Chapter V

LIDU - Location-based approach to IDentify similar interests between Users in social networks

Contents

We have observed that people work or live in different places but have trajectory correlations in their daily routines. The users' daily routines, therefore, can be captured by mobile social applications and shared in virtual communities in order to increase social interactions in real communities.

Since we have noted this viability to increase social interactions in real communities and the large widespread of smartphones and social networks, we propose a middleware of services based on user's daily routines, called LIDU: Location-based approach to IDentify similar interests between Users in social networks. The key idea is to increase social interactions by relating daily routines and points of interest based on trajectories of mobile users. For instance, a mobile social application jointly with a social network has to be able to answer the following questions:

1. Which of my friends stop in my preferred bakery in Grenoble at the same period of the day?
2. Do any of my friends pass near my apartment to get from their home to their work?
3. Which of my contacts will be passing into the campus of the University of Grenoble during the week? [1]

While these questions are interesting to obtain information of similarities between users' daily routines, some scientific challenges were considered in the designing of our approach. The challenges are mainly related to traditional and new problems involving social networks, mobile computing technologies and spatial data representation. We point out these challenges as follow:

- Determine the relations between users of social networks.
- Propose integrated software architecture according to the characteristics of mobility scenario, such as limited resources, network and sensors.
- Define the structure of user profiles in order to facilitate the association between trajectory and context data of users.
- Design a robust data model to describe the spatial environment, taking into account different levels of the spatial information. The data model helps the classification of spatial knowledge based on points of interest (e.g., bakery, apartment, campus), spatial relations (e.g., near my apartment) and geographic entities (e.g., Grenoble).
- Extend the data model to represent the relations between spatial and temporal data, which allows the characterization of user's trajectories in multiple context information.
- Consider the aspects of the quality of data (e.g., sensored data) and the sharing of personal users' information (respecting the privacy features).

[1]The user defines the contacts to share his/her daily routine.

- Explore the available knowledge in order to identify spatial and contextual similarities between users, taking into account the performance and robustness of the approach.

- Propose a generic system to provide adaptable services for different types of applications, such as a recommendation system.

With these challenges in mind we decided to propose a middleware due to its inherent characteristics, which focus on bridging the gap between applications and low-level constructs [169]. A middleware is able to achieve the requirements of these presented challenges and provide more features that facilitates the extension of our approach, such as scalability, heterogeneity, dynamicity, adaptability, knowledge managing, data association, quality of service and security. In summary, a middleware helps developers to create applications that make queries to the middleware services and get results back in an efficient way [170].

Figure 5.1 illustrates the architecture overview of our middleware approach. As we note, two main input data are acquired by the middleware, which are trajectory data (jointly with context information) and social connections between users. Social connection data are directly processed by the data-modeling algorithm. Clustering algorithm receives the GPS trajectories to discover the best representative trajectory of each user. After that, the correlation algorithm identifies the similar points of interest between users. Finally, the data-sharing algorithm is able to adapt the information according to the requirements of each application.

Formally, our middleware allows the execution of algorithms to capture, store, process and share similarity information derived from users' daily routines. Firstly, we use smartphones and their sensors to capture users' daily routines and context information. Secondly, all information is transferred and stored in a relational database located on a server application, which is used as a plug-in on a social network platform. Besides that, we explore the capabilities provided by clustering algorithms to analyze user trajectories and extract relevant information from a large amount of data. Finally, we use an optimized trajectory correlation algorithm to identify similar routines between friends in social networks.

Although the core of our middleware is situated in the middle of the presented architecture, the data acquisition process has to be well defined in order

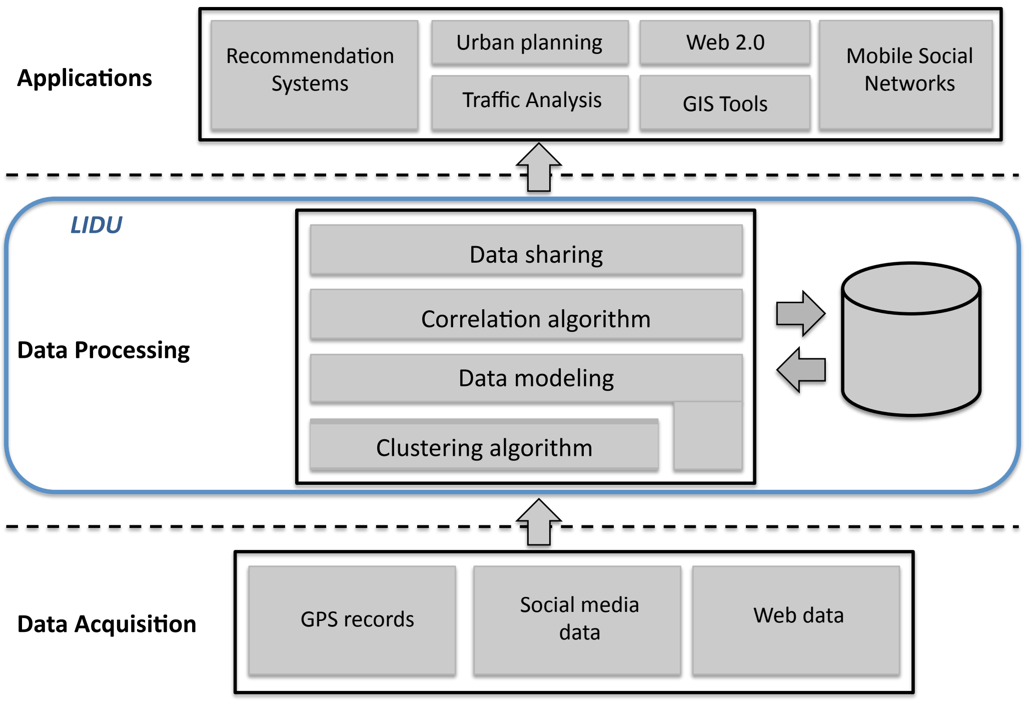


Figure 5.1: Architecture overview.

to provide the relevant data to our algorithms. Therefore, we present the data acquisition process, called profile building entity. The profile building can be denoted as an algorithm to acquire trajectory data and their context information through the use of smartphones. After capturing the profile building component sends the acquired data to the trajectory correlation component, which is the core of our middleware. At this moment, the algorithms into the middleware process the data. These two main entities are presented in Figure 5.2.

Therefore, we start showing the main parts that compose the data acquisition module, which was adapted and implemented to our approach.

1 Profile building

The user profile can be determined taking into account two basic types of data that are used for constructing and enriching the data model. These two basic types are defined as *personal* and *contextual* data. Personal data describes the main features of an entity and the contextual data characterizes the situation.

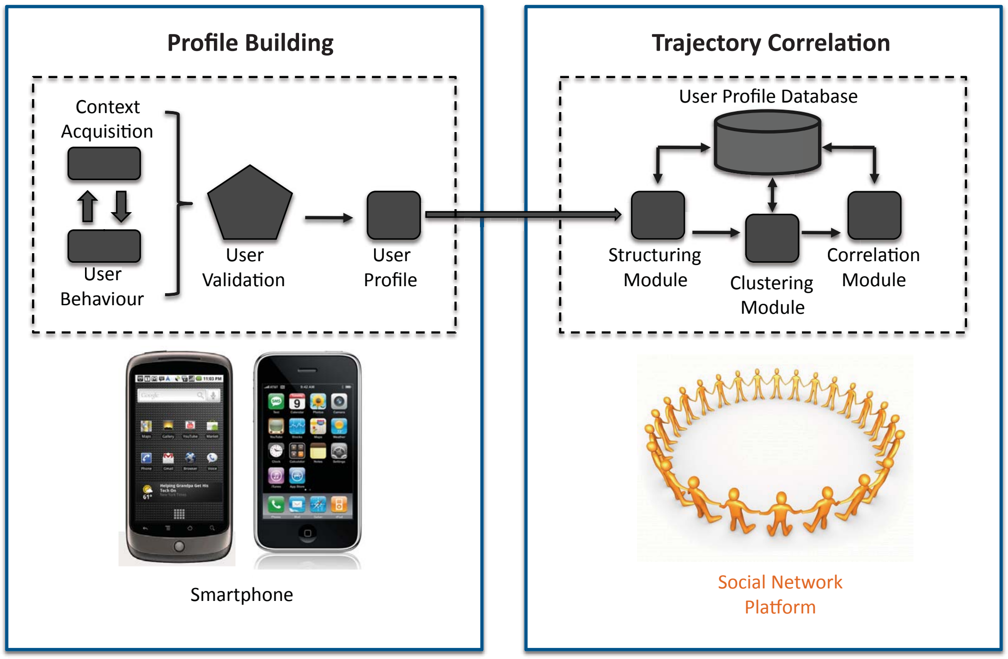


Figure 5.2: Main components of our middleware.

An entity can be a person, place, physical or computational object. For example, in a personal tracking application for mobile users, the personal data would be the information about the user, such as name, birthday, gender, etc. On the other hand, contextual data would be composed of movement records that the user performed over a period of time. A movement record can include such characteristics as the initial point, speed, direction, and time, as well as weather information. We define an entity as a mobile user using a smart phone equipped with GPS, digital camera and Internet connection (e.g. 3G or Edge).

For the contextual data organization, we have followed the concepts and relations of Context Top ontology, introduced by the authors of [7]. Figure 5.3 illustrates this ontology, where *Action* has a *Context* that is composed by some *Context Elements*. The context can also describe the situation of its elements through the property of *describeTheSituationOf*, which *hasContext* is its opposite property.

Based on the Context Top, we divide the context in five main dimensions: social, spatial, temporal, spatio-temporal and computational. The social dimension is related to the features associated with the user, such as user profile and social relations in a social network. The spatial dimension provides the

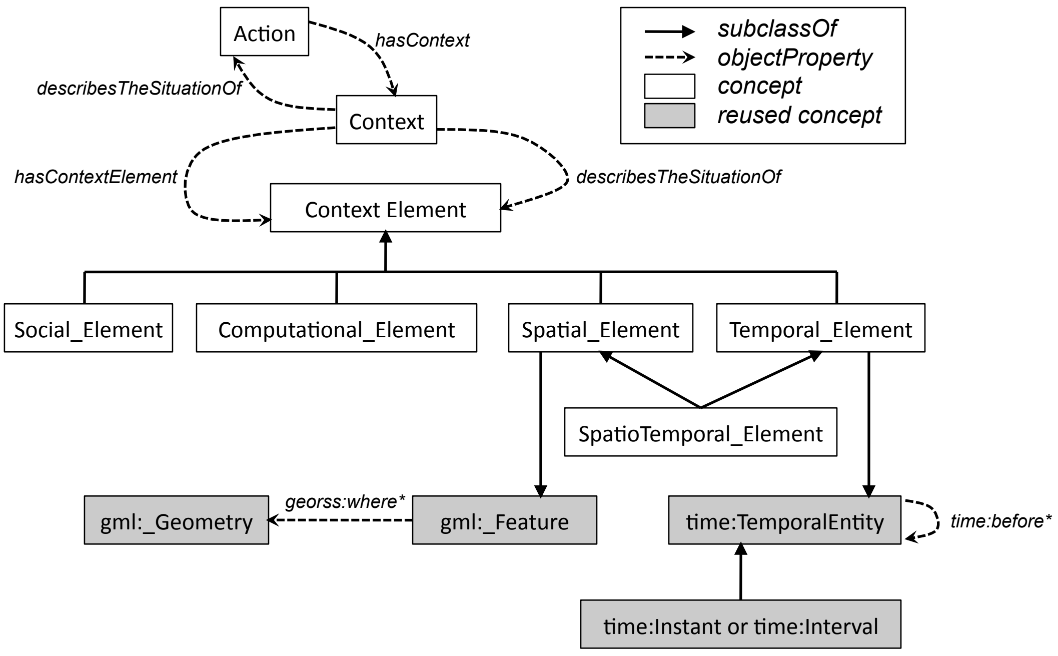


Figure 5.3: Context Top ontology concepts and relations [7].

spatial information about the environment where the action is done, for example: geographic coordinates, postal address, etc. The temporal dimension is composed by the information about time, such as the date, the days in a week, etc. The spatio-temporal dimension has the information derived from the spatial and temporal dimensions (e.g., weather). Finally, the computational dimension offers the facilities provided by the embedded software in the system (e.g., sensors, mobile applications, etc.). Therefore, the features that are used in each dimension are defined by the developer of the context-aware system. We have adopted the same data organization presented in Figure 5.3 for defining the context data generated by our profile building process.

We have also defined a third type of data, named *behavioral* data, which is derived from the association among personal and contextual data. We assume that behavioral data is defined as a user's daily routine that is generated based on the elements that compose a user trajectory and its associated contextual data. In other words, since the personal and contextual data are well acquired and associated, our approach allows the identification of a user's daily routine. We have two ways to identify a user's daily routine, based on a single trajectory or derived from a set of trajectories (e.g., a user that goes from home to work every day). Both ways can be executed by following our profile building

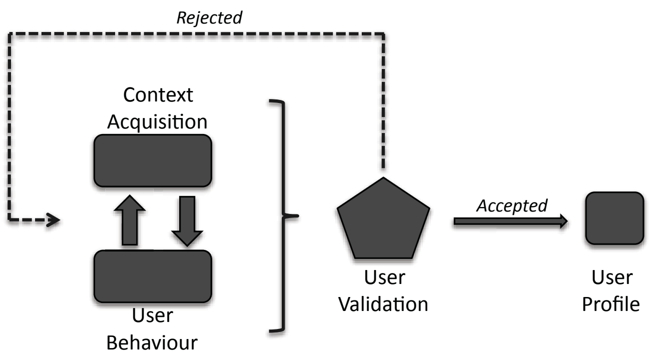


Figure 5.4: The profile building process.

process, illustrated on Figure 5.4.

The user can use a mobile application to register a single trajectory that describes his/her trajectory to go from home to work, for example. After visualizing and validating the trajectory that represents his/her daily routine, the user profile is created and the data is sent to the core of our middleware. Social connections are already available by some social network platform (e.g., Facebook, LinkedIn, Twitter) on the Internet. Therefore, this single trajectory and its contextual data are used to represent the user's daily routine.

On the other hand, the second way to define a user's daily routine is discovering his/her best representative trajectory from a set of trajectories. Since the user registers more than one trajectory to represent the same daily movement, a clustering algorithm technique can be applied to recognize the best representative trajectory. For example, a user took the similar path to go from home to work for 3 times in a period of 5 days. For the other 2 days, this user decided to change the path due to some incident or problem. For this reason, the 3 similar trajectories could be used to represent the best representative trajectory. Consequently, this best representative trajectory represents the user's daily routine.

In our approach, we provide a method to recognize a user's daily routine from one or multiple trajectories. Following the steps presented in Figure 5.2, the structuring module verifies if there is a previous trajectory for the user. If there is no trajectory, it creates a new user's daily routine. On the other hand, if multiple trajectories are found, clustering and aggregation techniques are used to identify the aggregated trajectory (a best representative of user's daily routine) [171]. As previously mentioned, we apply the OPTICS algorithm to classify user trajectories based on their daily routes.

The clustering and aggregation module provides the best representative trajectory for each user. This aggregated trajectory from one user is compared to other users by applying our trajectory correlation algorithm (Section 3). This approach enables groups of users to share similar routes to increase geospatial social interaction. The user daily routine then is enriched with additional information about each location in the database. The structuring module then exports the enriched information to update the user profile database.

Assuming that these data are available on the Internet and, consequently, are connected to some social network platform, the social relations can be used to enrich the database. The structuring module requests the social relations for each user who has registered his/her trajectory on the database. Hence, the comparison is performed based on the type of relation between users, for example: best friends, family, colleagues, etc. In our study, we assumed that the comparison of trajectories could use this feature as a filter to avoid security problems, mainly involving privacy.

While the capabilities to capture a sequence of positions, to enrich the database and to discover the best representative trajectory are the starting point of managing movement, designing a middleware based on trajectory data requires a structural approach. After obtaining these trajectories, modeling them becomes necessary for important operations, such as: i) to indentify patterns, which will be used for decision making (e.g. registering users trajectories within a city for optimizing traffic of vehicles); ii) to query information about the moving objects (e.g. enriching trajectory data with context information); iii) to optimize intelligent transport systems (e.g. motivating users to use car pooling alternatives in order to reduce the number of vehicles in urban regions).

2 Multi-layer data representation based on user routines

The main motivations to design a suitable data model are related to providing an easy way to manipulate trajectory data, to use structured query languages, to specify profiles through movements, to create and compare profile groups.

In parallel, the identification of the scenario is a significant requirement to design a conceptual data model. In this thesis, we take into account the scenario of an employee that goes from home to work and back everyday within a city, whose the user's daily routine can be represented at different abstraction levels. In addition, we consider a diversity of semantic data that enriches the knowledge on these trajectories. For a user daily trip, we can obtain information about possible user interests based on his/her movements. For example, whenever the user goes from work to home, he usually stops at a specific coffee shop.

Therefore, the conceptual model for trajectories must be able to analyze and manage simple trajectories (direct travels from origin to destination) as well as complex trajectories (where the trajectory is semantically composed of separate segments and/or different abstraction levels). Furthermore, the data model must relate any type of semantic annotation to trajectories, such as attributes of each trajectory and connections between the trajectory and an object stored in the database.

Often, it is important to understand the movement data at multiple abstraction levels for pattern recognition and analyzing movement behaviors as well as to deduce the relationships between users in location-based social networks. In order to create a flexible data model for mobile social application context, we propose a multi-layer data representation of moving objects based on user routines. A specific place as well as a segment or a whole trajectory can denote these user routines.

Several researchers have shown an interest in analyzing and representing spatio-temporal data [15][6][14]. This data is relevant in a number of areas such as social interaction, data mining, medicine and geographical information system. For instance, in the context of social interaction, we pointed out some approaches related to collecting and analyzing daily trajectories of humans, addressing issues such as daily routine, mobility, sport, trips, and social networks. In all these approaches, the amount of data produced is very large and is therefore challenging to interpret.

In parallel, the need for representing information about PoI on the Web has emerged [3] in order to manage and organize context-aware information. Interesting issues include how points or regions can be correlated through multi-layer representation [172] and how user trajectories could be analyzed in terms of their distance to another one [173].

Multi-layer data representation has been of interest for a long time due to its importance for spatial data representation [174][175][176]. In spite of the large number of issues about multi-layer data representation, there is a lack of multi-layer representation techniques for moving object trajectories. In [177], the authors present a design for multi-layer spatial objects in which both spatial objects and the vertices of their component geometry are labeled with level priority values. Although the data model supporting queries at different abstraction levels is very interesting, it is not intended for representing trajectories and not easily extendable for this context.

In [178], the authors present an interesting Rule-based Location Prediction method (RLP), to guess the user's future location for location-based services. However, they do not consider the partial containment relationship between spatial regions at different spatial levels. In [179] and [180], the authors introduce approaches to consider trajectory patterns between different spatial levels as well as the relation among user, location and trajectory. In particular, GeoLife [179] is a social networking service which increases social connectivity among users taking multiple geospatial scales into account while the work described by [180] focuses on Regions of Interest (ROI) as opposed to multiple abstraction levels. In this thesis we extend the PoI data model proposed by W3C working group and present a multi-layer data representation of correlated trajectories, taking into account the PoI at multiple abstraction levels.

As introduced in Chapter IV, PoI is composed of any number of the following entities:

- **label:** is a human explicit label to name PoI. This entity is important to identify a specific place, which can be used to support the definitions of labels in the different levels of our data model.
- **description:** a human explicit description about the PoI.
- **category:** this entity classifies PoI into a category. For example, it can be a primary attribute (e.g., museum, bar, restaurant), a popularity ranking, or a security rating.
- **time:** time is considered the most common context information, which is generally represented by the time instant that the location was acquired. Time is also used to estimate the duration of an object at a place [24].

- **link:** this entity is a generic manner to represent a relationship from a PoI to another PoI, or from a web resource to a PoI, both based on the RFC 4087 technique (point-to-point link).
- **metadata:** in this entity, we can insert formal metadata to the PoI (by reference, for example).

Therefore, we have used this definition to construct our data representation, which is presented in the next section.

2.1 Data representation

In our work, we assume that the interests of a user for a specific place, segment or trajectory can represent a user routine. For instance, a user likes to eat at the restaurant X everyday, where this restaurant is a point of interest. In the same way, a user prefers to take a specific street (segment) or a set of different segments (trajectory) to go from home to work. Along this line, a user routine can be defined following a multi-layer representation (see Figure 5.5), where n represent the identifier of each element of the routine, and the links are the relations between these elements at different abstraction levels.

Taking into account the representation presented in Figure 5.5, we classify user routines as Trajectory of Interest (ToI), Segment of Interest (SoI) and PoI at different abstraction levels. Therefore, we define this spatial information to be a multi-layer data representation in order to support the description of the user's daily routine.

According to Figure 5.5, a user routine is presented based on its layer. For instance, the last layer (*Layer 3*) can be represented by the name of the location according to the GPS coordinate (e.g. bakery's name, house number, etc.), based on the PoI data model proposed by W3C working group (with the same entities). Nevertheless, we inserted the entity called *user_id* to identify the owner of the PoI. In parallel, we reused and adapted the PoI data model to define the entities and values of SoI and ToI.

Following our multi-layer data representation and the reference model of W3C, the *Layer 2* is defined as the Segments of interest (SoI) that compose the user trajectory, where each SoI is composed of any number of the following entities:

- **user_ID:** is used to identify the owner of SoI.

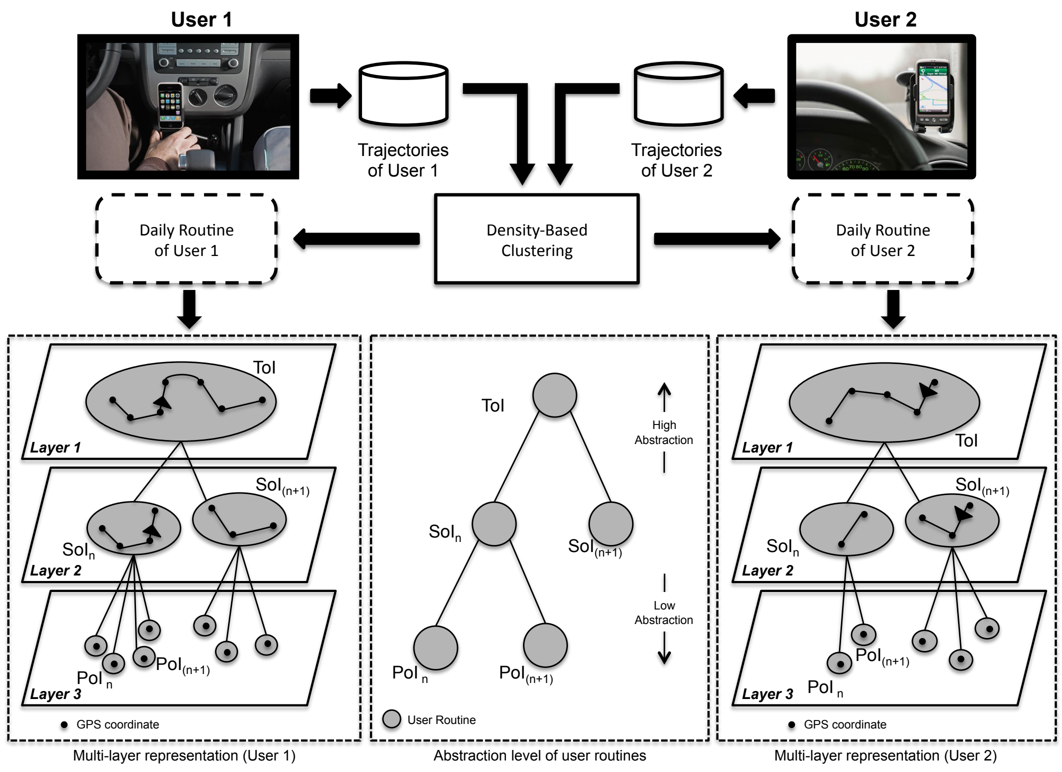


Figure 5.5: Our multi-layer data representation.

- **label:** is a human explicit label to name SoI, which can be generated by using the labels of PoI (e.g., from Work to Bakery).

- **description:** a human explicit description about the SoI.

- **category:** the classification of SoI into a category. Similar to the category of PoI, it can be a primary attribute (e.g., street, avenue, highway), a popularity ranking, or a security property.

- **time:** for this entity, we can have the time interval that the moving object stayed into SoI, based on the initial and final time instants. These time instants are derived from the time instants of the corresponding initial and final PoI's of the segment.

- **link:** similar to the PoI, this entity is a generic manner to represent a relationship from a SoI to another SoI, where the last PoI of the segment has a link with the initial PoI of the next segment.

- **metadata:** it the use of a formal metadata to SoI (by reference, for example).

Finally, ToI in the *Layer 1* could be represented by a whole user trajectory (e.g. to go from home to work). Therefore, we identify the following entities that compose each ToI:

- **user_ID:** is used to identify the owner of ToI.
- **label:** is a human explicit label to name ToI, which can be also generated by using the labels of PoI (e.g., from Work to Home) or by the labels of SoI (e.g., from street X to avenue Y).
- **description:** a human explicit description about the ToI.
- **category:** the classification of ToI into a category. Similar to the category of SoI, it can be a primary attribute (e.g., name of the region that the whole trajectory was registered), a popularity ranking, or a security property. The main characteristic for the security property is related to the access control polices for a user trajectory. Based on the level of the relationship with another user, the user can control the sharing of the whole trajectory (e.g., Public, List_of_Group_Access (specific group of friends in my social network), Private or List of users). While this property can be defined by the user in ToI, it can be also defined in the security properties of SoI and PoI.
- **time:** the time interval that the moving object stayed into ToI, based on the initial and final time instants. Similar to the time entity of SoI, these time instants are derived from the time instants of the corresponding initial and final PoI's of the trajectory. In addition, with this information we can identify the period of the day and the days of the week, for example.
- **metadata:** it the use of a formal metadata to ToI (by reference, for example).

Based on this structure, a user routine is presented as a general interest according to the abstraction level of the user/system. Besides that, a ToI is directly related to a set of SoI's and/or PoI's at low levels. To better understand this relation, we use a tree structure to show the relation between each information according to the multi-layer data representation.

Based on the illustrated data representation, we design our multi-layer data model for trajectories, taking into account different abstraction levels of user routines. In the following we provide the basic definitions to support our discussion.

1. **Trajectory** (T) is defined as a set of consecutive points captured through a GPS of one trip performed by a user. Each location (L) is composed of a set of information (latitude, longitude, altitude, direction, time stamp for each registered point (t_L) and an approximate speed provided by the GPS). $T = \{L_1, L_2, L_3, ..., L_n\}$, the time interval between two points is computed by the subtraction of $t_{L(k+1)} - t_{L(k)}$, where $(1 \leq k < n)$. This temporal information also allows the recognition of pause instants, according to the proposal of [24]. Although the points are characterized by latitude, longitude and altitude, we focus on points in 2D space (latitude and longitude) to represent the position of each user.

2. **User Routine** (UR) is defined as a human construct to represent a routine of a user based on his/her interest. UR typically denotes a user interest, where a user can identify an entire trajectory, segment of route (e.g. street name) or place (e.g., bakery X), according to the layers presented in Figure 5.5, typically represented by name and characterized by type, which may be used as a reference point or a target in a location based service request (e.g., route destination).

3. **Set of UR** (SUR) is defined as the set of user routines based on the abstraction level of multi-layer representation. The user routine of each abstraction level is defined according to its identifier (ur), such that $traj$, seg and poi represents the ToI, SoI and PoI respectively. Therefore, SUR is formed by a finite set and subset of user routines in different abstraction levels, e.g. $SUR = \{ur_{traj}\{ur_{(seg,1)}, ur_{(seg,2)}, ur_{(seg,n)}\}$, $ur_{seg}\{ur_{(poi,1)}, ur_{(poi,2)}, ur_{(poi,n)}\}$, ... , $ur_{(s-1)}\{ur_{(s,1)}, ur_{(s,2)}, ur_{(s,n)}\}\}$, where s represents the abstraction level. For instance, the set to represent a user trajectory in the campus of Joseph Fourier University is

$$\begin{aligned} SUR_{traj} = \{ \; & Chemistry\ Street\{Grenoble\ Informatics \\ & Laboratory, CERMAV\ Laboratory\}, \\ & Piscine\ Street\{ENSIMAG\ Laboratory\}, \\ & Library\ Street\{Central\ Library, Mathe- \\ & matics\ Laboratory\}\} \end{aligned}$$

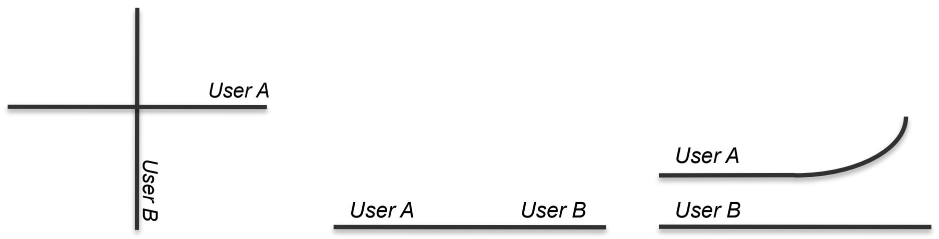
where *traj* can be represented by the user trajectory in the *Layer 1*, *Chemistry Street*, *Piscine Street* and *Piscine Street* are road segments in the *Layer 2*, and *Grenoble Informatics Laboratory*, *CERMAV Laboratory*, *ENSIMAG Laboratory*, *Central Library* and *Mathematics Laboratory* are local places in the *Layer 3*.

The intention to design a conceptual model is to offer basic procedures in order to support designers in the development of mobile social applications. A usual feature in the spatial multi-layer data model is the user routine corresponding to a given abstraction level (trajectory, road segment and local place).

When we consider that a graph of user interests is a tree, we can say that a user interest is associated with ur in different abstraction levels, which allows to indicate that a user interest belongs to the abstraction level s (*traj*, *seg* and *poi*) associated with ur. Since the multi-layer data representation is presented, we take into account the organization of objects for a defined abstraction level. Consequently, a low abstraction level offers the set of PoI's to describe a user trajectory at the highest abstraction level. We observe that for all user routine shown in the data representation, we may have a specific UR available at each abstraction level (s), such that $L \in ur_{poi}$. This representation offers a procedure to understand the set of abstraction levels.

Finally, since two users A and B have a relation in the social network, our data model allows the identification of similar user routines between them, taking into account the different situations, presented in Figure 5.6. We consider the representation of three main situations of social interaction between users.

For the first situation (Figure 5.6(a)), we observe that two users have a point of interaction at the crossing of two UR's (e.g. road segment). Assuming that a user A passes in a specific region (e.g., at campus of University of Grenoble) and the user B also passes at this campus, we cannot affirm that both users are sharing a location L in ur_{poi}. However, our data model provides



(a) Crossing. (b) Sharing a UI. (c) Nearing.

Figure 5.6: Three main representations of situations that we consider as similarities between two users.

a manner to identify this crossing in different abstraction levels. Since we identify common regions between both users, we can identify similar segments and points of interest, allowing the identification of similar routines in different abstraction levels.

In Figure 5.6(b) the point of interaction could be the complete set of PoI (e.g. all road segment or a part of it). For this example, when we identify that both users are sharing a street, it is not evident that they are sharing the same part of this segment. However, while the similar segment is identified, our algorithms verify if the locations represented in the PoI layer corresponds to the same part of the shared segment.

Finally, in Figure 5.6(c), the most important information is the proximity between users. Hence, this proximity can be determined according to each layer in our model. The user could define this proximity. Consequently, the users can consider a possible social interaction due to the proximity of their trajectories, segments or points of interest.

2.1.1 Multi-layer representation of correlated trajectories

As one or a set of user interests may describe a user routine, we need to consider every information of each abstraction level (*ToI*, *SoI* and *PoI*). We then define a user trajectory as a sequence of UR's, where the set of segments crosses between different abstraction levels in the required order. The following example presents a multi-layer representation in order to illustrate our approach.

- $set_{ToI} = \{ur_{traj}\}$

- $set_{SoI} = \{ur_{(seg,1)}, ur_{(seg,2)}, ur_{(seg,3)}\}$
- $set_{PoI} = \{ur_{(poi,1)}, ur_{(poi,2)}, ur_{(poi,3)}, ur_{(poi,4)}, ur_{(poi,5)}, ur_{(poi,6)}, ur_{(poi,7)}\}$

For instance, we can construct the following sets of UR (SUR):

- $SUR_1 = \{ur_{traj}\{ur_{(seg,1)}\{ur_{(poi,1)}, ur_{(poi,2)}\}\}\}$
- $SUR_2 = \{ur_{traj}\{ur_{(seg,2)}\{ur_{(poi,3)}, ur_{(poi,4)}, ur_{(poi,5)}\}\}\}$
- $SUR_3 = \{ur_{traj}\{ur_{(seg,3)}\{ur_{(poi,6)}, ur_{(poi,7)}\}\}\}$

The Figure 5.7 illustrates these sets of UR's related to each abstraction level. In the next definition, the user routine descriptor (D) contains the sequence of the determined user routines. For instance, we determine two different trajectory descriptors for user 1 (D_1) and user 2 (D_2):

- $D_1 = < ur_{(seg,1)}, ur_{(seg,2)}, ur_{(poi,7)} >$
- $D_2 = < ur_{(poi,1)}, ur_{(poi,3)}, ur_{(poi,5)} >$

We note that the descriptors can be composed by UR's at different abstraction levels due to multiple location names, which can be obtained from reverse geocoding services. Therefore, our data representation is also able to find a similarity although these UR's are at different abstraction levels. The concept of multi-layer representation is an important step to understand the relations and similarities between UR's, grouped in different user descriptors. For instance, if we consider D_1, the user describes a trajectory from a departure ur_{seg} (at the second abstraction level) to a destination in a ur_{poi} (at the third abstraction level). In case of D_2, the user describes his/her routine at the same level.

A multi-layer data representation should be able to identify the abstraction level of each UR. This data representation becomes an important element for providing the accurate information to identify the similarities between user routines. If we observe the previous trajectory descriptors and the three situations presented in Figure 5.6, we see some challenges to develop a data model at different abstraction levels. For instance, if we observe D_1 and D_2, we observe that the first user is passing in $ur_{(seg,2)}$ (at the second level) and the other user is passing in $ur_{(poi,3)}$ (at the third abstraction level). Therefore, our approach allows the identification of similar routines between users who are sharing UR's in different abstraction levels.

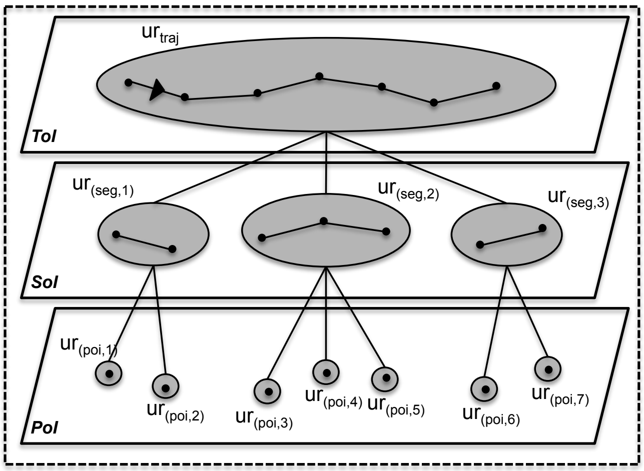


Figure 5.7: Example of multi-layer data representation.

2.1.2 Representation of temporal data

While the clustering algorithm processes the spatial information in order to identify the best representative trajectory for each user, the temporal information becomes relevant contextual information to enrich the services that are provided by our middleware. Hence, we designed a data representation of temporal information, which is detailed as follows.

Our approach follows the temporal representation presented in [148], where the time is processed after identifying the spatial similarities. Making use of the best representative trajectories, we obtain multiple information of time for each position in the user's trajectory. Figure 5.8 illustrates an example of a best representative trajectory with intermediary locations.

Assuming that this best representative trajectory was obtained by a dataset of *10* trajectories of a user to go everyday from home to work. Consequently, we have *10* working days for this example. The clustering algorithm then discovers that the user recorded *7* similar trajectories, by passing at the same streets and near to specific locations. Intuitively, we note that this user registered different time instants by location (illustrated by the points in the trajectory). We can see these different time instants in Table 5.1.

In Table 5.1, we may deduce that the user have traveled for three different trajectories in three working days to go from home to work. These days are *Day 2*, *Day 6* and *Day 10.* In contrast, we have seven trajectories that were recognized to construct the best representative trajectory. Given the

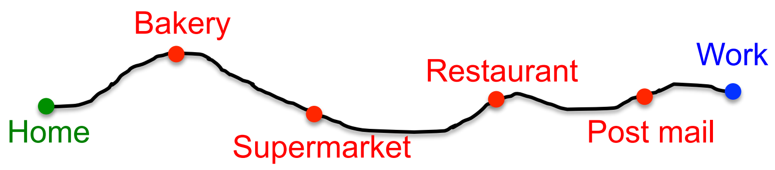


Figure 5.8: Example of a best representative trajectory with multiple locations between a departure (Home) and a destination (Work).

Locations	Day 1	Day 3	Day 4	Day 5	Day 7	Day 8	Day 9
Home	08:00	08:10	08:05	08:07	08:15	08:12	08:17
Bakery	08:07	08:18	08:12	08:15	08:23	08:20	08:25
Supermarket	08:15	08:26	08:20	08:23	08:30	08:28	08:32
Restaurant	08:25	08:36	08:29	08:31	08:37	08:35	08:39
Post mail	08:32	08:43	08:36	08:38	08:42	08:41	08:45
Work	08:40	08:50	08:43	08:45	08:50	08:47	08:52

Table 5.1: Time instants by location from a best representative trajectory of a user.

Supermarket as the location, we see that the user passed close to it at *08:15* in the first day, at *08:26* in the third day and at different time instants in the other 5 days.

Taking into account this example, we designed our representation of temporal data, where the key idea is to store all the time instants by location and represent them in a time interval. The time interval specifies all the time instants that the user passed close to each specific location. Finally, this data can be used to enrich the information that will be provided by our middleware.

3 The trajectory correlation algorithm to identify similar interests between users based on user's daily routines

Taking into account the idea to analyze user's daily routines in order to increase the number of social interactions between users, we propose an optimized algorithm based on Minimum Bounding Rectangles (MBR) [181] and the Hausdorff distance [182].

The Hausdorff distance is often used to determine the similarity of two shapes [183] and to measure errors for approximating a surface in generating a triangular mesh [184]. In our approach, we are interested to use Hausdorff distance computation in two different cases. Basically, the first case is applied when the algorithm finds a correlated area between two MBR's. It uses Hausdorff distance to compute the distance between the points that are in the correlated area. On the other hand, if there is no correlated area, the Hausdorff distance computation is used to compute the distance of near points between two MBR's. When the distance of two MBR's is found, the algorithm allows the expansion of both MBR's in order to find one or more points of social interactions, taking into account a threshold (D_{max}) for the expansion.

Firstly, we identify four extreme points of each trajectory (the northernmost, the southernmost, the westernmost and the easternmost). With these points, we create the MBR for the users' trajectories. Figures 5.9 illustrates the MBR for a specific trajectory.

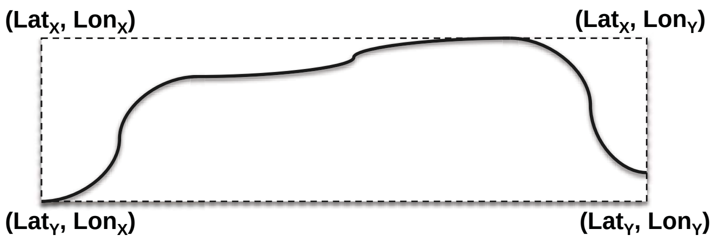


Figure 5.9: An example of MBR.

For instance, we consider two users A and B and the existence of MBR's for their respective trajectories. The four points to represent the rectangle of the user A are:

$$(Lat_{max(A)}, Lon_{min(A)}), (Lat_{max(A)}, Lon_{max(A)}),$$
$$(Lat_{min(A)}, Lon_{min(A)}), (Lat_{min(A)}, Lon_{max(A)}).$$

The points for the user B are:

$$(Lat_{max(B)}, Lon_{min(B)}), (Lat_{max(B)}, Lon_{max(B)}),$$
$$(Lat_{min(B)}, Lon_{min(B)}), (Lat_{min(B)}, Lon_{max(B)}).$$

Furthermore, we execute the trajectory correlation process according the algorithm as follows.

Algorithm 2 Main algorithm.

```
Input: two trajectories of users $A$ and $B$ with the points containing their respective coordinates.

Comment: It is verified if the two MBR's does not have a correlated area.

if $(Lat_{max(A)} < Lat_{min(B)})$ or $(Lat_{max(B)} < Lat_{min(A)})$ or $(Lon_{max(A)} < Lon_{min(B)})$ or $(Lon_{max(B)} < Lon_{min(A)})$ then
  Execute HausDist of MBR(A) and MBR(B);
  if HausDist $< D_{max}$ then
    Expand MBRs;
    Restart main algorithm;
  else
    There is no correlated area;
    Stop main algorithm;
  end if
end if

Comment: Otherwise, we select the correlated area and execute the HausDist algorithm.
Select correlated area (Alg. 3);
Execute HausDist (Alg. 4);

Output: The points in the correlated area and the distances between the points of $A$ in relation to the points of $B$.
```

As we can observe in the main algorithm, when there is no correlation between two MBR's, we execute an algorithm to compute the Hausdorff distance between two MBR's. The main reason to carry out this algorithm is related to the problem involving extreme points in the MBR faces. For example, we have a point in the right face of the MBR(A) and another point in the left face of the MBR(B). Although the MBR(A) is close to the left face of MBR(B), there might be no intersection, as presented in Figure 5.10. Then, we might have a problem, because two near points are not present in the correlated area.

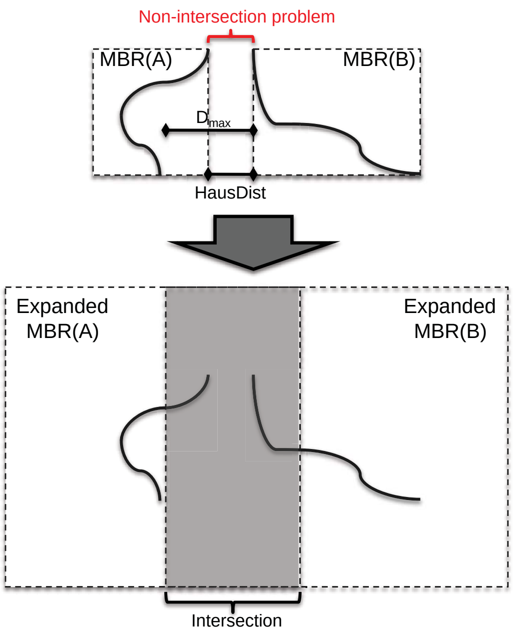


Figure 5.10: MBR Expansion for the non-intersection problem.

To solve this problem, we propose a MBR expansion algorithm, which computes the Hausdorff distance of two MBR's in order to verify if the expansion is possible or not according to the threshold D_{max}. The Hausdorff distance from the MBR(A) to the MBR(B) can be determined by exploiting the characteristic for each MBR area, there has to be at least one object that touches it. Therefore, we identify the area in MBR(A) closest to a face in MBR(B). After that, the algorithm computes the Hausdorff distance (HausDist) of these two faces and compare the result with D_{max}. If HausDist is less than D_{max}, then both MBR's expands their related areas from the current distance to the result of D_{max}. Figure 5.10 shows the MBR expansion process for the no intersection problem.

On the other hand, if there is an intersection of MBR's, the algorithm 3 is executed in order to determine the correlated area.

Since the correlated area of MBR's is found, the main algorithm executes the Hausdorff distance computation of the points. Assuming that a and b are points of sets A and B respectively and that they are in the correlated area, then the Algorithm 4 is executed.

Algorithm 3 Selection process

```
Input: $Lat_{min(A)}$, $Lat_{max(A)}$, $Lat_{min(B)}$, $Lat_{max(B)}$
if $Lat_{max(A)} > Lat_{max(B)}$ then
    Select $Lat_{max(B)}$
else
    Select $Lat_{max(A)}$
end if
if $Lat_{min(A)} > Lat_{min(B)}$ then
    Select $Lat_{min(A)}$
else
    Select $Lat_{min(B)}$
end if
if $Lon_{max(A)} > Lon_{max(B)}$ then
    Select $Lon_{max(B)}$
else
    Select $Lon_{max(A)}$
end if
if $Lon_{min(A)} > Lon_{min(B)}$ then
    Select $Lon_{min(A)}$
else
    Select $Lon_{max(B)}$
end if
Output: correlated area
```

Algorithm 4 Hausdorff distance algorithm

```
Input: points of trajectories $A$ ($a_i$ such as $i = 1$ to $n$) and $B$ ($b_j$ such as $j = 1$ to $m$), where $n$ and $m$ are the total of points in the trajectories $A$ and $B$ respectively.
HausDist = 0
for all point $a_i$ of A do
    shortest = Inf ;
    for all point $b_j$ of B do
        $distance_{ij}$ = distance ($a_i$ , $b_j$)
        if $distance_{ij}$ < shortest then
            shortest = $distance_{ij}$
        end if
    end for
    if shortest > HausDist then
        HausDist = shortest
    end if
end for
Output: the shortest distance of a point in the trajectory $A$ and another point in the trajectory $B$.
```

While the similar user routines are identified between two best representative trajectories, the algorithm starts the comparison between temporal information. To compare the temporal similarities between users, we consider all the similar locations identified. We have adopted the Parzen-window method [185] to identify temporal similarities by location. Parzen-window has been used in a large number of research areas, such as pattern recognition, data classification, image processing and tracking. We decided to use the Parzen window due to the well representation of each time instant at the time interval, where the density of the points can be easily recognized and visualized in the graph.

By definition, the Parzen-window is a density-based estimation that considers the data-interpolation technique [186]. Assuming that we have a random variable (x), then this technique computes the probability density function (PDF) in which the random variable was derived. In summary, it superposes kernel functions at each observation (x_i). Hence, the PDF ($f(x)$) of the Parzen-window is computed by

$$f(x) = \frac{1}{n} \sum_{n=1}^{n} \frac{1}{h_n^{dim}} K\left(\frac{x - x_i}{h_n}\right), \tag{5.1}$$

where $K()$ is the kernel function, dim is the dimensional space and h_n is the window width. Based on this equation, we are able to compute the value of $f(x)$ at a certain location (point). Along this line, we can determine a window function at x and define the total of observations xi that are close to the window.

For our approach, we determined the Gaussian PDF as the kernel function for Parzen-window density computation. Thus, the PDF $f(x)$ with the Gaussian function becomes

$$f(x) = \begin{cases} \frac{1}{n} \sum\limits_{k=1}^{n} \frac{1}{(h\sqrt{2\pi})^{dim}} e^{\left(-\frac{1}{2}\left(\frac{x-x_k}{h}\right)^2\right)} & \text{if } t_b < x < t_e \\ 0 & \text{otherwise,} \end{cases} \tag{5.2}$$

such as t_b is the initial time instant and t_e is the final time instant within the time interval for each location. As we are analyzing a point in comparison to another points in the time interval, the value of $dim = 1$. An important element related to the use of Parzen-window is the value of the window size (h). According to [187] and [188], when the Gaussian kernel is being used, the optimal value of h is defined by

$$h = \left(\frac{4\sigma^5}{3n}\right)^{\frac{1}{5}}, \tag{5.3}$$

where n is the number of time instants in the time interval and σ is the standard deviation of the samples.

Therefore, we can obtain the frequency that the user is near to a certain location. Since we have identified a similar routines between two users, we can compare the temporal graphs to know the probability of rendezvous between them at a certain period of time. Taking into account the example of the supermarket (Table 5.1) for a user A, we construct a time interval between 08 : 15 and 08 : 32, with the time instants [*08:15, 08:20, 08:23, 08:26, 08:28, 08:30, 08:32*]. Then, the PDF of the Parzen function then generates the graph presented in Figure 5.11 in order to represent all the time instants that user A passed near to supermarket in these seven days.

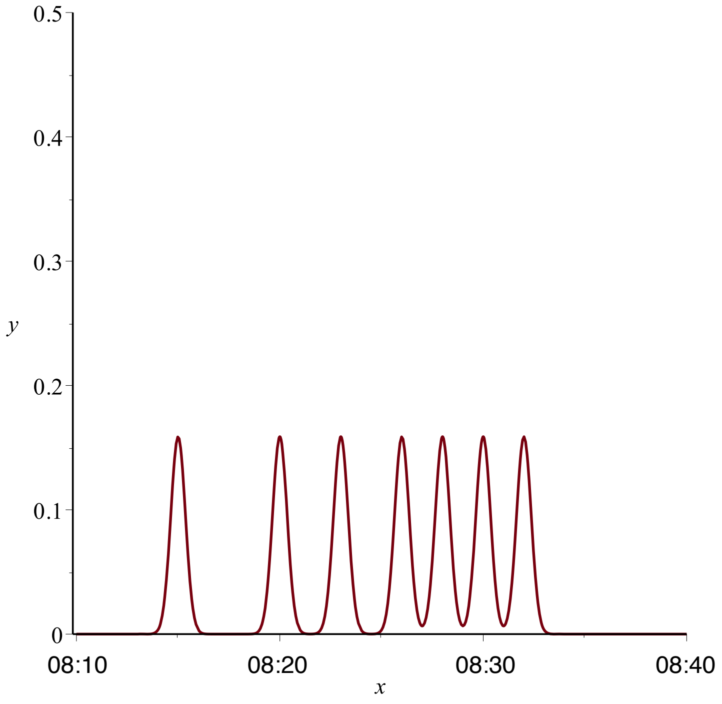


Figure 5.11: Time instants that user A have passed near to supermarket in the 7 days.

As we observe, the graph shows the probability of each time instant that user A was near to supermarket in the interval. Next, we assume that another user B (who is friend of A) have also passed near to the same supermarket in other ten days. Given a time interval of user B between 08 : 10 and 08 : 50, with the time instants [*08:10, 08:15, 08:16, 08:16, 08:20, 08:21, 08:30, 08:40, 08:42, 08:50*], the PDF of the Parzen function generates the graph presented in Figure 5.12.

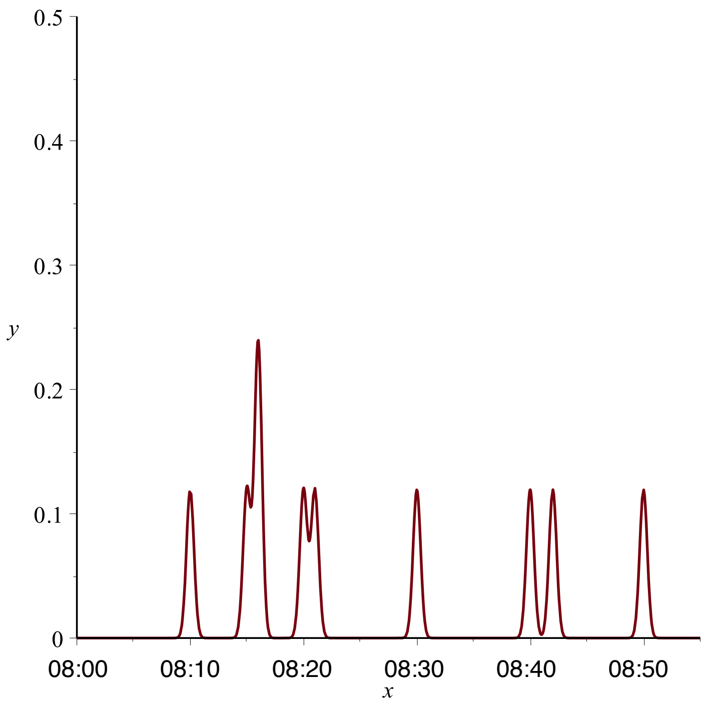


Figure 5.12: Time instants that user B have passed near to supermarket in the 10 days.

Intuitively, we observe that these graphs can represent all time instants in which both users have passed near to each location. Since we discover similar routines between users A and B, we can use these graphs to estimate the rendezvous between them, considering the temporal similarity. One way to compare these graphs is through the superposition, by observing the common areas. Another manner is to compute the probability from the highest value of time instant to the lowest value.

4 Sharing routines between users

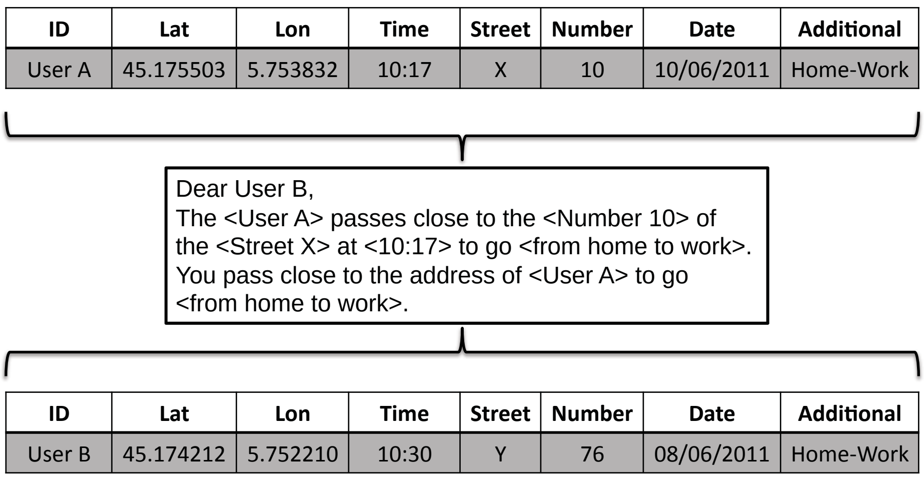
A well-known solution of Web applications that involves sharing and estimation of user interests is called recommendation system. In general, recommendation systems are classified in two groups, which are content-based and collaborative filtering systems [165]. In terms of content-based systems, a recommendation is performed based on the user preferences in relation to a specific content. For example, if a user prefers to listen country music than the other genres, the system recommends new songs having the "country" genre as the preference for that user. On the other hand, collaborative filtering systems recommend some information based on similar features of users and/or data. This kind of system is commonly used to recommend information that is preferred by a group of similar users.

Taking into account the characteristics of our approach, we have considered the collaborative filtering system as the best method to share the routines between users. These routines are represented by the similar user interests, which are identified by the trajectory correlation algorithm. For example, the recommending system is able to answer the question about a friend who is passing into the campus of the University of Grenoble during the week. Therefore, the collaborative filtering system verifies the user routines (in terms of spatio-temporal information) of a group of users to identify similar interests between them

Following the steps of our approach the data sharing algorithm can send a message to the user alerting that a friend passes in front of a specific number of the street X all the weekdays between 10:00 AM and 10:30 AM. This message can also contain accurate information about distance, which is acquired by the Hausdorff distance algorithm.

The final part of our middleware is the data-sharing algorithm, which enables the generation of an enriched information based on the processed data. It reads all the fields related to a correlated point in order to automatically create the message that will be sent to one or both users. Figure 5.13 shows the creation of a message by using context information, which will be sent to the user B about a possible point of social interaction with the user A.

The data-sharing algorithm can be applied to several types of applications, for example: mobile social applications, social networks, SMS, and others. Besides that, our proposal allows the inclusion of a color-based scheme for



ID	Lat	Lon	Time	Street	Number	Date	Additional
User A	45.175503	5.753832	10:17	X	10	10/06/2011	Home-Work

Dear User B,
The <User A> passes close to the <Number 10> of the <Street X> at <10:17> to go <from home to work>. You pass close to the address of <User A> to go <from home to work>.

ID	Lat	Lon	Time	Street	Number	Date	Additional
User B	45.174212	5.752210	10:30	Y	76	08/06/2011	Home-Work

Figure 5.13: The context information of a correlated point in the database of the user B about the user A.

the visualization of potential points of interaction, taking into account the probability of interaction among users. Finally, in the next chapter, we present the evaluation of our approach, taking into account different scenarios.

5 Conclusion

Virtual community platforms provide solutions to social connectivity, giving people the capability to share interests, opinions, and personal information with other users. Nevertheless, we argue that the absence of context-aware mechanisms in virtual communities could be one of the main reasons that social interactions are frequently missed. The users' daily routines, therefore, can be captured by mobile social applications and shared in virtual communities in order to improve the social connections in real communities.

In this chapter, we introduced our location-based approach to identify similar interests between users in social networks (LIDU). The key idea is to provide a middleware of services to acquire daily routines in order to find near points and, consequently, increase social interactions in real communities.

We presented a flexible multi-layer data model for mobile social application context based on user routines. We designed a conceptual view to be adaptable and acceptable to a set of generic features as well as to assist developers in designing solutions with the inherent complexity of trajectory semantics (spatio-temporal data). Besides that, we discussed how our data model could offer mobile social applications with direct support for trajectories. Next, we presented an algorithm to execute the trajectory correlation process based on Minimum Bounding Rectangles (MBRs) and the Hausdorff distance (Haus-Dist) for finding spatial similarities. Furthermore, we used Parzen-window technique to identify similarities of temporal data.

To validate our Approach, we implemented and tested a mobile social application for tracking daily routines. Additionally, we developed a plug-in on a virtual community platform to receive the user profiles and to execute the trajectory correlation algorithm. Our results are presented in the next chapter.

Chapter VI

Evaluation of our approach

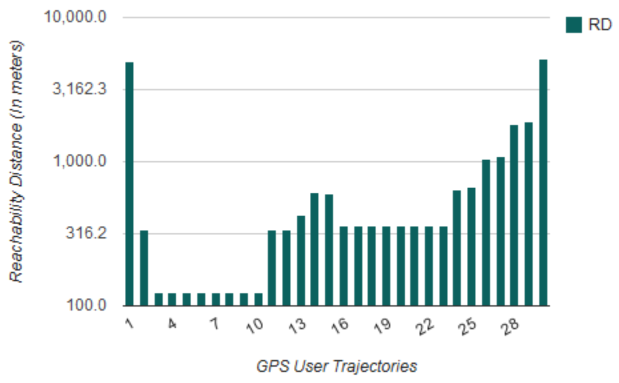
Contents

In this chapter, we present the results obtained by the evaluations of our approach in different scenarios. These evaluations were divided in three parts, which are: trajectory data acquisition, clustering algorithm and trajectory correlation algorithm. Since the main scientific contributions of this thesis are related to the clustering and trajectory correlation algorithms, we start presenting these algorithms. After that, we present the mobile application that was developed to perform the trajectory data acquisition process. In the following sections we present these parts and discuss the results obtained in each evaluation.

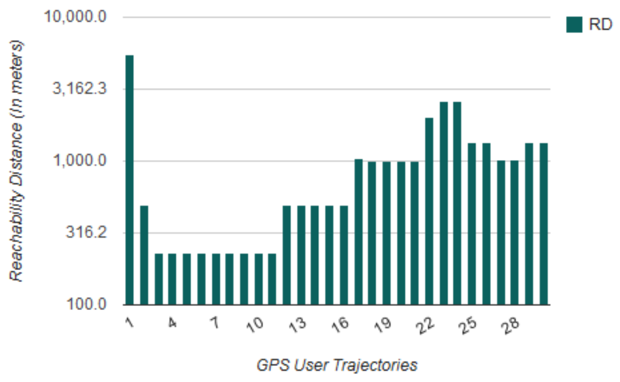
1 Clustering algorithm

To demonstrate the efficiency of the clustering algorithm we have applied our approach to two separate users, based on their registered trajectories in Dublin, Ireland. The overall approach can be summarized in three steps. First of all clustering is applied to individual user trajectories over a period of one month. A user's daily routine is a trajectory from home to work. After obtaining distinct groups an aggregated trajectory has to be chosen.

With the help of visualization and aggregation techniques, a best representative trajectory for each user is obtained. This aggregated trajectory obtained from several user trajectories is then compared to other users by applying our trajectory correlation algorithm. This will enable groups of users to share similar routes to increase geospatial social interaction. We now explain the different input parameters we have used in order to verify the results.



(a) User 1 (ϵ = 1000 & minNbs = 3).

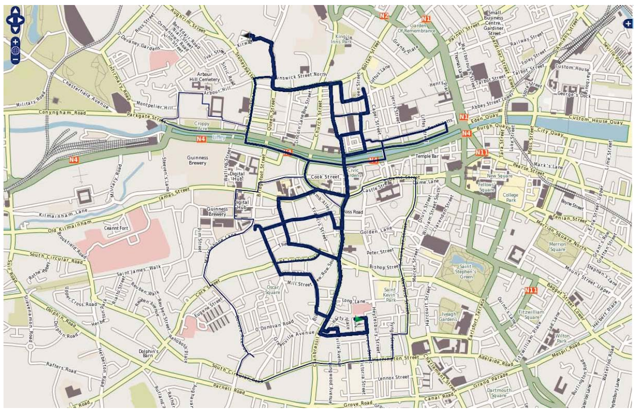


(b) User 2 (ϵ = 1000 & minNbs = 3).

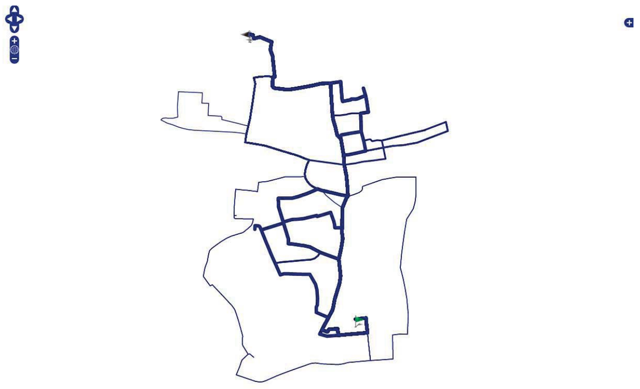
Figure 6.1: Reachability plots showing clustering structure.

OPTICS clustering algorithm requires two input parameters: distance threshold (ϵ) and minimum neighbors *(minNbs)*. The authors of OPTICS [1] suggest that the value of these two parameters have to be large enough to yield good results. We structured our experiment in a way that we choose a range of distance threshold values as well as minimum neighbors. For our scenario, we defined the distance threshold between 1000 meters and 15000 meters $\Rightarrow$ *(1000 $\leq \epsilon \leq$ 15000)*. Similarly, for minimum neighbors we selected a value of 1 up to 10 $\Rightarrow$ *(1 $\leq$ minNbs $\leq$ 10)*.

The experiment was run with a combination of values for both parameters.



(a) Three clusters showing distinct routes of User 1 (overlay on map).

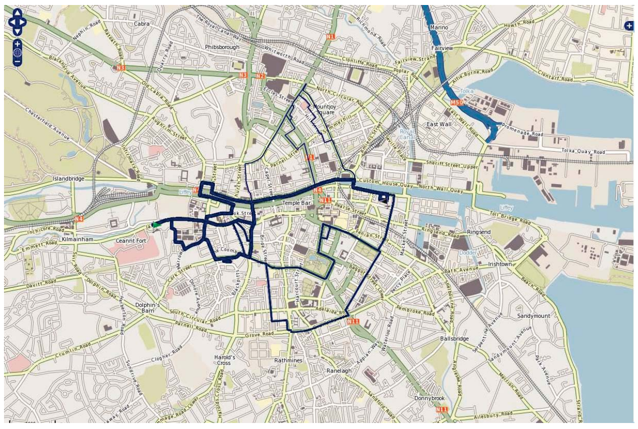


(b) Three clusters showing distinct routes of User 1 (without overlay).

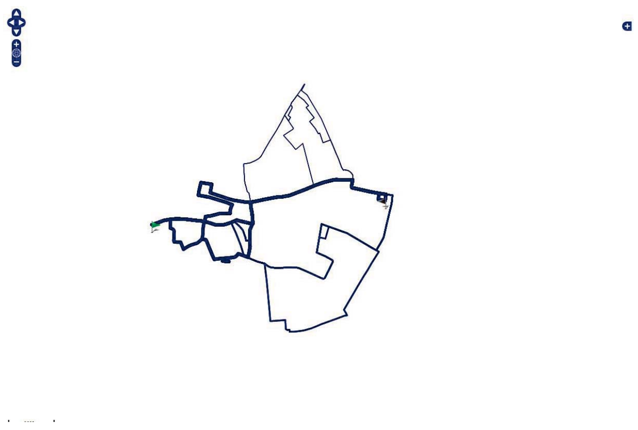
Figure 6.2: Clusters of user 1.

Based on the statistics and a range of reachability plots we obtained, we found the best combination of values ⇒ *(ϵ = 1000 & minNbs = 3)*. This condition revealed a satisfactory result in terms of the clustering structure from the reachability plots.

The reachability plots obtained are illustrated in Figures 6.1(a) and 6.1(b). The plots show re-ordering of objects (trajectories in the dataset) on x-axis while y-axis demonstrates the reachability distances between trajectories. Automatic cluster extraction techniques from a graph were presented in [1][189]. This data independent visualization provides analysts a high-level understanding of clustering structure. From these graphs clusters can be identified based on Gaussian-bumps or valleys. As a general rule the cluster starts from a



(a) Three clusters showing distinct routes of User 2 (overlay on map).



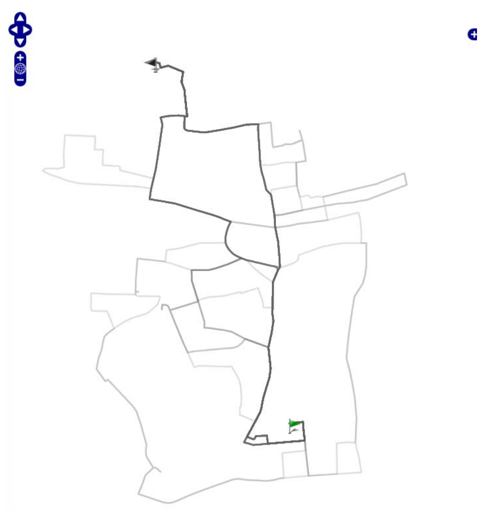
(b) Three clusters showing distinct routes of User 2 (without overlay).

Figure 6.3: Clusters of user 2.

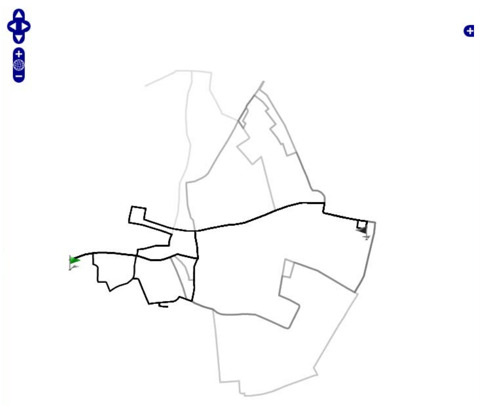
steep-down area and ends at a steep-up area.

Based on the first plot in Figure 6.1(a), we can clearly see that there are two dominant clusters in user trajectories (trajectory 2 to 13 and trajectory 14 to 25) shown by the valleys in the plot. The other cluster is a group of trajectories, which does not specifically form a valley however they are grouped together into one cluster. The second graph (see Figure 6.1(b)) also shows three clusters with varying cardinalities (trajectory 2 to 16, 17 to 22 and 23 to 30). In both the graphs, the first trajectory is considered as noise (see OPTICS algorithm [1]).

In Figures 6.2(a), 6.2(b), 6.3(a) and 6.4(b), the three clusters (from both graphs) are drawn in different styles. The representative routes for each cluster are drawn with different thickness for visualization purposes.



(a) Best representative aggregated user trajectories (user 1).



(b) Best representative aggregated user trajectories (user 2).

Figure 6.4: Best representative trajectories of users 1 and 2.

The clusters show three distinct routes both users adopted over a period of one month to travel from home to work. On average each user trajectory contains almost 100 points. The clustering structure also forms distinct groups based on a specific route on a specific day of the month. For example in Figures 6.2(a) and 6.2(b), cluster 2 holds trajectories starting from trajectory *14* to trajectory *25* that include *11* days routes. For this specific case we can acquire knowledge about the patterns related with a particular day of a week or a month. For example, if we observe the order in which the trajectories were recorded in case of cluster 2 we obtain *(1,2,3,4,7,8,9,12,13,14,15)*. We can apply heuristics and visualization techniques such as heat maps in order to gain more insights into user behaviors. As apparent from the above sequence user 1 always follows a similar or close route during at least three consecutive days of a month such as *(1,2,3)*, *(7,8,9)* and *(13,14,15)*.

After analyzing the clustering structure the next step is to find an aggregated trajectory or a best representative of a particular user route. For this purpose we have applied a simple yet interesting visualization technique. When all three clusters from both users are visualized using a single grey scale color scheme, it reveals the most frequent route adopted. The color has to be selected in a way that it must be transparent enough to visualize these changes. The phenomenon is illustrated in Figures 6.4(a) and 6.4, where user 1 and user 2 best representatives can be visualized and extracted respectively for further analysis.

2 Trajectory correlation algorithm

Since the clustering algorithm recognizes the best representative trajectory for each user, the trajectory correlation algorithm is executed. For this example, the algorithm firstly generates the MBRs for each best representative user trajectory and identifies the correlation between both MBRs. After that, it computes the Hausdorff distance of the points in the correlated area.

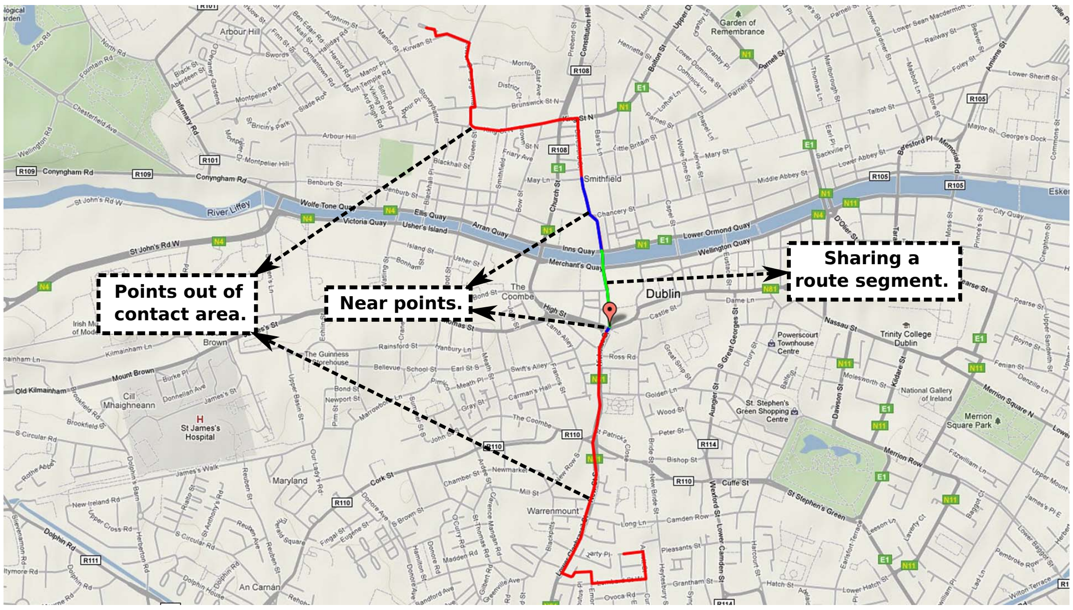


Figure 6.5: Best representative trajectory of user A in comparison to user B.

In order to present the accuracy and efficiency of our system we used a color-based scheme to represent the points in the same road segment, the near points and the points out of the correlated area. Figures 6.5 and 6.7 show the trajectory of the users A and B respectively with the colors representing the near points between them. The green color represents the same segment that is used by both users for their daily routines. The blue color denotes the possible points of interaction, which is in the correlated area among the

MBRs. Finally, the red color indicates the points that are out of the correlated area. Additionally, the system allows the generation of messages making use of the context information.

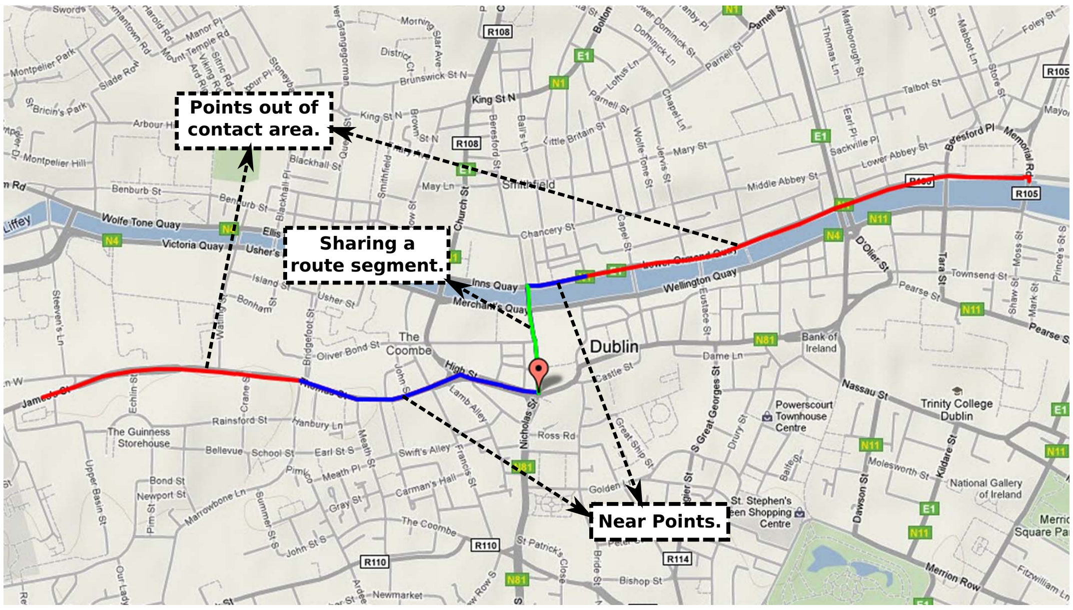


Figure 6.6: Best representative trajectory of user B in comparison to user A.

Based on the results, we observe that both analyzed users have common interests and our algorithm was able to identify the similar routines between them. These similarities are presented according to the situations described in the last chapter. Taking into account the different abstraction levels of our data model, these results illustrate the common segments of interest (SoI) between two users. This is possible due to the use of enriched information that is associated with each location in the database. In other words, each coordinate is registered in the database with its associated context information (e.g., postal address, time, speed of the moving object, weather, etc). Therefore, this enriched information facilitates the identification of similar segments and comes as an additional feature to increase the accuracy of the final result.

These results of our correlated trajectory algorithm are associated with two trajectories containing user routines at the same abstraction level of our multi-layer data model (see Figure 5.5). However, our algorithm also allows to identify similar user routines in different abstraction levels. That is possible due to our top down processing to find the similar interests between two trajectories. Firstly, we compare the highest abstraction levels of both users, taking into account the region around each trajectory. Since we find the correlated regions of both trajectories, we perform the comparison in the next layer for finding similar road segments between users' trajectories, which allows to obtain more details about the type of similarity (e.g. near, sharing). Finally, we carry out the comparison at the lowest abstraction level in order to find similarities between local places, such as: bakery X, hospital Y, and others.

Figure 6.7 illustrates the same comparison, but at a different (less detailed) abstraction level. The routine of user **B** is *Grenoble*, since his/her whole trajectory is within Grenoble (Level 1 of our data model). On the other hand, the routine of user A is represented by road segments (Level 2 of our data model). Based on that, the trajectory algorithm finds the similarities between the routines of user B (at the level of Trajectory of Interest (ToI)) in comparison to the routine of user **A** (at the level of Segments of Interest (SoI)). As the routine of user **A** is a subset of the set of the ToI represented by *Grenoble*, the map is shown with a green dot over *Grenoble*. Figure 6.7 presents an example of how a multi-layer data model can provide information at different abstraction levels.

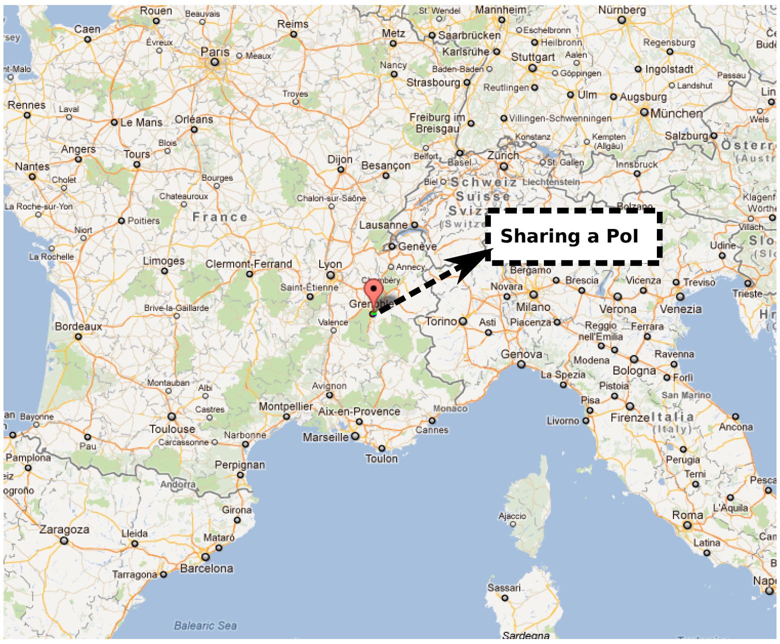


Figure 6.7: Best representative PoI (Grenoble) of user B in comparison to user A at a different abstraction level.

Since the similar user routines are identified, we can process the similarity analysis in the temporal data, comparing the time intervals in which the users have passed in a specific location (as presented in Chapter V). With the final results, we can provide complete information about users' similarities to the applications.

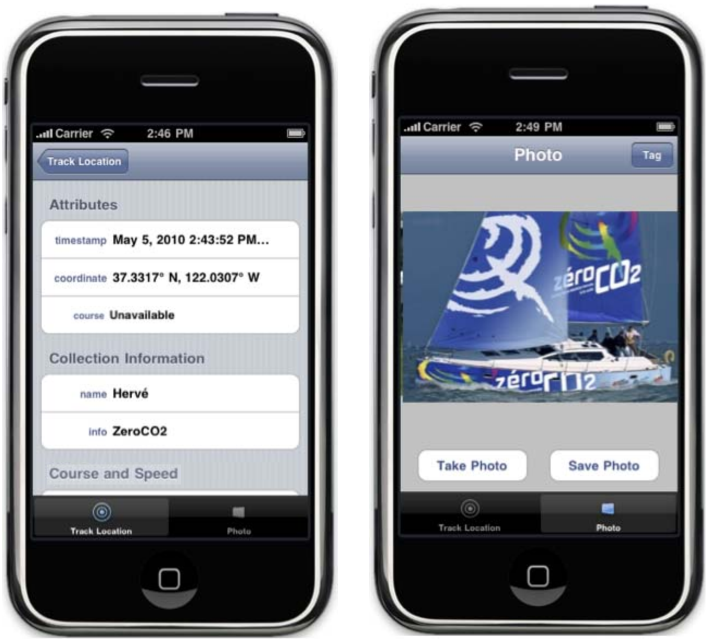
3 Trajectory data acquisition

To evaluate the efficiency of our trajectory data acquisition in a real situation, we implemented our proposal for the ZeroCO2 project [190]. We designed our system to be a digital logbook during a boat expedition around the Mediterranean Sea. The logbook, which was created as a book to record readings from the ship log [191], is an essential instrument to the navigation and has to be used daily. In general, the crew uses paper-based logbooks to register all information and, frequently, the information is collected from distinct equipments. Hence, we concluded that our system was able to create a complete logbook for this boat expedition. In addition, the challenging scenario of the sea added some problems involving the recurrent absence of Internet connection and the lack of battery charging.

Our system was responsible to track the trajectory followed by the boat, adding all context information to each registered coordinates. Although our system proposes the use of audio, video and photo as data, we used only photos for this first experiment in the project ZeroCO2. Taking into account this scenario, we face new challenges that have motivated us to improve the context-aware system proposed in the previous section.

The mobile application interface is shown in Figure 6.8. As we can observe, there are two main functions: the tracking mechanism and the digital camera. The tracking mechanism is responsible for registering the geographic coordinates to construct the trajectory. The interface shows the position, speed, date and, if Internet connection is available, wind speed and humidity. The digital camera takes a picture and, automatically, adds the context information to it. There is also the option "Tag" with which you can add the information manually.

An important result discovered during our tests is related to the use of metadata following some standard, such as Web Ontology Language OWL



Carrier 2:46 PM

Track Location

Attributes

timestamp May 5, 2010 2:43:52 PM...

coordinate 37.3317° N, 122.0307° W

course Unavailable

Collection Information

name Hervé

info ZeroCO2

Course and Speed

(a) Tracking mechanism. (b) Digital camera.

Figure 6.8: Tracking mechanism and digital camera.

[192]. Several solutions adopt this language to obtain inferred information about a context. However, it needs to add a large number of information in the metadata file to perform this task. Consequently, the mobile application generates several unused information into the metadata file, causing some problems of memory overflow in the mobile application. Therefore, we optimized the content of our metadata files registering only the relevant information. Besides that, we developed our own local parser to get the information of each tag and to infer about context information using the HTML parser.

The first evaluation was done during a travel around the Marseille coast. We ran this first test to calibrate the distance filter option and to execute the performance evaluation in the mobile phone. This option is responsible to define the detail level of the trajectory. We assigned the value fifty meters to the distance filter, which means that a position will be registered if it is higher than fifty meters in comparison with the last position registered. With the first results, we refined our system to the second test: a travel from Marseille to Ajaccio (Corsica Island).

Figure 1.8 shows the performance evaluation of our application in the mo-

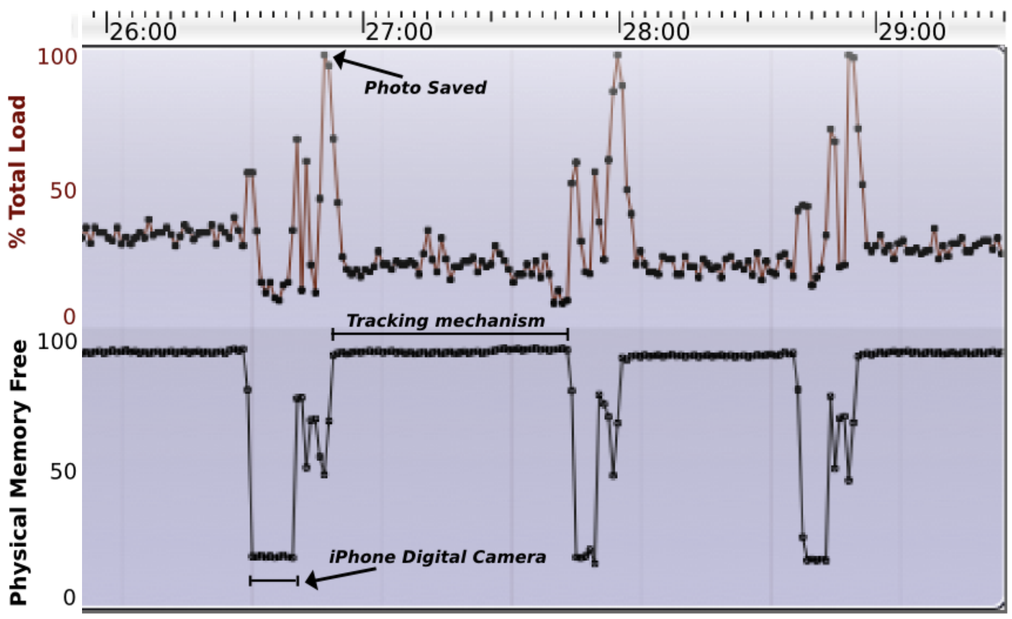
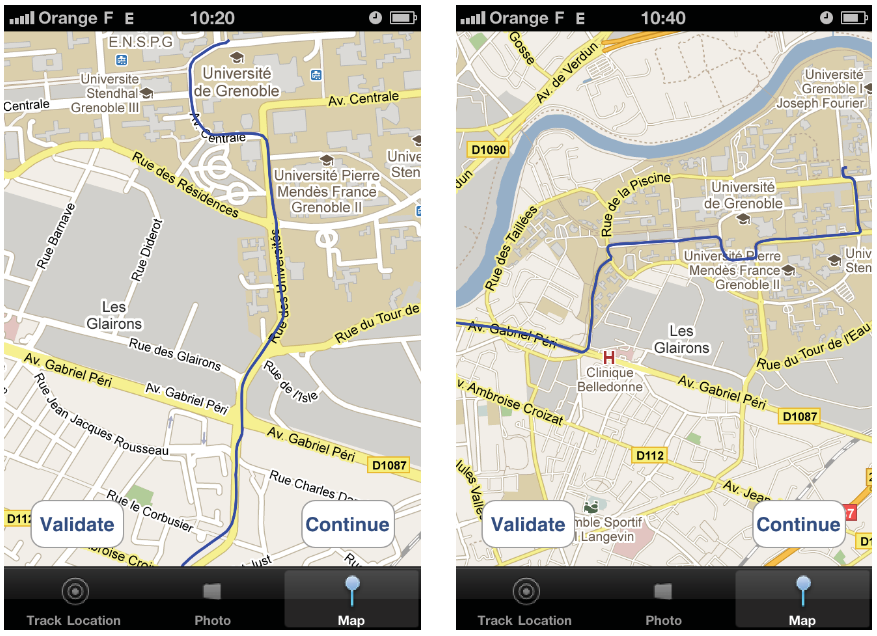


Figure 6.9: Physical Memory Free and Total Load in the iPhone.

bile phone during the interval from 26 to 29 minutes. The evaluation was conducted during the first tests, using the XCode Instruments [193] version 2.7. We observed that the *Total Load* (i.e., System and User) and the *Physical Memory Free* followed the same behavior while the mobile application functions were in operation. According to the results, the tracking mechanism requires approximately 10% of the memory and 25% of the processing to capture and register the positions. Likewise, while the iPhone digital camera is working, the memory used is approximately equal to 80% and the total load did not change. After taking the photo, the function *Save Photo* can be selected. When the *Save Photo* function is activated, the maximum load is used to associate and register all data and context information in the hard disk. Finally, the memory is cleaned when all data and information are associated and saved and the total load returns to follow the tracking mechanism. These results were important to guarantee that the user can use the application for a long time without stopping it due to memory or processing overhead problems.

Other important results are related to the mobile phone battery consumption during the trajectory registration. In the first test, when the distance filter had been configured to register each movement of the user, the iPhone battery level was down to 10% after 2 hours. After setting the distance filter to



(a) Daily trajectory of the user X. (b) Daily trajectory of the user Y.

Figure 6.10: Mobile Social Application.

fifty meters, the iPhone battery level was down to 10% after 3 hours. Another factor that can affect this result is the frequency that photos are captured.

The mobile application was one of the three modules developed in this project. Hence, more details about the usability, the desktop application and the server solution can be consulted in Appendix A.

This results achieved by the tests in a real scenario were important to observe the stability of processing and the memory use during the data acquisition process. Therefore, our solution can be used to efficiently obtain the necessary data (trajectory and context information) to our middleware.

Based on these results, we adapted our mobile application to capture trajectories and context information about each location in urban centers. Figure 6.10 shows the interface of our mobile social application, containing the trajectories of two users who registered their daily trajectories from home to work.

As we can observe in the example presented in Figure 6.10, both users live and work in different places. The user X registers his/her daily trajectory that represents his/her daily routine to go from home to work. The user Y does the same registering process. Then, each user visualizes the trajectory on the map. If this trajectory represents a good trajectory, the user validates it. Otherwise, the trajectory is rejected.

In terms of social networks, as previous mentioned, we enriched the database with the relations between users who register and share their trajectories. This data can be used to define different levels of relationships, such as: best friends, family, colleagues, friends, others. Along this line, we can define some controls to share personal trajectories only with users that have a certain level of relation with us. Therefore, in this evaluation, we used Facebook Developer Platform [194] to capture the relations between two best friends and assumed that they share their trajectories.

Analyzing the presented results and taking into account the use of context information to describe user routines, we conclude that our approach can be applied to a large number of applications, for instance: to offer a system that increases social interactions in real communities based on virtual communities (relations between friends in social network platforms); to develop a system that encourages rides among friends (car pooling); and others. Therefore, the data-sharing algorithm provides the information according to the requirements required by the application.

4 Conclusion

In this chapter, we presented some elements related to the evaluation of our approach. We started showing the implementation of our mobile application, which acquires trajectory data and context information of users. To validate the application, we tested it in a challenging scenario of a yacht traveling in the sea. After that, we adapted the application to register trajectories in urban canters, capturing all context information of each place by using reverse-geocoding techniques. Additionally, we developed a plug-in on a virtual community platform to receive the data containing user relations in social networks.

Next, we presented the evaluation of the OPTICS algorithm to discover the best representative trajectory of each user, which determines the user's daily routine. We explored the capabilities provided by this clustering algorithm to analyze user trajectories and extract relevant information from them. We focused on clustering and aggregating multiples trajectories generated by the same user in order to identify habits or preferences. The results showed that this clustering algorithm is efficient to the requirements of our middleware.

Finally, we introduced the results of our trajectory correlation algorithm, which finds similarities between multiple user trajectories based on each user preference and PoI. The results demonstrated that the similar routines between two or more users could be identified. Therefore, we conclude that our research provided interesting avenues for exploring Location-based Social Network (LBSN) applications. These avenues and the conclusion of this thesis are presented in the next chapter.