered, they are continuously monitored by a subset of suitable contributors who are dynamically selected.

The goal of such monitoring is to detect sudden or long-standing meaningful changes of human movement within the detected regions like people driving slower than usual or walking in unusual directions. These mobility shifts can be signs of events of interest, like unplanned demonstrations or serious traffic problems, whose early perception is of great help for many public and private stakeholders.

All in all, bearing in mind the open challenges of mobility mining based on MCS listed before, the salient contributions of the present work are the following.

(i) A novel mechanism uses only a subset of contributors to monitor the state of the composed transit areas: since the rest of users are deactivated, it reduces the extra cost of taking part of this type of solution.

(ii) A new solution explicitly detects changes in the movement of people within the target areas.

(iii) As far as privacy is concerned, the model of the detected areas only exposes general mobility information without disclosing any personal details of the contributors. This allows sharing the detected areas with third-party services without suffering serious privacy leaks.

Finally, the remainder of the paper is structured as follows. To start with, Section 2 is devoted to describing in detail the concept of dense transit area and the logic structure of the proposed system. Section 3 puts forward the procedure to discover such transit areas. Next, how these areas are monitored is stated in Section 4. Then, Section 5 discusses the main results of the experiments. An overview about mobility mining is put forward in Section 6. Finally, the main conclusions and the future work are summed up in Section 7.

2. Dense Transit Areas Detection System

This section is devoted to explaining the goal of the proposal along with the architecture. For the sake of clarity, Abbreviations summarizes the key acronyms and symbols used in the following sections.

2.1. Dense Transit Area Definition. The main goal of the system is to detect the spatial areas within a city where a high density of human transit exists. In our setting, such human transit is defined as the routes that people follow to move from one place to another (e.g., home, work, and school).

Definition 1. A city's region of influence is the spatial region comprising all the frequent origins $\mathcal{O}$ and destinations $\mathcal{D}$ of its citizens.

Definition 2. A route of a person p, $r(p)$, is the continuous movement in a city's region of influence from an origin $o_r \in \mathcal{O}$ to a destination $d_r \in \mathcal{D}$.

Definition 3. The lifetime of a route $r(p)$, $t(r(p))$, is the time interval $[t_o, t_d]$ between the instant at which p departed from $\mathcal{O}_r(t_o)$ and the arrival time at $\mathcal{D}_r(t_d)$.

Definition 4. A subroute of a route $r(p)$, $r(p)^s$, is the part of the continuous movement of r during a time interval $[t_i, t_f] \subseteq t(r(p))$.

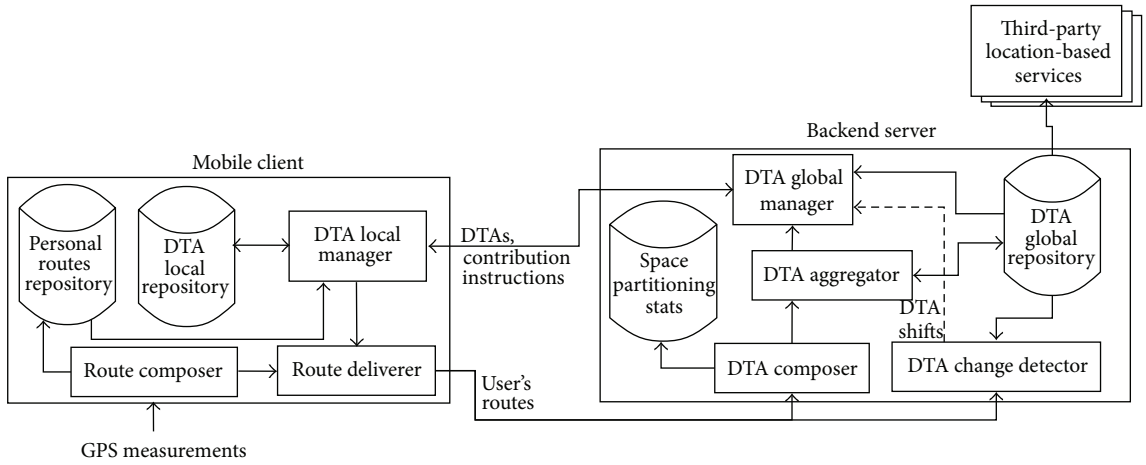


Figure 2: System architecture.

Bearing in mind the aforementioned concepts of human movement, we can then come up with a transit area definition.

Definition 5. A dense transit area (DTA) is a spatial region that has been visited by a set of subroutes $\mathscr{R}^s$, $|\mathscr{R}^s| \gg 1$, from a set of people $\mathscr{P}$, $|\mathscr{P}| \gg 1$.

Consequently, a DTA represents a spatial region of a city that is visited by a large number of citizens' routes (e.g., ring roads, central avenues, or parks). For example, Figure 1 shows two DTAs, each one comprising 3 different subroutes. Note that these routes may have any purpose like commuting, going to the school, or shopping. As a result, the set of DTAs of a city comprise the areas that capture most of the movement of its population. The following sections describe how such DTAs can be perceived.

2.2. System Architecture. As it has been previously stated, the system follows an opportunistic MCS approach so as to detect the DTAs. Therefore, it relies on a set of participants or contributors that voluntarily accept to undertake sensing tasks.

From an architectural point of view, Figure 2 depicts that the system comprises two different elements, a thin client running in the personal or vehicle-mounted devices of the users and a back-end server.

The mobile client is in charge of detecting, at each moment, the routes of the device's holder and sending them to the central server. Due to the adopted opportunistic sensing, this task is carried out in unconscious mode; namely, the client runs in the background and opportunistically collects and delivers the routes without active involvement of user.

Next, the server, on the basis of the collected routes, composes and manages the DTAs. It is important to note that such DTA generation is undertaken in an incremental manner at the same time users cover their routes, so the system does not rely on any type of previously gathered data.

2.3. System Operation Modes. The present system supports two different modes of execution, *DTA discovery* and *DTA monitoring*. Depending on the active mode, the system focuses on a particular task and it changes from one mode to the other when certain conditions arise.

(i) Firstly, the *DTA discovery* phase is the initial mode of the system. During this state, the system focuses on generating a set of DTAs, $\mathscr{A}$, as complete and detailed as possible. To do so, the system collects all the routes from all the participants and timely composes $\mathscr{A}$ on the basis of these routes.

(ii) Once the system has been able to generate a suitable set $\mathscr{A}$ then it transitions to the *DTA monitoring* mode. In this second mode, the system focuses on controlling the evolution of certain mobility features of the DTAs in $\mathscr{A}$ to early detect potential mobility shifts. To do so, the system processes the routes of a subset of participants that allow reliably perceiving the aforementioned features. Finally, in case the system actually detects a potential shift, it moves to the initial *DTA discovery* mode in order to fully capture the mobility change in $\mathscr{A}$.

By means of these two modes, the system is able not only to detect the DTAs but also to perceive the movement inside them. Besides, for the monitoring task the system only uses part of the contributors so it does not require all the participants to report their locations all the time. For the sake of clarity, Figure 3 shows the state machine of the system. We will see the inner functionality of the system with respect to both modes in the upcoming sections.

3. DTA Discovery

During the *DTA discovery mode*, the system's components of both the thin client and the back-end server are intensively executed so as to generate an initial or refined set of DTAs $\mathscr{A}$. In more detail, the steps followed by the system are put forward next.

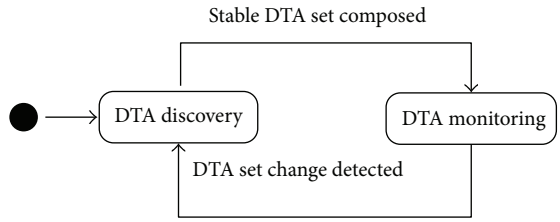


Figure 3: System's state machine where the *DTA discovery* is the initial state of the system.

3.1. Users' Routes Generation. In order to generate the routes covered by the user, both the *route composer* and the *route deliverer* components work in a collaborative manner.

3.1.1. Route Composer. As Figure 2 shows, this element is part of the mobile client running in each user's mobile device. Its key goal is to detect the current route $r(p)$ that the device's holder p is covering at each moment.

For this goal, this module only relies on the device's GPS sensor to extract the routes' raw locations. In particular, the sensor periodically provides the module with a new timestamped location l comprising the tuple $\langle x, y, t\rangle$ where $\langle x, y\rangle$ is the location in terms of latitude-longitude coordinates at instant t.

On the basis of the collected locations, the system incrementally composes the sequence of timestamped locations, $r(p)_l = \{l_1, l_2, \ldots, l_n\}$, of the ongoing route $r(p)$. This is done by a two-step procedure.

Firstly, the algorithm removes erroneous or irrelevant locations that the GPS sensor may return [7]. This is done by means of the distance-based filtering applied to each new location l_{new} described in [8] that allows performing this cleaning in real time.

Secondly, if l_{new} is not discarded by the aforementioned filter then it is appended to $r(p)_l$ by following a spatiotemporal gap identification approach. To do so, a maximum distance, $\mathcal{S}_{\text{gap}}$, and time interval, $\mathcal{T}_{\text{gap}}$, between two consecutive locations are defined. If the spatial or temporal distance between l_{new} and the last location in $r(p)_l$, l_{last}, exceeds $\mathcal{S}_{\text{gap}}$ or $\mathcal{T}_{\text{gap}}$, it identifies l_{last} as ending point d_r of the ongoing route $r(p)$ and l_{new} as the starting point o_r of a new $r(p)$. Otherwise, l_{new} is appended to $r(p)_l$ as the new last location.

Finally, the current sequence of the ongoing route, $r(p)_l$ (and the one of the just-completed route, $r(p)_l^{\text{ended}}$, if any), is sent to the *route deliverer* component.

3.1.2. Route Deliverer. This element is in charge of controlling the routes sequences that are delivered to the central server from the mobile client.

When the system runs in the *DTA discovery* mode, this component only processes each completed route sequence $r(p)_l^{\text{ended}}$. In particular, it carries out two different tasks: (1) it delivers the route to the central server and (2) it stores such route in the local *personal routes repository* within the mobile client (see Figure 2). Such repository keeps the last routes covered by the user. As we will see later, this repository is instrumental when the system runs in the *DTA monitoring* mode.

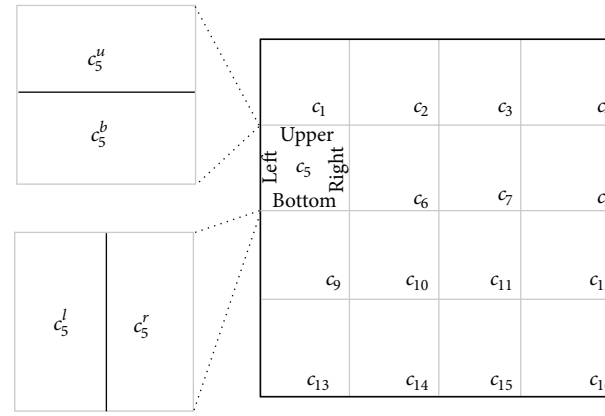


Figure 4: Graphical representation of the set $\mathcal{C}(c_5)^{\text{sub}}$.

3.2. DTA Generation. Once the server receives the users' routes, it makes up the DTAs by means of two of its modules, the *DTA composer* and the *DTA aggregator*.

3.2.1. DTA Composer. This module of the back-end server is responsible for actually detecting the new DTAs of the city. Hence, it receives all the completed routes from all the users when the system runs in *DTA discovery*.

Bearing in mind Definition 5, we can regard a DTA as a spatial region exhibiting a high density of routes from many different users. Consequently, this module adopts an approach based on computing certain features of the incoming routes with respect to a predefined spatial partition so as to calculate the density of routes in each part of the city and, thus, uncover its DTAs.

To do so, *DTA composer* firstly divides the whole spatial region of the city under study into squared cells of the same size, $\mathcal{C}$. In turn, each cell $c_i \in \mathcal{C}$ is further divided into four *subcells* each one covering a different spatial region inside c_i; namely, $\mathcal{C}(c_i)^{\text{sub}} = \{c_i^u, c_i^b, c_i^l, c_i^r\}$. As Figure 4 shows, these subcells split the cell regarding the different manners a route can traverse it.

On the basis of this multilevel spatial partition, the module calculates the route density in each cell and subcell by the procedure shown in Algorithm 1. This algorithm is launched whenever a new route $r(p)_l^{\text{ended}}$ from any user is received.

First of all, the algorithm gets the timestamp at which the incoming route started (line 2 of Algorithm 1). Next, it maps the sequence of timestamped locations of the incoming route to the multilayered spatial partition described above. As a result, the route is translated into a sequence of cells, $r(p)_c^{\text{ended}}$ (line 3). As Table 1 shows, each cell $c \in r(p)_c^{\text{ended}}$ comprises five movement attributes related to $r(p)_l^{\text{ended}}$.

These five attributes can be computed using simple mathematics and computational geometry [9]. In that sense, Figure 5 shows an example of how a route comprising six

Input: A completed route's abstraction $r(p)_l^{\text{ended}}$
Output: Set of new generated DTAs $\mathscr{A}_{\text{new}}$
(1) $\mathscr{A}_{\text{new}} \leftarrow \emptyset$
(2) $t \leftarrow r(p)_l^{\text{ended}}.t_{\text{init}}$
(3) $r(p)_c^{\text{ended}} \leftarrow map(\mathscr{C}, r(p)_l^{\text{ended}})$
(4) **for each** $c \in r(p)_c^{\text{ended}}$ **do**
(5) **if** $c.speed \leq speed_{walking}$ **then**
(6) $c_{\text{tr}} \leftarrow \mathscr{C}_{\text{tr}}^{w}.get(c)$
(7) **else**
(8) $c_{\text{tr}} \leftarrow \mathscr{C}_{\text{tr}}^{\text{nw}}.get(c)$
(9) **if** $update\_cell(c_{\text{tr}}, c, t) = true$ **then**
(10) $dta_{\text{new}} \leftarrow$ new DTA(c_{tr})
(11) $\mathscr{A}_{\text{new}} \leftarrow \mathscr{A}_{\text{new}} \cup dta_{\text{new}}$
(12) **else if** $c.c_{\text{sub}} \neq \emptyset$ **then**
(13) $c_{\text{tr}}^{\text{sub}} \leftarrow c_{\text{tr}}.get\_subcell(c.c_{\text{sub}})$
(14) **if** $update\_cell(c_{\text{tr}}^{\text{sub}}, c.c_{\text{sub}}, t) = true$ **then**
(15) $dta_{\text{new}} \leftarrow$ new DTA(c_{tr})
(16) $\mathscr{A}_{\text{new}} \leftarrow \mathscr{A}_{\text{new}} \cup dta_{\text{new}}$
(17) **return** $\mathscr{A}_{\text{new}}$
(18) **function** $update\_cell(c_{\text{tr}}, c, t)$
(19) $c_{\text{tr}}.users \leftarrow c_{\text{stats}}.users \cup p$
(20) $c_{\text{tr}}.n\_routes$ ++
(21) $c_{\text{tr}}.get\_time(t).get\_side(c.side_{\text{in}}).routes_{\text{in}}$ ++
(22) $c_{\text{tr}}.get\_time(t).get\_side(c.side_{\text{out}}).routes_{\text{out}}$ ++
(23) $c_{\text{tr}}.get\_time(t).speed \leftarrow inc\_avg(c_{\text{stats}}.n\_routes, c_{\text{stats}}.get\_hour(h).speed, c.speed)$
$c_{\text{tr}}.route\_len \leftarrow c_{\text{tr}}.route\_len + c.route\_len$
(24) $a \leftarrow c.side\_length \times c.side\_length$
(25) **if** $c_{\text{tr}}.route\_len/a \geq \delta_{\min} \wedge |c_{\text{tr}}.users| \geq \mu_{\min}$ **then**
(26) **return** true
(27) **return** false

Algorithm 1: DTA detection algorithm.

Table 1: Attributes for each cell c in a cell-based route sequence $r(p)_c^{\text{ended}}$.

Attribute	Meaning
side_length	Length of each side of the cell. This value is the same for all $c \in \mathscr{C}$
speed	Average speed of $r(p)_l^{\text{end}}$ in c
route_len	Spatial length of $r(p)_l^{\text{end}}$ in c
$side_{\text{in}}$	Side of the cell at which the route got into it (*upper, bottom, right,* or *left*)
$side_{\text{out}}$	Outgoing side of the route for c
c_{sub}	Subcell of $\mathscr{C}(c)^{\text{sub}}$ fully covered by $r(p)_{\text{abs}}^{\text{ended}}$ in c

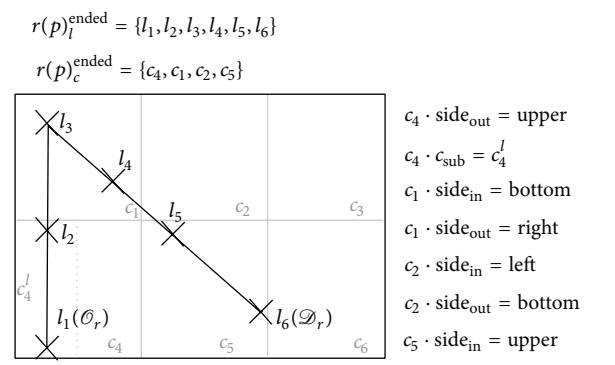


Figure 5: Example mapping of a route and the attributes of each of its cells.

timestamped locations is mapped to a sequence of four different cells $r(p)_c^{\text{ended}} = \{c_4, c_1, c_2, c_5\}$ along with some attributes of each cell. In that sense, we can see that not all the cells include the c_{sub} attribute (see Table 1). This is because, in many cases, the subroute in a cell covers more than one of the subcells. For example, in Figure 5 the subroute of $r(p)_l$ in cell c_4 is fully covered by the subcell c_4^l whereas in cell c_1 the subroute is partially covered by the 4 subcells.

Once the mapping is completed, the algorithm uses the resulting sequence $r(p)_c^{\text{ended}}$ to update the *space partitioning stats* repository (see Figure 2).

This repository stores aggregated transit data of the space partition $\mathscr{C}$. In particular, such data is organized in two entities, $\mathscr{C}_{\text{tr}}^{w}$ and $\mathscr{C}_{\text{tr}}^{\text{nw}}$. The former stores information about routes having a low speed so that they are likely to have been covered walking whereas the latter stores information about

nonwalking routes as they exhibit higher speed representing vehicle-based routes. Both $\mathscr{C}_{\text{tr}}^{w}$ and $\mathscr{C}_{\text{tr}}^{\text{nw}}$ store for each $c \in \mathscr{C}$ the transit properties shown in Table 2.

The rationale of having two separate instances is that urban dynamics might be different depending on whether we consider movement on foot or in vehicles. For example, vehicle-based routes are constrained by the road network of the city whereas walking-based routes usually do not show so constrained displacements.

Consequently, for each cell c in $r(p)_c^{\text{ended}}$ the system updates $\mathscr{C}_{\text{stats}}^{w}$ or $\mathscr{C}_{\text{stats}}^{\text{nw}}$ depending on its speed attribute and a domain-dependant threshold $speed_{walking}$ (lines 5–8 of Algorithm 1). This update process is carried out by the *update_cell* function (line 9). This function takes a cell from $r(p)_c^{\text{ended}}$, c, and its associated element in the repository c_{tr}. As a result, it returns a Boolean value indicating whether the historical transit data in c_{tr} allows classifying it as a DTA. If that is the case, such entity gives rise to a new DTA (line 10) that is eventually added to the set of new DTAs (line 11). In this generation, the system removes the *users* attribute to keep DTAs anonymized. Otherwise, the system repeats the same process but this time with the subcell of c, c_{sub} (see Table 1) (lines 12–16 in Algorithm 1).

This way, the algorithm follows a top-down approach so as to generate the DTAs, as it firstly tries to generate cell-based DTAs. If that is not possible, it focuses on detecting smaller DTAs with subcell granularity. Finally, the algorithm returns the set of new DTAs generated on the basis of the incoming route (line 18).

Concerning the inner functionality of the *update_cell* function (lines 18–27), it firstly updates c_{tr} with the attributes of its associated cell c by simple or incremental addition (lines 19–23). In that sense, attributes speed, $\text{routes}_{\text{in}}$, and $\text{routes}_{\text{out}}$ are disaggregated by a temporal criterion $\mathscr{T}_{\text{crit}}$. This way, it is possible to know the speed and information about incoming and outgoing sides in a particular time interval $t \in \mathscr{T}_{\text{crit}}$ with predefined granularity. As a matter of fact, if an hour granularity is chosen then $\mathscr{T}_{\text{crit}}$ is defined as an array $\mathscr{T}_{\text{crit}} = \{0, 1, \ldots, 23\}$.

Once c_{tr} has been updated, *update_cell* also detects whether it actually can give rise to a DTA (lines 25-26). For that goal, the function checks two features of c_{tr}, (1) its density of routes and (2) the number of users that have visited at least once the cell.

The first one can be calculated with respect to the total length of historical subroutes within c_{tr} and its geometric area. Since we are using square cells, we can easily compute such area as the cell's side length squared (line 24). At the end, if such density and number of users are over two domain-dependant thresholds, δ_{min} and μ_{min}, the function concludes that c_{tr} actually represents a DTA, thus returning the Boolean value *true*. Otherwise, *false* is returned.

3.2.2. DTA Aggregator. The resulting set, $\mathscr{A}_{\text{new}}$, from the previous algorithm is directly delivered to this module (see Figure 2). In that sense, $\mathscr{A}_{\text{new}}$ comprises DTAs at a cell or subcell granularity. However, real DTAs can cover spatial areas larger than the predefined size of a cell. For that reason, the *DTA aggregator* element applies a fusion procedure to the resulting DTAs to merge such areas.

The key idea of this procedure is that two closed DTAs can be merged together, creating a larger DTA, if the transit information they represent is strongly related. In our setting, we infer that two DTAs represent the same transit flow if people move at a similar speed and direction in both areas. More specifically, we distinguish between two types of similarities among DTAs, namely, the following:

(i) Parallel-transit similarity, sim_{par}: this similarity arises when the subroutes of the DTAs have a very similar direction and speed.

(ii) Common-transit similarity, sim_{com}: this similarity occurs when the DTAs have a certain number of common subroutes covering them.

For example, the two DTAs in Figure 6(a) comprise quite different transit flows in terms of both speed (one of them is mostly covered by vehicle-based routes whereas the other one has more walking routes) and direction (the routes in each DTA go in reverse with respect to the other). However, Figure 6(b) shows a parallel-transit similarity between the two DTAs whereas Figure 6(c) depicts a common-transit similarity. Therefore, the DTAs in these two last situations could be merged to make up a new and larger DTA.

In order to compute each similarity we made use of the number of incoming and outgoing subroutes that each DTA comprises (see Table 2). Thus, given two DTAs a and b their parallel similarity $\text{sim}_{\text{par}}(a, b) \in [0, 1]$ is calculated as follows:

$$\text{dissim}_{\text{par}}^{\text{in}}(a, b) = \frac{\sum_s \left| a.\text{routes}_{\text{in}}^{s}/a.n\_\text{routes} - b.\text{routes}_{\text{in}}^{s}/b.n\_\text{routes} \right|}{4}, \tag{1}$$

$$\text{dissim}_{\text{par}}^{\text{out}}(a, b) = \frac{\sum_s \left| a.\text{routes}_{\text{out}}^{s}/a.n\_\text{routes} - b.\text{routes}_{\text{out}}^{s}/b.n\_\text{routes} \right|}{4}, \tag{2}$$

$$\text{sim}_{\text{par}}(a, b) = 1 - \left(\frac{\text{dissim}_{\text{par}}^{\text{in}}(a, b) + \text{dissim}_{\text{par}}^{\text{out}}(a, b)}{2} \right). \tag{3}$$

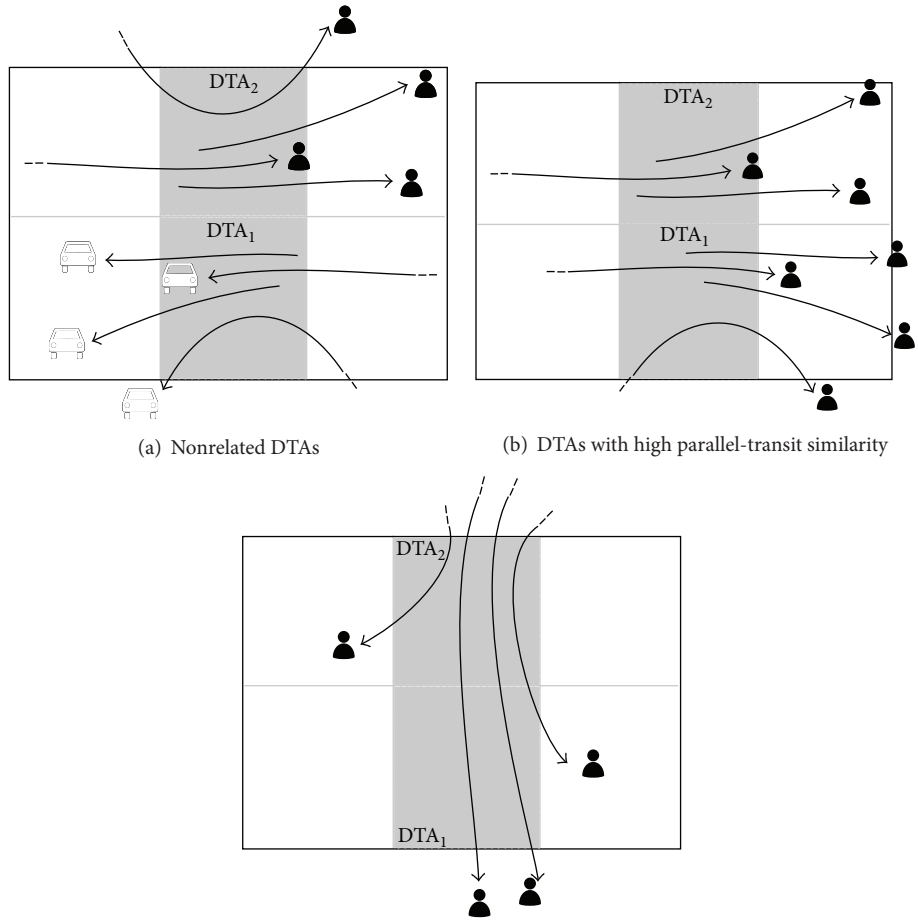
Equations (1) and (2) calculate the dissimilarity between the two rates of incoming and outgoing subroutes for each of the four sides of a DTA. Finally, (3) aggregates both rate differences to generate the final parallel similarity.

Furthermore, the common-transit similarity $\text{sim}_{\text{com}}(a, b) \in [0, 1]$ is calculated as follows:

$$\text{sim}_{\text{com}}(a, b) = 1 - \left(\frac{\left| a.\text{routes}_{\text{in}}^{s} - b.\text{routes}_{\text{out}}^{s'} \right|}{2} + \frac{\left| a.\text{routes}_{\text{out}}^{s} - b.\text{routes}_{\text{in}}^{s'} \right|}{2} \right), \tag{4}$$

Table 2: Attributes for each cell c_{tr} in $\mathscr{C}_{tr}^{w}$ and $\mathscr{C}_{tr}^{nw}$ and DTA in $\mathscr{A}$ (except *users*).

Attribute	Meaning
type	Type of movement within the cell, namely, *walking* or *nonwalking*
users	Number of users who have at least one route covering the cell c
n_routes	Number of different historical subroutes that have traversed c
route_len	Total length of all the subroutes that have covered c
speed	Average speed of the routes while traversing c
$routes_{in}^{s}$	Number of subroutes that have gone into c through a particular side s of the cell ($s \in \{upper|bottom|left|right\}$). For example, $routes_{in}^{left}$ informs about how many routes have covered c by getting into its left side
$routes_{out}^{s}$	Number of subroutes that have left c through each side of the cell



(a) Nonrelated DTAs

(b) DTAs with high parallel-transit similarity

(c) DTAs with high common-transit similarity

Figure 6: Examples of DTAs similarities. Each DTA is shown as a grey square.

where s is the common side between a and b from a perspective whereas s' is the adjacent side from b point of view. For example, in Figure 6(a) s = bottom (for DTA_2) whereas s' = upper for (DTA_1). As we can see, sim_{com} basically measures the subroutes that actually move between the two DTAs under consideration.

All in all, Algorithm 2 shows the mechanism applied by the *DTA composer* so as to fuse DTAs. Basically, this algorithm takes a set of DTAs to be merged, $\mathscr{A}_{target}$. Then, each pair of adjacent DTAs is compared to measure its parallel (lines 4–7) and common (lines 9–12) similarity. In that sense, only DTAs with the same *type* attribute (see Table 2) are compared. This way, we ensure that both contain similar routes in terms of speed. It is also important to note that both similarities are calculated with respect to the time criterion $\mathscr{T}_{crit}$. Thus, if any of these similarities is above its associated threshold sim_{min}, it means that both DTAs comprise a similar or common human dynamics for different time periods. As a result, the two DTAs can be merged.

Input: Set of DTAs $\mathscr{A}_{\text{target}}$
Output: Set of merged DTAs $\mathscr{A}_{\text{target}}^{\text{merged}}$
(1) $\mathscr{A}_{\text{target}}^{\text{merged}} \leftarrow \emptyset$
(2) **for each** $a \in A_{\text{target}}$ **do**
(3) $\quad \mathscr{A}_{\text{adj}}^{a} \leftarrow find\_adj\_DTAs(A_{\text{new}}, a)$
(4) $\quad$ **for each** $a_{\text{adj}} \in A_{\text{adj}}^{a}$ **do**
(5) $\quad\quad$ **if** $\sum_{t \in \mathscr{T}_{\text{crit}}} \text{sim}_{\text{par}}(a, a_{\text{adj}}, t)/|\mathscr{T}_{\text{crit}}| \geq \text{sim}_{\text{min}}$ **then**
(6) $\quad\quad\quad a \leftarrow mergeDTA(a, a_{\text{adj}})$
(7) $\quad\quad\quad A_{\text{target}} \leftarrow A_{\text{target}} - a_{\text{adj}}$
(8) $\quad \mathscr{A}_{adj}^{a} \leftarrow find\_adj\_DTAs(A_{\text{new}}, a)$
(9) $\quad$ **for each** $a_{\text{adj}} \in A_{\text{adj}}^{a}$ **do**
(10) $\quad\quad$ **if** $\sum_{t \in \mathscr{T}_{\text{crit}}} \text{sim}_{\text{com}}(a, a_{\text{adj}}, t)/|\mathscr{T}_{\text{crit}}| \geq \text{sim}_{\text{min}}$ **then**
(11) $\quad\quad\quad a \leftarrow mergeDTA(a, a_{\text{adj}})$
(12) $\quad\quad\quad A_{\text{target}} \leftarrow A_{\text{target}} - a_{\text{adj}}$
(13) $\quad \mathscr{A}_{\text{target}}^{\text{merged}} \leftarrow \mathscr{A}_{\text{target}}^{\text{merged}} \cup a$
(14) **return** $\mathscr{A}_{\text{target}}^{\text{merged}}$

Algorithm 2: DTA fusion algorithm.

Finally, Algorithm 2 is executed by the *DTA aggregator* by two different manners. In the first one, the algorithm is automatically launched when a new set of DTAs is delivered from the *DTA composer*. Then, the resulting set of fused DTA, $\mathscr{A}_{\text{target}}^{\text{merged}}$, is appended to the global set of DTAs, $\mathscr{A}$, in the *DTA global repository* (see Figure 2). Additionally, since this first type of execution only fuses the DTAs generated due to a single route, the fusion mechanism is also periodically launched over the DTA set, $\mathscr{A}$, so as to detect correlated DTAs in the whole city under study.

3.3. DTA Discovery-DTA Monitoring Transition. At the same time the server composes the DTAs it controls their state so as to decide whether the system remains in the *DTA discovery* mode or it can move to the *DTA monitoring* phase.

In that sense, the system should transition from the discovery to the monitoring stage if a stable set of DTAs has been composed. In that sense, the *DTA global manager* module implements this decision process.

3.3.1. DTA Global Manager. This module defines a sampling time period $\mathscr{T}_s$ and for each period calculates the number of new DTAs n_{dta} and the number of received routes from all the users n_{routes}. If the ratio $n_{\text{dta}}/n_{\text{routes}}$ is below a decision threshold dt_{min} then it means that the system has not composed many new DTAs with respect to the incoming routes. Consequently, the module infers that the system has reached a consistent set of DTAs and eventually transitions to the *DTA monitoring* as it is depicted in Figure 3.

During this transition, the *DTA global manager* distributes the set $\mathscr{A}$ of DTAs uncovered by the server among all the contributors. In the client side, the *DTA Local Manager* receives such set and stores it in the *DTA local repository* (see Figure 2). As we will see in the next section, locally storing this data is of great help in the *DTA monitoring* mode.

On the whole, the system during the *DTA discovery* mode composes a set of DTAs by means of an approach that combines a multilayered partition of the space and its posterior fusion to come up with a reliable set of DTAs. Next, during *DTA monitoring* stage the system focuses on controlling the fact that the current set of DTAs actually represent the dynamics of the city under study.

4. DTA Monitoring

In the *DTA monitoring* mode, the system focuses on selecting a subset of users and uses their reported routes to compare the current state of the human transit in the city with respect to the one represented by the DTAs to see whether any discrepancy exists. This allows deactivating certain users and, thus, avoiding their continuous contribution to the system with the consequent resources saving.

As a result, the behaviour of the system changes with respect to the one described in the previous section so as to adapt to this new goal. In particular, the steps of the system in this mode are described next.

4.1. User Subset Selection. The first task the system does when it starts to operate in monitoring mode is to select the target set of monitoring users. This selective process is undertaken by the *DTA global manager* component.

4.1.1. DTA Global Manager. For the aforementioned goal, this module uses the sampling time period t_s and selects, for each period, the subset of users providing the best coverage of the detected DTAs. This selection process follows the following steps.

(1) For each DTA in $\mathscr{A}$, the module asks the mobile clients to report their *availability* to visit this DTA within the current sampling period.

(2) Among all the users reporting an availability score over a domain-dependant threshold av_{min}, it chooses the top k users according to this value.

(3) These top k users are committed to report their ongoing routes to the server during the current sampling interval.

4.1.2. DTA Local Manager. This module in the mobile client is in charge of calculating the *availability* score used by the aforementioned selection process.

To do so, each time the mobile clients switch to the *DTA monitoring* mode, this module reads the historical routes of the user stored in the *personal routes repository*. Then, it counts the number of visits to each DTA stored in its *DTA local repository*, for each time period in $\mathcal{T}_{\text{crit}}$. Then, it removes all these historical routes from the repository. This way, only the routes covered by the user during the previous *DTA discovery* cycle are used to generate the visit statistics. This avoids the usage of rather deprecated routes.

With this information, the module can easily calculate probability of visiting a DTA, *dta,* at particular time interval $t \in \mathcal{T}_{\text{crit}}$ as

$$p\,(dta)_t = \frac{nv\,(dta)_t}{nr_t}, \tag{5}$$

where $nv(dta)_t$ is the number of visits to *dta* during t and nr_t is the number of routes that started during t. On the basis of such probability the module calculates the availability to visit a DTA at a particular time interval t, $\text{av}(dta)_t \in [0, 1]$, as follows:

$$\text{av}\,(dta)_t = \text{contribution}_{\text{factor}} \times p\,(dta)_t\,, \tag{6}$$

where $\text{contribution}_{\text{factor}}$ is a corrective factor to avoid the fact that a user contributes to the monitoring state during too many consecutive time intervals. To do so, the module counts the number of times the user has been selected by the server during the last $nc_{\max}$ instants, nc_m, and computes such factor as follows:

$$\text{contribution}_{\text{factor}} = 1 - \frac{nc_m}{nc_{\max}}. \tag{7}$$

Finally, $\text{av}(dta)_t$ for each DTA is locally stored and sent to the central server in each selection process.

4.2. User Subset Routes Generation. As in the previous mode, this task is performed by the same two modules in the client side, *route composer* and *route deliverer* (see Figure 2).

4.2.1. Route Composer. This component in the client has the same functionality in both operational modes. Hence, it just composes the sequences of timestamped locations of the user's routes and sends them to the *route deliverer* as it has been described in Section 3.1.

4.2.2. Route Deliverer. The behaviour of this element changes slightly under the monitoring stage. In particular, it now forwards to the central server the ongoing route's sequence $r(p)_l$ instead of the completed routes. Therefore, each time the *route composer* appends a new location to such sequence, this module immediately delivers the new version to the server. This way, the central server is informed of how the users move in real time under this operation mode.

4.3. DTA Change Detection. The key tasks in this operation mode are to perceive potential changes of the urban dynamics in the target city. This process involves the *DTA change detector* component.

4.3.1. DTA Change Detector. This module processes all the ongoing routes reported by the subset of users so as to extract the current movement features inside each DTA in terms of direction and speed. Then, it periodically compares such current values with the historical ones stored in the *DTA global repository*. Algorithm 3 shows this process in more detail.

In short, the algorithm firstly maps each route to its cell-based representation (line 3). After that, it updates the current state of each DTA covered by the route's cells (lines 5–9).

Next, the similarity between the current and the historical state of each DTA in terms of transit direction and speed is measured (lines 11–13). In that sense, we use the parallel similarity for the directional features (see Section 3.2.2). This allows detecting whether the subroutes currently visiting the DTA enter and exit it in the same way and speed that is reflected in the historical data. If that is not the case, it means that people or vehicles are moving in an unusual direction or speed. This might be originated due to planned events (e.g., road works) or unplanned ones (e.g., sudden riots or car accidents).

Consequently, if either the speed or direction similarity is below a predefined threshold then the algorithm infers that the human dynamics inside the DTA under review might have moved with respect to its default state. As a result, it is appended to the list of DTAs with potentially changed DTAs, $\mathcal{A}_{\text{shift}}$ (line 16).

4.3.2. Low DTA Coverage Problem. It might occur that the selection process described in Section 4.1 is not capable of configuring a suitable set of top-k users for one or more DTAs. In order to cope with this limitation, the system indirectly calculates the potential dissimilarity of these uncovered DTAs by using the K-nearest neighbours (KNN) method.

In more detail, for each DTA with low coverage, dta_{lc}, the *DTA change detector* gets the speed and direction similarity, sim_{dir}, $\text{sim}_{\text{speed}}$ of its k closest DTAs with suitable coverage, $\mathcal{A}(\text{dta})_k$. Then, it extrapolates such measurements to infer the associated similarities of dta_{lc} as follows:

$$\begin{aligned} \text{sim}_{\text{dir}} &= \frac{\sum_{i \in |k|} \lambda^i_{\text{dist}} \times \text{sim}^i_{\text{dir}}}{\sum_{i \in |k|} \lambda^i_{\text{dist}}}, \\ \text{sim}_{\text{speed}} &= \frac{\sum_{i \in |k|} \lambda^i_{\text{dist}} \times \text{sim}^i_{\text{speed}}}{\sum_{i \in |k|} \lambda^i_{\text{dist}}}, \end{aligned} \tag{8}$$

where λ^i_{dist} is a decay factor proportional to the distance between the ith DTA in $\mathcal{A}(\text{dta})_k$ and dta_{lc} whereas $\text{sim}^i_{\text{dir}}$ and $\text{sim}^i_{\text{dir}}$ are the direction and speed similarities of the ith DTA in $\mathcal{A}(\text{dta})_k$.

Lastly, these two inferred values are used like the ones calculated by Algorithm 3 for the anomaly detection in DTAs described above.

Input: Ongoing route $r(p)_l$, DTAs current features $\mathscr{A}_{\text{curr}}$, DTA historic features $\mathscr{A}$
Output: Set of DTAs with meaningful shifts $\mathscr{A}_{\text{shift}}$
(1) $\mathscr{A}_{\text{shift}} \leftarrow \emptyset$
(2) $t \leftarrow r(p)_c.t_{\text{init}}$
(3) $r(p)_c \leftarrow map(\mathscr{C}, r(p)_{\text{seq}})$
(4) **for each** $c \in r(p)_c$ **do**
(5) $\quad dta_{\text{curr}} \leftarrow \mathscr{A}_{\text{curr}}.get(c)$
(6) $\quad dta_{\text{curr}}.n\_routes$ ++
(7) $\quad dta_{\text{curr}}.get\_time(t).get\_side(c.side_{\text{in}}).routes_{\text{in}}$ ++
(8) $\quad dta_{\text{curr}}.get\_time(t).get\_side(c.side_{\text{out}}).routes_{\text{out}}$ ++
(9) $\quad dta_{\text{curr}}.get\_time(t).speed \leftarrow inc\_avg(dta_{\text{curr}}.n\_routes,$
(10) $\qquad dta_{\text{curr}}.get\_hour(h).speedc.speed)$
(11) $\quad dta_{\text{hist}} \leftarrow \mathscr{A}.get(c)$
(12) $\quad \text{sim}_{\text{dir}} \leftarrow \sum_{t \in \mathscr{T}_{\text{crit}}} \text{sim}_{\text{par}}(dta_{\text{hist}}, dta_{\text{curr}}, t)/|\mathscr{T}_{\text{crit}}|$
(13) $\quad$ **if** $\text{sim}_{\text{dir}} \leq \text{sim}_{\text{min}}$ **then**
(14) $\quad\quad \mathscr{A}_{\text{shift}} \leftarrow \mathscr{A}_{\text{shift}} \cup dta_{\text{hist}}$
(15) $\quad \text{sim}_{\text{speed}} \leftarrow |(dta_{\text{hist}}.speed - dta_{\text{curr}}.speed)/\max(dta_{\text{hist}}.speed, dta_{\text{curr}}.speed)|$
(16) $\quad$ **if** $\text{sim}_{\text{speed}} \leq \text{sim}_{\text{min}}$ **then**
(17) $\quad\quad \mathscr{A}_{\text{shift}} \leftarrow \mathscr{A}_{\text{shift}} \cup dta_{\text{hist}}$
(18) **return** $\mathscr{A}_{\text{shift}}$

Algorithm 3: DTA monitoring algorithm.

4.4. DTA Monitoring-DTA Discovery Transition. If a DTA is included in $\mathscr{A}_{\text{shift}}$ during t_{shift} consecutive sampling periods, the module concludes that the flow of people/vehicles within such an area has actually changed. As a result, it alerts the *DTA global manager* (see Figure 2).

When this module receives such an alert, it automatically starts the transition of the whole system to go back to the initial *DTA discovery* mode. The goal in this case is to use again all the contributors' routes so as to accurately confirm the preliminary change detected by the *DTA change detector*.

4.5. Exploitation of the DTA Global Repository. As Figure 2 depicts, the whole set of DTAs that the system generates and updates in its two-operation-mode cycle is stored in the *DTA global repository*.

Such a repository is exposed to third-party location-based services that can make use of it. In that sense, it is important to recall that the transit information that each DTA contains and, thus, is used by the stakeholders is completely anonymized. According to Table 2, each DTA informs of the average velocity of usual directions of routes visiting such area but it does not comprise any type of personal information of the users. Consequently, it prevents a malicious third-party service from getting access to personal details of the contributors by means of the public DTAs.

5. Evaluation

In order to state a comprehensive view of our proposal, we evaluated it on a real-world dataset. Besides, we compared our proposal with an existing approach to perceive abnormal movements in the mobility mining domain.

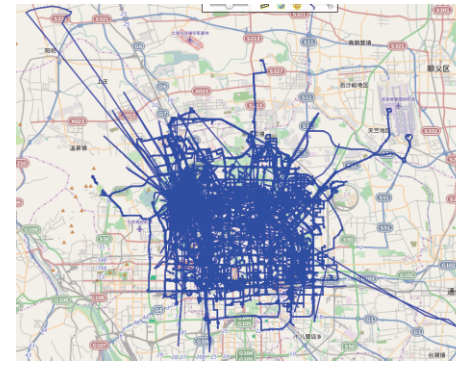


Figure 7: Digital trace of the GL dataset.

5.1. Experiment Setup

5.1.1. Dataset. In order to test our system we have used the *GeoLife dataset* (GL) [10], a public collection of human trajectories produced by 178 users carrying different GPS feeds in a period of over three years. Figure 7 depicts its trace that falls into the square of latitude 40.17 to 39.86 and longitude 116.14 to 116.34 covering a large area of Beijing city (China). More details about the dataset are provided in Table 3.

5.1.2. Settings. Table 4 summarizes the default configuration of the system for the evaluation.

Furthermore, a week-hour granularity has been chosen for $\mathscr{T}_{\text{crit}}$ so it takes the form of an array $\mathscr{T}_{\text{crit}} = \{0, 1, 2, \ldots, 167\}$ representing the 168 hours within a week. Consequently, the transit features of the DTAs are disaggregated with respect to such an array.

Table 3: GL dataset general information.

Users	Locations	Routes	Time period
178	23667828	17621	2007-04 → 2011-10

Table 4: System default configuration.

Parameter	Module	Value
$\mathcal{S}_{\text{gap}}$	*Route composer*	1000 m
$\mathcal{T}_{\text{gap}}$	*Route composer*	15 m
$speed_{walking}$	*DTA composer*	1.2 m/s
$\delta_{\min}$	*DTA composer*	2 m/m^2
$\mu_{\min}$	*DTA composer*	5
$\text{sim}_{\min}$	*DTA aggregator*	0.8
t_{shift}	*DTA change detector*	3
$av_{\min}$	*DTA global manager*	0.8
k	*DTA global manager*	10
$dt_{\min}$	*DTA global manager*	0.2
$nc_{\max}$	*DTA global manager*	4
t_s	*DTA global manager*	360 s

5.2. Effect of the Cell Size. When a grill-based partitioning is used to compose the representation of spatiotemporal routes, the cell size is one of the parameters that has an important impact on the whole system. In our case, this size also indicates the default size of the DTAs. As a result, Figure 8 shows, for different cell sizes, the number of DTAs composed by the system and the number of times the system detected a shift in these DTAs.

As we can see, setting a small cell size implies the generation of a large number of DTAs. This allows representing the transit information of the city with fine granularity. For example, Figure 8 shows that when a 100 m cell is used, the system generated 681 DTAs.

In certain scenarios, such large number of areas could not be convenient as stakeholders are more interested in perceiving the movement within a city in a more general way. In that sense, increasing the cell size involves, unsurprisingly, the generation of a reduced number of DTAs and, thus, a coarse-grained transit perception.

However, as Figure 8 also depicts, this increment comes with a remarkable side effect; larger DTAs become more *sensitive* to changes. Since they cover a large spatial region, they are also more likely to suffer movement shifts of its inner routes. For instance, according to Figure 8, given 3000 m cells, the system is only composed of 41 DTAs, but, on the contrary, the number of average number of changes per area was 123. In that sense, a large number of DTA changes imply that the system transitions from one operation mode to the other many times. In large deployments this can imply a large usage of resources from both the server's and contributors' point of view.

All in all, considering the existing trade-off among cell size, providing DTA granularity, and number of DTA changes, providing mode-transition *stability*, Figure 8 depicts that the configuration providing an acceptable granularity and stability is the one with a cell size of 1000 m. Given this size, 270 DTAs were generated where each one changed, on average, 69 times.

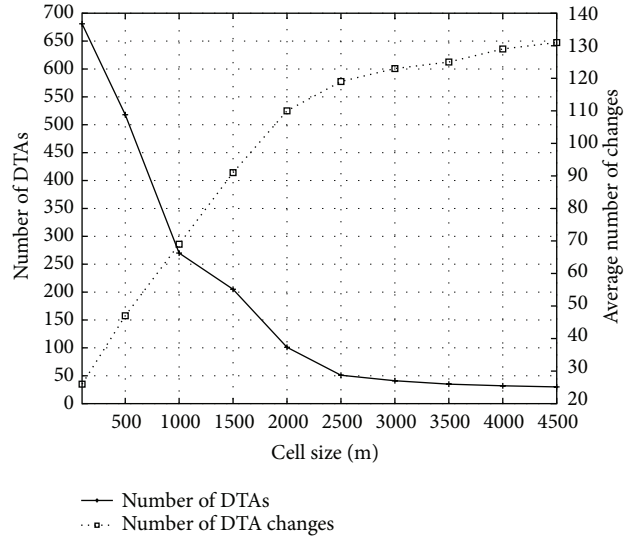


Figure 8: Number of DTAs and average number of changes per DTA with respect to the cell size of $\mathcal{C}$.

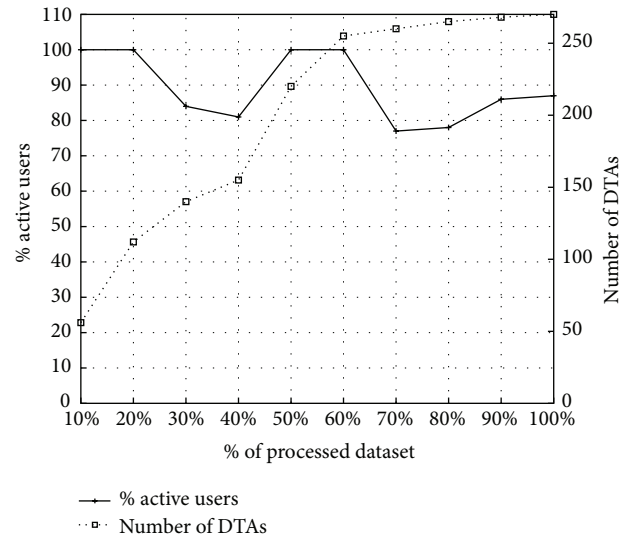


Figure 9: Percentage of active users and number of DTAs as the system proceeded.

5.3. System Evolution. One of the key features of the proposed system is its capability to compose the DTAs at the same time users are covering their routes without relying on any type of offline or previous data analytic stage. Therefore, Figure 9 shows the evolution of the system while it processed the GL dataset in terms of number of generated DTAs and percentage of active users reporting their routes at each moment.

According to this figure, we see that the system was able to reduce the number of active users about 20% with respect to the total number of contributors. Regarding the generation of DTAs, the system steadily generates them. Once

Table 5: Precision of DTA direction change detection with respect to a spatial distance approach.

	DSSIM				
	0.2	0.4	0.6	0.8	1.0
System precision	0.21	0.32	0.43	0.74	0.88

it processed 60% of the dataset, the DTA generation was reduced in meaningful terms. At this point, the system was able to compose a map of DTA detailed enough to capture the whole urban dynamics represented by the contributors. As a result, a reduction of the number of active users is also perceived because the system spends more time in *DTA monitoring* mode.

On the whole, according to the stated results, the generation of the DTA-based model in real time is feasible in terms of accuracy. However, it is true that the time required by the system to compose such model could be a drawback when a fast detection of the whole model is required.

5.4. DTA Change Detection Accuracy. Finally, we also analysed the accuracy of the approach when it comes to detecting mobility shifts in a DTA. Since such changes are based on speed and directional features, we have compared the capability of the system to detect variations of these two features with respect to two different existing solutions in the spatiotemporal data mining field.

5.4.1. Direction-Based Change Accuracy. Regarding direction-based changes, we have compared our proposal with the DSSIM function [11]. This method measures the dissimilarity between two spatiotemporal trajectories during a time period $[t_1, t_n]$ as follows:

$$\text{DISSIM}(R, S) = \int_{t_1}^{t_n} D(R(t), S(t))\, dt, \tag{9}$$

where R and S are the two trajectories to compute and D is the Euclidean distance norm.

Given such a metric, we measured the average distance between the subroutes visiting each DTA in a particular sampling interval and the ones comprising the historical information of the DTA. The idea is that if long distances were detected then the direction of the current routes visiting the DTA and the historical ones is different.

Therefore, we correlated such average distances with the number of times the system perceived a transit change in a DTA so as to study whether a DTA change detection actually represents long dissimilarities between historical and current routes. The results of this study are shown in Table 5.

This table represents the DSSIM distances normalized with respect to the size of the DTA. Therefore, DSSIM values close to 1 indicate that there were large differences between the historical and the current routes in a DTA whereas smaller values stand for higher levels of similarity. In the light of these values, Table 5 shows the precision of the system to detect DTA changes. Such measurement is computed as follows:

Table 6: Precision of the DTA speed change detection with respect to [6].

	% routes with speed alert within a DTA					
	10	20	30	40	50	60
System precision	0.21	0.32	0.76	0.81	0.91	0.93

$$\text{Precision} = \frac{\#\text{DTA changes}}{\#\text{DTA changes} + \#\text{miss DTA changes}}, \tag{10}$$

where #DTA changes is the number of times the system detected a change for DTAs having the DSSIM values under consideration whereas #miss DTA changes is the number of times the system did not consider that a DTA had changed given the DSSIM value under consideration.

As we can see, the system had a higher precision for large DSSIM values that indicate very evident differences between current and historical routes. In that sense, our proposal achieved acceptable precision results, about 0.8, when DSSIM ranged between 0.8 and 1.0. However, the system's precision decayed for DSSIM values below 0.6. This is because, at such point, certain differences of the routes do not imply changing their incoming or outgoing sides in the DTA which are the features used by the proposal to detect transit changes and, thus, become *invisible* to the system.

5.4.2. Speed-Based Change Accuracy. For the speed feature we followed a similar approach to the one for the direction. In this case, we used the event-based mechanism proposed in [6] to detect abnormally high or low speeds of moving objects in real time. Although it was initially designed for the maritime environment, it can be easily adapted to any kind of mobility domain.

In more detail, we counted the percentage of routes for which the aforementioned mechanism reported an abnormal speed within each DTA. Then, we calculated the precision of the system for each DTA as it was done for the direction. The comparison of these two values is shown in Table 6.

Results in this table confirm a similar behaviour of the system to the one for the direction; the larger the speed variations, the more precise the system. For example, when 60% of the routes within a DTA reported an abnormal speed, the precision of our system was 0.93. At the same time, when a small set of routes have unusual speed values then the precision of the system was affected with a remarkable decrease. For example, if the number of routes with abnormal speed moves from 30% to 20% then the system precision drops from 0.76 to 0.32. Nevertheless, the precision of the system in this case is slightly higher than the one based on direction.

5.5. Results and Conclusions. From the whole evaluation described above, we can draw up the following main conclusions.

(i) Firstly, the size of the cell has a great impact not only in terms of model granularity but also in terms of system stability. Consequently, its configuration should be carefully chosen in each deployment bearing in mind that large DTA could lead to an intensive usage of the contributors' device capabilities.

(ii) The generation from the scratch of the DTA map comes with the price of a long convergence period in case of large city deployments like the one used in this evaluation. However, this could be a minor problem in case of scenarios where it is not possible to collect a reliable dataset for offline mobility mining.

(iii) The accuracy of the system to detect routes' speed or direction changes within DTAs is slightly below that of the two approaches taken as reference, DSSIM and the event-based abnormal-speed detector [8]. Nonetheless, we should note that both approaches need the whole sequence of timestamped locations to operate. On the contrary, the present proposal only stores a few fields per DTA to compute such changes with the consequent saving of resources.

6. Related Work

In this section we review the state of the art of MCS within the mobility mining discipline with respect to our proposed solution. Moreover, due to their relationship with our proposal, we also provide a general overview of current efforts for human mobility modelling and the different techniques for regions of interest detection.

6.1. MCS-Based Mobility Mining Solutions. Several success stories demonstrate the suitability of MCS to support the development of mobility mining applications. In that sense, we can observe two major trends of MCS-based solutions.

On the one hand, an important course of action makes use of MCS for mapping activities by collecting the spatiotemporal traces of contributors [12, 13]. The idea here is to compose collaborative maps comprising not only road networks but also routes that are interesting in different domains like cycling or hiking.

On the other hand, due to the MCS potential for providing near real-time information, this sensing paradigm has been widely used for traffic monitoring. In that sense, a well-established line of work proposes the distributed architectures to keep track or predict road traffic congestions within an area by means of the mobility reports generated by the on-board units of vehicles [14, 15].

These MCS-based traffic monitoring architectures have recently profited from smartphones' sensing capabilities. As a result, now we can find solutions that combine static and vehicle-mounted and smartphone sensors to detect the road traffic in an area [16, 17].

Like the different lines of research described above, our mechanism also makes use of the MCS paradigm to uncover the transit state in a particular area of interest. However, the mobility knowledge extracted by the aforementioned solutions focuses on road traffic features whereas our solution intends to capture the human dynamics from a wider perspective as our model based on DTAs is independent of any road network topology.

6.2. Mobility Models Generation. In recent years, various works have considered the processing of spatiotemporal traces from users to extract relevant knowledge [18]. These digital breadcrumbs can be collected from several sources like location-based social networks [19], motion sensors [20], or smart cards [21]. In that sense, GPS traces have been one of the most popular datasources in this field.

In order to compose a model that represents the mobility of an area of interest, trajectory pattern mining has been widely studied, and works within this discipline can be classified into three lines of research, namely, frequent item mining, trajectory clustering, and graph-based trajectory mining [22].

However, during the last years a host of studies have put forward novel approaches to generate probabilistic models for mobility modelling that do not explicitly compose trajectory patterns. In these types of approaches, a database of historical routes is usually used to make up such models. In that sense, Bayesian networks [23], hidden Markov models [24], or Markov decision processes [25] have been some of the applied solutions.

In this case, our work provides a novel approach to represent the urban dynamics of an area by proposing an intermediate solution between trajectory pattern solutions and probabilistic modelling. For example, it does not explicitly generate individual or collective trajectory patterns but comprises information about how routes move within a DTA in terms of direction and speed. Furthermore, unlike the works cited above, our solution does not rely on any type of historical data as it composes the DTA map at the time it receives the routes.

6.3. Regions of Interest Detection. When it comes to detecting the regions of interest (ROIs), given a collection of spatiotemporal traces, a well-known approach consists of applying different types of clustering algorithms to such traces to uncover the target ROIs.

On the one hand, density-based clustering has been one of the prominent solutions in this area [26, 27]. These methods cluster using measures such as the distance between GPS points or density connectivity in a two-dimensional Cartesian space. In that sense, our proposal does not rely on any type of density-based clustering applied to trajectories.

Despite this high accuracy, density-based clustering algorithms need to keep the whole set of GPS traces so as to uncover the clusters and their complexity is exponential with respect to the available number of points. On the contrary, our system makes use of an aggregation method to compute certain features of the routes in a predefined space partitioning. This lightweight approach does not require keeping all the GPS traces in disk avoiding potential privacy leaks.

On the other hand, a different approach makes use of frequency map based spatial-temporal clustering methods

[28–31]. In this case, the area is partitioned into a fine grid of cells and assigns a weight to each cell around each GPS point based on the duration of the GPS staying at that point or its low speed. Next, a cell is considered a ROI when its weight is above a predefined threshold.

Our DTA detection mechanism also relies on a predefined spatial partitioning. However, unlike the aforementioned works, our approach does not intend to detect meaningful ROIs where people tend to remain stopped, but it centers on actually detecting ROIs (DTAs) related to the movement of people.

7. Conclusions

Opportunistic mobile crowdsensing has become a foremost parading in the mobile computing era. The endless improvement of the inner equipment of mobile phones and vehicle-mounted devices has made such a paradigm instrumental so as to uncover mobility phenomena in the urban domain.

In this context, the present work introduces a novel approach to leverage such paradigm and compose a map of DTAs within a city representing some of its mobility features. In that sense, a novel multilevel grill partitioning of the space to represent these DTAs has been put forward.

Besides, such a representation avoids storing any type of personal information of the contributors so that the generated map of DTAs can be disclosed without potential privacy leaks. Last but not least, a novel mechanism based on MCS to detect such DTAs based on two different operation modes has been stated. These two operation modes allow the system to select the best set of contributing users depending on whether the goal is to discover new DTAs or just control the existing ones.

Finally, future work will focus on enriching the sensing mechanism by incorporating context-aware features in the mobile clients so that contributors can proactively decide when and where to activate their sensing capabilities. This would reduce the dependency with the central server.

Abbreviations

DTA: Dense transit area
$\mathscr{A}$: Set of discovered DTAs
$r(p)$: Ongoing route of the user p
$r(p)_l$: Sequence of timestamped locations of the route $r(p)$
$r(p)^{\text{ended}}$: Finished route of a user p
$r(p)_l^{\text{ended}}$: Sequence of timestamped locations of the route $r(p)^{\text{ended}}$
$\mathscr{S}_{\text{gap}}$, $\mathscr{T}_{\text{gap}}$: Temporal and spatial gaps between consecutive routes
$\mathscr{C}$: Grilled partition of the city under study
c: Squared cell in $\mathscr{C}$
$\mathscr{C}(c)^{\text{sub}}$: Set of subcells of a cell c
$r(p)_c^{\text{ended}}$: Sequence of cells c of the route $r(p)^{\text{ended}}$
$r(p)_c$: Sequence of cells c of the route $r(p)$
$\mathscr{C}_{\text{tr}}^{w}$: Historical data of walking routes in $\mathscr{C}$
$\mathscr{C}_{\text{tr}}^{\text{nw}}$: Historical data of nonwalking routes in $\mathscr{C}$
c_{tr}: Historical data of a cell in $\mathscr{C}_{\text{tr}}^{w}$ or $\mathscr{C}_{\text{tr}}^{\text{nw}}$.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This paper has been possible partially by the European Commission through the H2020-ENTROPY-649849 and the Spanish National Project CICYT EDISON TIN2014-52099-R (Grant BES-2015-071956) granted by the Ministry of Economy and Competitiveness of Spain (including ERDF support).

References

[1] Z. Yan and D. Chakraborty, "Semantics in mobile sensing," *Synthesis Lectures on the Semantic Web: Theory and Technology*, vol. 4, no. 1, pp. 1–143, 2014.

[2] H. Ma, D. Zhao, and P. Yuan, "Opportunities in mobile crowd sensing," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 29–35, 2014.

[3] C. Krner, M. May, and S. Wrobel, "Spatiotemporal modeling and analysis-introduction and overview," *KI-Künstliche Intelligenz*, vol. 26, no. 3, pp. 215–221, 2012.

[4] A. M. Hendawi and M. F. Mokbel, "Predictive spatio-temporal queries: a comprehensive survey and future directions," in *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems (MobiGIS '12)*, pp. 97–104, ACM, 2012.

[5] A. Steinfeld, D. Manes, P. Green, and D. Hunter, "Destination entry and retrieval with the ali-scout navigation system," Tech. Rep. UMTRI-96-30, University of Michigan,Transportation Research Institute (UMTRI), 1996.

[6] F. Terroso-Saenz, M. Valdes-Vela, and A. F. Skarmeta-Gomez, "A complex event processing approach to detect abnormal behaviours in the marine environment," *Information Systems Frontiers*, vol. 18, no. 4, pp. 765–780, 2016.

[7] J. Zhang and M. F. Goodchild, *Uncertainty in Geographical Information*, Taylor & Francis, London, UK, 2002.

[8] F. Terroso-Saenz, M. Valdes-Vela, E. den Breejen, P. Hanckmann, R. Dekker, and A. F. Skarmeta-Gomez, "CEP-traj: an event-based solution to process trajectory data," *Information Systems*, vol. 52, pp. 34–54, 2015.

[9] M. De Berg, M. Van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry*, Springer, Berlin, Germany, 2000.

[10] Y. Zheng, X. Xie, and W.-Y. Ma, "Geolife: a collaborative social networking service among user, location and trajectory," *IEEE Data Engineering Bulletin*, vol. 33, no. 2, pp. 32–39, 2010.

[11] E. Frentzos, K. Gratsias, and Y. Theodoridis, "Index-based most similar trajectory search," in *Proceedings of the 23rd International Conference on Data Engineering (ICDE '07)*, pp. 816–825, Istanbul, Turkey, April 2007.

[12] M. Haklay and P. Weber, "OpenStreetMap: user-generated street maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.

[13] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell, "Bikenet: a mobile sensing system for cyclist experience mapping," *ACM Transactions on Sensor Networks*, vol. 6, no. 1, article 6, 2009.

[14] L. Zhang, D. Gao, W. Zhao, and H.-C. Chao, "A multilevel information fusion approach for road congestion detection in VANETs," *Mathematical and Computer Modelling*, vol. 58, no. 5-6, pp. 1206–1221, 2013.

[15] J. Wan, J. Liu, Z. Shao, A. V. Vasilakos, M. Imran, and K. Zhou, "Mobile crowd sensing for traffic prediction in internet of vehicles," *Sensors*, vol. 16, no. 1, article 88, 2016.

[16] Á. Petkovics and K. Farkas, "Efficient event detection in public transport tracking," in *Proceedings of the International Conference on Telecommunications and Multimedia (TEMU '14)*, pp. 74–79, Heraklion, Greece, July 2014.

[17] S. Hu, L. Su, H. Liu, H. Wang, and T. F. Abdelzaher, "SmartRoad: a crowd-sourced traffic regulator detection and identification system," in *Proceedings of the 12th International Conference on Information Processing in Sensor Networks (IPSN '13)*, pp. 331–332, ACM, Philadelphia, Pa, USA, April 2013.

[18] Y. Zheng and X. Zhou, *Computing with Spatial Trajectories*, Springer Science & Business Media, New York, NY, USA, 2011.

[19] H. Gao and H. Liu, "Mining human mobility in location-based social networks," *Synthesis Lectures on Data Mining and Knowledge Discovery*, vol. 7, no. 2, pp. 1–115, 2015.

[20] T. Guo, Z. Yan, and K. Aberer, "An adaptive approach for online segmentation of multi-dimensional mobile data," in *Proceedings of the 11th ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE '12)*, pp. 7–14, Scottsdale, Ariz, USA, May 2012.

[21] N. J. Yuan, Y. Wang, F. Zhang, X. Xie, and G. Sun, "Reconstructing individual mobility from smart card transactions: a space alignment approach," in *Proceedings of the 13th IEEE International Conference on Data Mining (ICDM '13)*, pp. 877–886, IEEE, Dallas, Tex, USA, December 2013.

[22] M. Lin and W.-J. Hsu, "Mining GPS data for mobility patterns: a survey," *Pervasive and Mobile Computing*, vol. 12, pp. 1–16, 2014.

[23] J. Krumm, R. Gruen, and D. Delling, "From destination prediction to route prediction," *Journal of Location Based Services*, vol. 7, no. 2, pp. 98–120, 2013.

[24] J. Zhou, A. K. H. Tung, W. Wu, and W. S. Ng, "A 'semi-lazy' approach to probabilistic path prediction in dynamic environments," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*, pp. 748–756, ACM, 2013.

[25] B. D. Ziebart, A. L. Maas, A. K. Dey, and J. A. Bagnell, "Navigate like a cabbie: probabilistic reasoning from observed context-aware behavior," in *Proceedings of the 10th International Conference on Ubiquitous Computing (UbiComp '08)*, pp. 322–331, ACM, Seoul, South Korea, September 2008.

[26] C. Zhou, D. Frankowski, P. Ludford, S. Shekhar, and L. Terveen, "Discovering personal gazetteers: an interactive clustering approach," in *Proceedings of the 12th Annual ACM International Workshop on Geographic Information Systems (GIS '04)*, pp. 266–273, ACM, 2004.

[27] B. W. Sharman and M. J. Roorda, "Analysis of freight global positioning system data: clustering approach for identifying trip destinations," *Transportation Research Record*, no. 2246, pp. 83–91, 2011.

[28] S. Scellato, M. Musolesi, C. Mascolo, V. Latora, and A. T. Campbell, "NextPlace: a spatio-temporal prediction framework for pervasive systems," in *Pervasive Computing: 9th International Conference, Pervasive 2011, San Francisco, USA, June 12–15, 2011. Proceedings*, K. Lyons, J. Hightower, and E. M. Huang, Eds., pp. 152–169, Springer, Berlin, Germany, 2011.

[29] Z. Li, J. Han, B. Ding, and R. Kays, "Mining periodic behaviors of object movements for animal and biological sustainability studies," *Data Mining and Knowledge Discovery*, vol. 24, no. 2, pp. 355–386, 2012.

[30] T. Bhattacharya, L. Kulik, and J. Bailey, "Extracting significant places from mobile user GPS trajectories: a bearing change based approach," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '12)*, pp. 398–401, ACM, Redondo Beach, Calif, USA, November 2012.

[31] X. Xiao, Y. Zheng, Q. Luo, and X. Xie, "Inferring social ties between users with human location history," *Journal of Ambient Intelligence and Humanized Computing*, vol. 5, no. 1, pp. 3–19, 2014.