Weka-STPM: a Software Architecture and Prototype for Semantic Trajectory Data Mining and Visualization

Vania Bogorny [1] and Hercules Avancini [1] and Bruno Cesar de Paula [1] and Cassiano Rocha Kuplich [2] and Luis Otavio Alvares[2]

[1 1] Departamento de Informatica e Estatistica
Universidade Federal de Santa Catarina
Campus Trindade, Florianopolis, Brazil
e-mail: [vania, hercules, bclp]@inf.ufsc.br
[2] Instituto de Informatica, Universidade Federal do Rio Grande do Sul
Av. Bentro Gonçalves, 9500, Porto Alegre, Brazil
e-mail: [xxxxx, alvares] @ inf.ufrgs.br

Abstract. *Enormous quantities of trajectory data are collected from many sources, as GPS devices and mobile phones, as sequences of spatio-temporal points. These data can be used in many application domains such as traffic management, urban planing, tourism, bird migration, and so on. Raw trajectory data, as they are generated by mobile devices have very little or no semantics, and in most applications a higher level of abstraction is needed to exploit these data for decision making. Although several different methods have been proposed so far for trajectory querying and mining, there are no software tools to help the end user on semantic trajectory data analysis. In this paper we present a software architecture for semantic trajectory data mining as well as the first software prototype to both enrich trajectory data with semantic information and data mining. As a prototype we extend the Weka data mining toolkit with the module Weka-STPM, which is interoperable with databases constructed under OGC specifications. We tested Weka-STPM with real geographic databases, and trajectory data stored under Postgresql/PostGIS DBMS.*

1. Introduction

The use of mobile devices as GPS and cell phones has significantly increased in the last few years. This kind of devices and other technologies like RFID can generate sequences of space-time points, capturing the trajectories of moving objects. Trajectory data, acquired for operational level use, have increased enormously. However, very little has been done for the decisional level. One of the reasons is because trajectory data are usually generated as a sequence of (id, x, y, t) points, where id is the identification of the moving object, x and y are the geographic coordinates, and t is the time instant that this point was collected. It is not easy to obtain useful information or knowledge from this kind of data, since the data themselves are very limited and have no direct link to other information, except the *owner* of the trajectory, through the moving object identification, and information as speed, direction or acceleration. Clearly, there is a need to consider trajectories from a higher level of abstraction, instead of simple (id, x, y, t) points.

The main approach to obtain decision level information or knowledge from huge amounts of data is data mining. Several data mining methods for trajectories have been proposed in the last few years, like for instance [Tsoukatos and Gunopulos 2001,

Li et al. 2004, Laube et al. 2005, Cao et al. 2006, Gudmundsson and van Kreveld 2006, Nanni and Pedreschi 2006, Verhein and Chawla 2006, Lee et al. 2007, Li et al. 2010]. In general, these approaches have focused on the geometrical properties of trajectories, without considering an application domain. As a consequence, these methods tend to discover geometric patterns, which for some applications may not help the user in extracting more meaningful information. *Geometric patterns* are normally extracted based on the concept of dense regions or trajectory similarity. *Semantic patterns*, however, are independent of x, y coordinates, and can be located in sparse regions and may not have geometric similarity [Alvares et al. 2007]. For example, considering only the geometry of the trajectories one would discover a dense region where several trajectories meet or pass through, but without considering semantics, it would be difficult to discover the meaning of this place. By considering geographic information at least, it would be possible to discover that the dense place is a shopping center or a university, for instance. In a tourism application, considering the geographic location/position of hotels and touristic places it would be possible to discover patterns of moving objects going from hotels, sparsely located in a city instead of dense areas, to museums.

Semantics plays an essential role in several applications. It is becoming a strong research issue in GIS (Geographic Information Systems), and several works have been proposed, as for instance, [Janowicz et al. 2008] and [van Hage et al. 2010].

Semantic patterns cannot be obtained in post-processing steps, after the patterns have been generated, because many semantic patterns will never emerge if the mining algorithm relies only on the geometrical properties of trajectories.

Recently, Spaccapietra [Spaccapietra et al. 2008] proposed the first model to deal with trajectory data from a semantic point of view. This model is based on the concepts of *stops* and *moves*. Stops are the important places of a trajectory from an application point of view, where the moving object has stayed for a minimal amount of time. Moves are the subtrajectories between stops or between the starting point of the trajectory and the first stop or between the last stop and the ending point of the trajectory. Considering this model, a trajectory is a sequence of stops and moves. Figure 1 illustrates this concept. Figure 1(a) is a sample point trajectory with no semantics, while Figure 1(b) represents the same trajectory considering a tourism application, where we can see that the trajectory of the moving object starts at an airport and stays there from 8AM to 8:30AM. Then, it goes to Ibis Hotel where it arrives at 9:30AM and it leaves at 10AM. After that, the moving object goes to the Eiffel Tower, where it stays from 11AM to 1PM. Finally, it goes to the Louvre Museum and stays there from 2PM to 6PM.

To transform a sample trajectory into a semantic one (sequence of stops and moves) is not a trivial task. To do it, one should consider several aspects as the type of application and what geographic background information in the region of the trajectory is important for the specific problem in hand. Indeed, both dimensions of space and time have to be considered in the process.

Complex data preprocessing and data transformation functions are necessary to add semantics to trajectories in order to facilitate their analysis and knowledge extraction from the user's point of view. Another problem in trajectory knowledge discovery is that as far as we know there are no available toolkits with friendly and graphical user interfaces to help the user to perform the whole knowledge discovery process on trajectories of

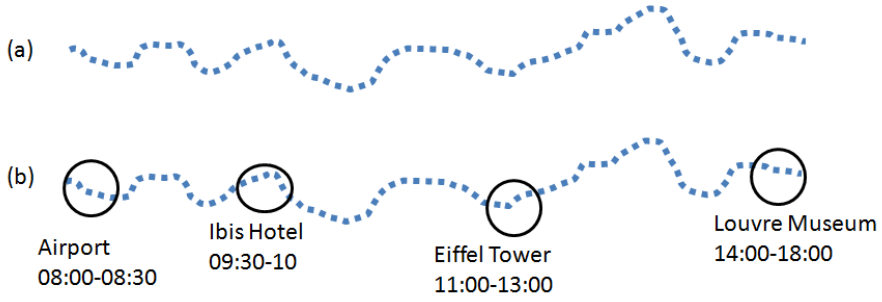

Figure 1. (a) Sample trajectory and (b) semantic trajectory

moving objects, including their semantic enrichment.

In order to reduce these problems, this paper presents a general software architecture for semantic trajectory pattern mining and a free software prototype called Weka-STPM (Semantic Trajectory Preprocessing Module), that has been constructed and integrated to Weka [Frank et al. 2005], being available at [Bogorny 2010] for download. Weka is a free and open source data mining toolkit developed at Waikato University for traditional data mining only. It is the most well known and used classical data mining toolkit in academic research. Initially, in 2006, we extended Weka for the automatic preprocessing of geographic static data for data mining [Bogorny et al. 2006]. In this paper we propose the extension of that work to support spatio-temporal data, but not only preprocessing, but the semantic enrichment, data mining, post processing and pattern visualization of data that move in space and time. It is a completely new tool, and as far as we know it is the first tool for semantic trajectory data mining that covers the complete knowledge discovery process. As a module of Weka, it allows the user to directly apply the mining algorithms available in Weka to mine semantic trajectories.

1.1. Related Works and Contribution

Most trajectory data mining works, as classical data mining methods, focus on the mining step itself, without thinking about the whole knowledge discovery process, that includes data preprocessing, data transformation, data mining and postprocessing. As a consequence, a lot of effort from the user is needed to manually prepare the data for data mining, as for instance transform the data into different space and time granularities. In trajectory data mining this problem increases because space and time have to be considered.

Another problem is that several trajectory data mining algorithms consider only one dimension, either space or time. In [Lee et al. 2007], [Lee et al. 2008b] and [Lee et al. 2008a], for instance, mining algorithms are proposed to respectively extract groups of subtrajectories, classification patterns, and trajectory outliers, but both dimensions of space and time are considered only in [Lee et al. 2008a].

Besides considering only the mining step itself, trajectory data mining algorithms as for instance [Gudmundsson and van Kreveld 2006, Laube et al. 2005, Li et al. 2004, Nanni and Pedreschi 2006, Tsoukatos and Gunopulos 2001] focus on the trajectory samples, without considering any context or semantic information, making the pattern interpretation very difficult from the user point of view.

Among these works, only a few as [Trasarti et al. 2010] [Baglioni et al. 2009]

consider context information that helps the user to understand trajectory patterns. Another problem from the user´s perspective is that only a few works are implemented in toolkits. To the best of our knowledge only the works of [Trasarti et al. 2010] [Baglioni et al. 2009] are implemented in a toolkit, but these works are on the top of Oracle, which in turn runes on Hermes [Pelekis et al. 2006] moving object database. The last is also a prototype but not available for public use.

-In the last few years our group has tried to make contributions for the trajectory data mining field taking the user into account: trying to extract more meaningful and easily understandable patterns, aiming to cover the complete discovery process and providing a tool. In [Alvares et al. 2007] and [Palma et al. 2008] we developed methods to enrich trajectories with more semantic information. In [Alvares et al. 2010] we presented a short introduction of a tool for mining one single level trajectory patterns. In [Bogorny et al. 2009] we presented a data mining query language that is able to extract semantic trajectory patterns at different space and time granularities. These works, however, still have some problems. In [Palma et al. 2008] speed has not been considered directly to find important parts in trajectories, but a proportion of the points that should belong to clusters defined with the area and minimal time parameters, which are not very intuitive from the user's point of view. In [Bogorny et al. 2009] we have not treated data cleaning/reduction, pattern analysis and visualization. The focus was on the data mining query language only.

In this paper we integrate in one framework and in one free and open source toolkit, several pieces that are necessary to support automatic multiple-level semantic trajectory knowledge discovery. In summary, we make the following contributions in relation to our previous works:

- We extend the work of [Alvares et al. 2010] with new data cleaning/reduction methods to improve the performance of the algorithms IB-SMoT and CB-SMoT.
- We add a data preprocessing module to automatically add a standard trajectory identifier and time attribute, so that the user does not have to do this manually.
- We change the method CB-SMoT [Palma et al. 2008], by using the average speed of the trajectory as the parameter to find low speed regions in trajectories, simplifying the information to the user and improving the results of the discovered clusters.
- We store the geometry of the stops for their later spatial or spatio-temporal analysis. In previous works we used only the geometry of the spatial feature that intersected a stop, but since clusters generated by the method CB-SMOT may not intersect any spatial feature, we store the geometry of the cluster for later analysis.
- We provide a graphical user interface to spatially visualize the data and the patterns, what is not provided in [Bogorny et al. 2009], where the focus was to provide a data mining query language, and not a data mining toolkit.
- We present a methodology to perform semantic trajectory knowledge discovery.
- The main contribution of this work is the architecture and a tool, that provides to the user, support to perform the complete knowledge discovery process in spatio-temporal data, covering data preparation, data mining, and visual pattern analysis in postprocessing.

1.2. Scope and Outline

The scope of this work is limited to the proposal of a general architecture and software

prototype for spatio-temporal data mining. The architecture supports a new methodology that covers the whole knowledge discovery process of trajectories, including data cleaning, semantic enrichment of the data (which is a task not present in classical data mining), data transformation to multiple space and time granularities, data mining, and pattern visualization. This architecture is implemented in a well known open source software named Weka.

The remainder of the paper is structured as follows. Section 2 presents some basic concepts. Section 3 presents the proposed architecture, while Section 4 describes the Weka-STPM module. Section 5 shows some results of the proposed software evaluating the main steps of the discovery process with STPM. Finally, Section 6 presents the conclusion of the paper and future works.

2. Basic Concepts

Trajectory data generated by mobile devices are raw data, and therefore they may be cleaned, and sometimes reconstructed in order to be explored by the user to extract some interesting information.

Apart from organizing these data, the main step for data mining is to add meaning to trajectories according to an application point of view. We do that by using the concepts of stops and moves. Two algorithms have been developed for this purpose. The first one, introduced in [Alvares et al. 2007], considers the intersection of a trajectory with the user-specified relevant feature types (relevant object type or candidate stop) for a minimal time duration, which is called IB-SMoT (Intersection-Based Stops and Moves of Trajectories).

In general words, the algorithm verifies for each point of a trajectory T if it intersects the geometry of a relevant feature type R_C. In affirmative case, the algorithm looks if the duration of the intersection is at least equal to a given threshold Δ_C. If this is the case, the intersected candidate stop is considered as a stop, and this stop is recorded. If a point does not belong to a subtrajectory that intersects a candidate stop for Δ_C, it will be part of a move.

Figure 2 (a) illustrates this method. In this example, there are four candidate stops with geometries R_{C_1}, R_{C_2}, R_{C_3}, and R_{C_4}. Let us consider a trajectory T represented by the space-time points sequence $\langle p_0, \ldots, p_{15} \rangle$, and $t_0, \ldots, t_{15}$ are the time points of T. First, T is outside any candidate stop, so we start with a move. Then T enters R_{C_1} at point p_3. Since the duration of staying inside R_{C_1} is long enough, R_{C_1} is the first stop of T, and $\langle p_0, \ldots, p_3 \rangle$ is its first move. Next, T enters R_{C_2}, but for a time interval shorter than Δ_{C_2}, so this is not a stop. We therefore have a move until T enters R_{C_3}, which fulfills the requests to be a stop, and so R_{C_3} is the second stop of $T$and $\langle p_5, \ldots, p_{13} \rangle$ is its second move.

The second algorithm is called CB-SMoT [Palma et al. 2008], and is a clustering method based on the variation of the speed of the trajectory. The intuition of this method is that the parts of a trajectory in which the speed is lower than in other parts of the same trajectory, correspond to interesting places. In [Palma et al. 2008] CB-SMoT was based on the method DBSCAN [Ester et al. 1996] to find the clusters, considering as input a minimal number of trajectory points that should belong to a cluster (the area parameter) and a minimal time. In the trajectory domain, however, it might be difficult to estimate the minimal area (number of points) to define a cluster. Therefore, in this paper we extend

the work of Palma replacing these parameters, which are difficult to define from the user's point of view, and make use of two new parameters: a maximal average speed for a cluster, that is specified as a percentage of the average speed of the trajectory, minimal time (minTime) as the minimal duration of the low speed subtrajectory to constitute a cluster, and maximal speed limit, which is a parameter to avoid high speed points in a cluster.

-In a first step the algorithm discovers the low speed clusters. In the second step, the algorithm identifies where these potential stops (clusters) found in the previous step are located, considering the candidate stops. In case that a potential stop does not intersect any of the given candidate objects, it still can be an interesting place. In order to provide this information to the user, the algorithm labels such places as *unknown stops*. Unknown stops are interesting because although they may not intersect any relevant spatial feature type given by the user, a pattern can be generated for unknown stops if several trajectories stay for a minimal amount of time at the same unknown stop. In this case, the user may investigate what this unknown stop is.

Figure 2 (b) illustrates the method CB-SMoT. Considering the trajectory $T = \langle p_0,\ p_1,\ \ldots,\ p_n \rangle$ represented in Figure 2 (b), the first step is to compute the clusters. Suppose that T has 4 potential stops, the clusters G_1, G_2, G_3 and G_4, represented by ellipsis. In this example the user has specified 4 candidate stops, identified by the rectangles R_{C1}, R_{C2}, R_{C3} and R_{C4}. The cluster G_1 intersects the candidate stop R_{C1} for a time greater than Δ_{c1}, then the first stop of the trajectory is R_{C1}. The same occurs with the cluster G_2, considering R_{C3}, which is the second stop of the trajectory. The clusters, G_3 and G_4 do not intersect any candidate stop. Therefore, G_3 and G_4 are unknown stops.

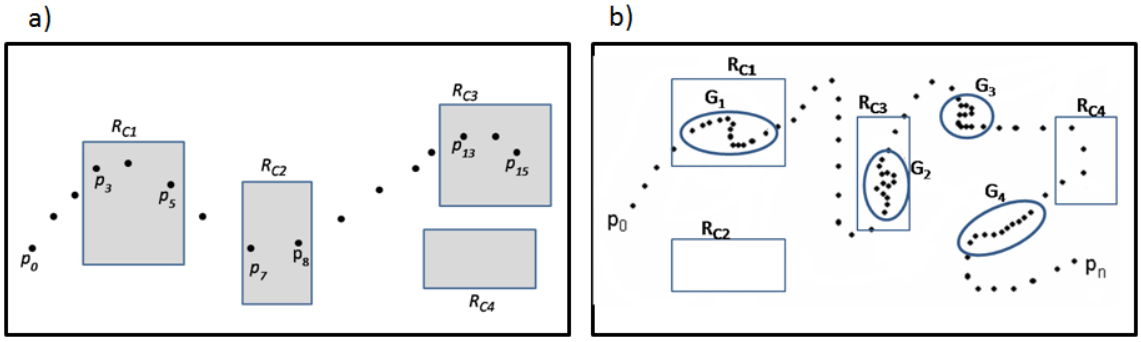

Figure 2. a) Example of the IB-SMoT method, and b) Example of the CB-SMoT method

The two methods cover a relevant set of applications. IB-SMoT is interesting in applications where the speed is not important, like tourism and urban planning. In this kind of application, the presence or the absence of the moving object in relevant places is more important. However, in other applications like traffic management, CB-SMoT, which is based on speed variation, would be more appropriate.

3. The proposed Architecture

Figure 3 presents an overview of the proposed framework. On the bottom of the framework are the data repositories, including trajectory raw data, spatial data, semantic trajectories, patterns and background knowledge. Raw trajectory data and spatial data (geographic data about the domain) are the input data. The preprocessing methods will add

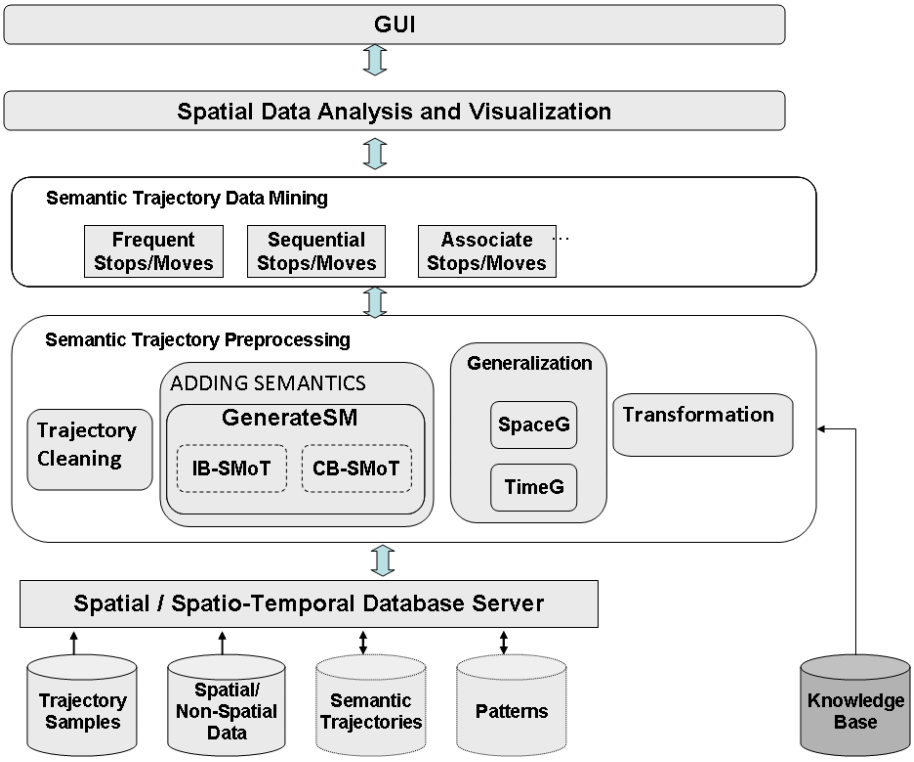

Figure 3. General Software Architecture

semantic information to the raw trajectories and generate a database with semantic trajectories. After mining, the patterns can also be stored in a database for post-processing analysis.

In the center are the data preprocessing and data mining modules. On the top of these modules we developed a layer for visual data analysis and interpretation, as well as a graphical GUI to provide to the user several facilities for preprocessing, mining and visualizing trajectory data and trajectory patterns. Since we are talking about spatial and spatio-temporal data, there is a need for an interface to visualize the data in bi-dimensional space, in the form of a map. Therefore, the software architecture provide this interface where the user can overlay different layers of information.

Semantic trajectory preprocessing has four main modules: Trajectory cleaning, adding semantics, aggregation in higer granularity levels (generalization) and transformation to the data mining algorithm input format.

3.1. Trajectory Cleaning

The Trajectory Cleaning module performs many verifications over the trajectory dataset in order to eliminate noise, what is very common in this kind of data, and to assure that the trajectory dataset is in the format required by the Adding Semantics module.

Some of the main verifications that can be performed over trajectories include, but are not limited to:

- the calculated speed between two consecutive points should not be greater than a specified threshold;
- the trajectory points should be in a temporal order;

- the trajectories should not have more than one point with the same timestamp;
- each trajectory should have a given minimum number of points. If there is a need for trajectory reconstruction, this is performed in this module.

Another lack of cleaning issues is related to data reduction. One of the main problems of data generated by GPS devices is that by default points are collected every second. For some analysis, like the semantic enrichment, this massive volume of data makes the algorithms to perform very poorly. Therefore, there is a need to reduce the amount of data, as for instance, to aleatory remove a certain percentage of the original data or simply among every two points, remove one. This simple technique may not affect the result of trajectory data analysis but may reduce the processing time in about 50%.

In this step the user can also reconstruct trajectories with either spatial or temporal gaps. For instance, a trajectory that has the points collected every second, but that has a gap of one hour between two consecutive points: it should be considered as one trajectory or as two?

In summary, any treatment of data before starting the semantic enrichment of patter mining, should occur in this step

3.2. Adding Semantics

To prepare trajectory data for data mining, the main step is to add semantics to these trajectories. We do that by using the concepts of stops and moves and the algorithms IB-SMoT and CB-SMoT. Notice that any new method to add semantics to trajectories can be plugged into the *Adding Semantics* module.

The output of the *Adding Semantics* module is a set of *semantic trajectories*, stored in the database as two relations of stops and moves. Stops are represented by the following attributes:

```
STOP (Tid integer, stopid integer, stop_name varchar,
      stop_gid integer, start_time timestamp, end_time timestamp,
      the_geom geometry)
```

where:

- *Tid*: is the trajectory identifier.
- *stopid*: is the stop identifier. It is an integer value starting from 1, in the same order as the stops occur in the trajectory. This attribute represents the sequence as stops occur in the trajectory.
- *stop_name*: is the name of the relevant spatial feature type (geographic database relation, e.g., hotels, restaurants) where the moving object has stayed for the minimal amount of time. In case the trajectory does not intersect any spatial feature, the stop name will be *unknown*.
- *stop_gid*: is the identification of the instance (e.g. Ibis) of the spatial feature type (e.g. Hotel) in which the moving object has stopped.
- *start_time*: is the time in which the stop has started, i.e., the time that the object enters in a stop.
- *end_time*: is the time in which the moving object leaves the stop.

- *the_geom*: is the geometry of the stop.

In a relational data model, the attributes *stop_name* and *stop_gid* are a foreign key to a geographic relation, allowing spatial queries. The relation of moves has the following schema, with four attributes more than the stop relation:

```
MOVE (Tid integer, Mid integer, stop_name1 varchar,
      stop_gid1 integer, stop_name2 varchar,
      stop_gid2 integer, start_time timestamp,
      end_time timestamp, the_move multiline )
```

where:

- *Mid*: is the identifier of the move in the trajectory. It starts with 1, in the same order as the moves occur in the trajectory.
- *stop_name1 and stop_name2* : are the names of the spatial feature type in which the move respectively starts and finishes.
- *stop_gid1 and stop_gid2*: are the identifier (feature instance) of the start and end stop of the move.
- *the_move*: is the set of points that corresponds to the spatial properties of a move.

3.3. Generalization

A key issue in data mining is the aggregation of data at higher abstraction levels. This is performed by the *Generalization Module*. The stops and moves are generated at the lowest granularity level (instances of objects for the spatial dimension and timestamp for the time dimension). However, it is almost impossible to find patterns at this granularity level. It is very difficult for some events to occur at the same second, as for instance, several trajectories arriving at home at exactly the same moment. To overcome this problem, in our framework the user can specify different granularity levels, for instance to consider intervals of one hour. This means that one event that occurs at 18:10PM will be considered in the same period as another that occurred at 18:20PM. Depending on the application, the time granularity can be year, month, week, day, hour, etc. Analogously, the space granularity can change, including even the semantics of the object. For instance, considering hotels and touristic places at a low abstraction level like IBIS hotel and Eiffel Tower, would lead to the discovery of more specific patterns, like a move *from IBIS Hotel to Eiffel Tower*. Considering both at a higher level of abstraction would lead to higher level patterns like *from Hotel to Touristic Place*.

Furthermore, the user can specify what should be considered in the mining step: (i) only the space dimension; (ii) the space and the time of the beginning of the stop or move; (iii) the space and the time of the end of the stop or move; or (iv) the space and the time of both begin and the end of the stop or move.

Stops and moves have both space and time information transformed by the functions *spaceG* and *timeG*. For the granularity transformation these functions may use background knowledge as concept hierarchies or pre-defined time granularities, which are stored in the *knowledge base*. Details about space and time generalization for trajectories are available in [Bogorny et al. 2009].

3.4. Transformation

The Transformation module uses as input the relations of stops and moves, stored in the database, generated by the *Generalization* module, and generates an output file in the format required by a specific mining algorithm or tool. Although each tool can use a specific format, there are two main types. One, the most used, can be seen as an horizontal type, where each line corresponds to one trajectory and each column corresponds to one stop or move. The other type is a vertical one, where each line corresponds to a stop or move of a trajectory. This second type is mostly used for sequential pattern mining, so both formats should be provided.

After the transformation, the user can apply different data mining algorithms to compute frequent stops/ moves, sequential stops/moves and association stops/moves.

4. Weka-STPM

Weka is a free and open source data mining toolkit developed at the university of Waikato[Frank et al. 2005]. This tool has originally been build for classical data mining, and is the most well known and used data mining tool in academic comunities. Our research group has extended this tool to preprocess spatial data [Bogorny et al. 2006], and is being extended in this paper to spatio-temporal data preprocessing, data mining and post processing.

The prototype presented in this paper is an evolution of Weka-GDPM [Bogorny et al. 2006], called Weka-STPM (Semantic Trajectory Preprocessing Module), Within Weka-STPM the user is able to connect to several DBMS through JDBC, like for instance Oracle, Postgres/PostGIS, and MySQL. Weka-STPM follows OGC standards [OGC 2008], and therefore becomes interoperable with several spatial and spatio-temporal DBMS, such as Hermes [Pelekis et al. 2006], which is a spatio-temporal DBMS developed on the top of Oracle Spatial. Weka-STPM has a new module that is used to visualize spatial data and spatio-temporal patterns. Weka is a free and open source non-spatial data mining toolkit developed in Java. It has a non-spatial data preprocessing module named weka.Explorer, where data can be obtained from a database, a web site, or an arff (input text file in the format required by Weka) file. The module STPM is fully integrated into Weka in order to automatically access the database and add semantics to trajectory data. Weka-STPM is an extension of Weka for spatio-temporal data.

In order to support STPM, we first extended the Weka database connection interface. We added the button *Trajectory Data*, which calls the STPM module, shown in Figure 4. As can be observed in Figure 4, the user provides the database schema name, in this example called public, and STPM loads all geographic database tables to the boxes *Trajectory Table* and *Relevant Features*. When the user chooses the database schema, Weka-STPM will look in the database relation *geometry_columns, standardized by the OGC*, and retrieves the content of the attribute f_table_name that contains the name of all relations that have a geometric attribute. All spatial relations in a geographic database that have a spatial attribute have a record in the relation geometry_columns.

Since trajectory data may have several different formats, the software needs to know which attribute represents the geometry and the time. In other words, the trajectory table should have an attribute called *Tid*, of type integer, that is used as the trajectory

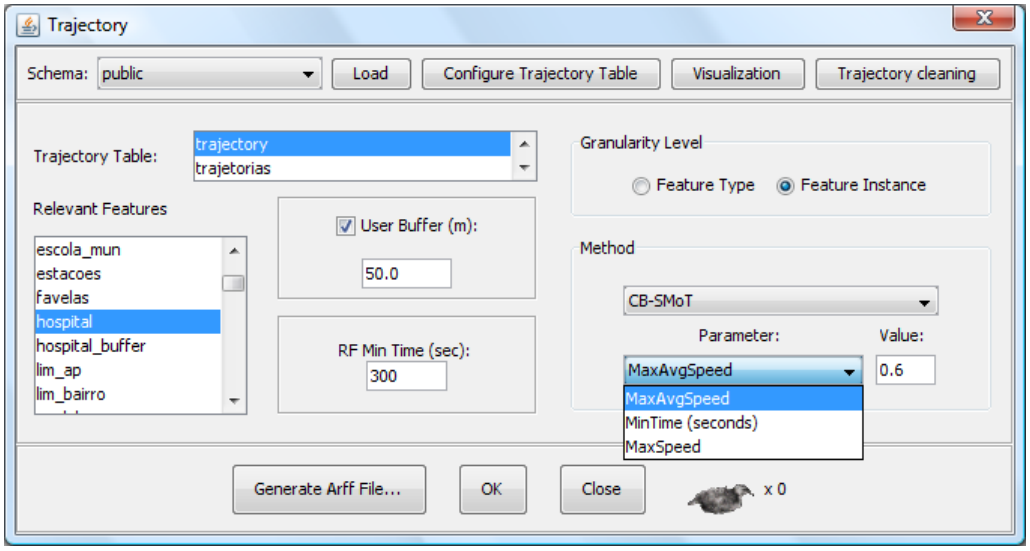

Figure 4. STPM main interface

identification; and the attribute that stores the time information should be called *time*, and be of type timestamp. In order to either generate or to transform these attributes we added the button *Configure Trajectory Table*, that opens the interface shown in Figure 5 (left). It presents some examples of scripts that can be adapted to do these transformations and that can be executed from this interface, avoiding that the user has to move to a database management system to do this step manually. So the system helps the user to provide this information.

The *Trajectory Cleaning* button calls the interface shown in Figure 5 (right). From this interface the user may clean trajectory data, choosing one of three methods implemented for this task. The first allows the user to remove all points that have its speed much higher than the average speed of the trajectory. For instance, if the average trajectory speed is 100 km/h and points with speed above 200km/h (Max speed) should be considered noise, than all points with speed above max speed are removed from the trajectory dataset. The second cleaning method is flexible, where the user can also choose how many points to remove from the dataset. For instance, if he wants to remove one point between every two points, then one point remains an the next is removed. This method reduces 50% the amount of trajectory points, which is quite good for GPS data that generate a point every one or two seconds. The third option allows the user to remove a percentage of points from trajectories. For instance, if the user wants to reduce the amount in 20%, then if a dataset has one million points, 200 hundred thousand points will be aleatory removed.

After configuring and cleaning trajectory datasets, finally the user can add semantics to trajectories, choosing the target trajectory table and the relevant spatial feature types of interest for the application (candidate stops) with the respective minimum time duration (*RF Min Time*). For each different relevant feature type a minimal amount of time can be specified (RF Min Time (sec)). The parameter *User Buffer* is the radius (size) of the zone around relevant features represented by points or lines, to overcome spatial imprecision. For instance, if a school is represented as a point, a trajectory will never intersect a school area, unless we use a buffer around schools.

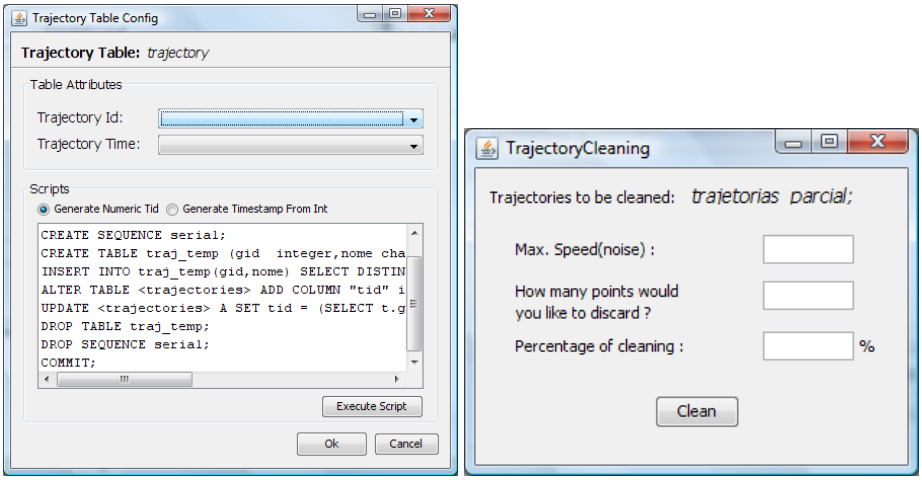

Figure 5. (left) Configure Trajectory Table and (right) Trajectory Cleaning

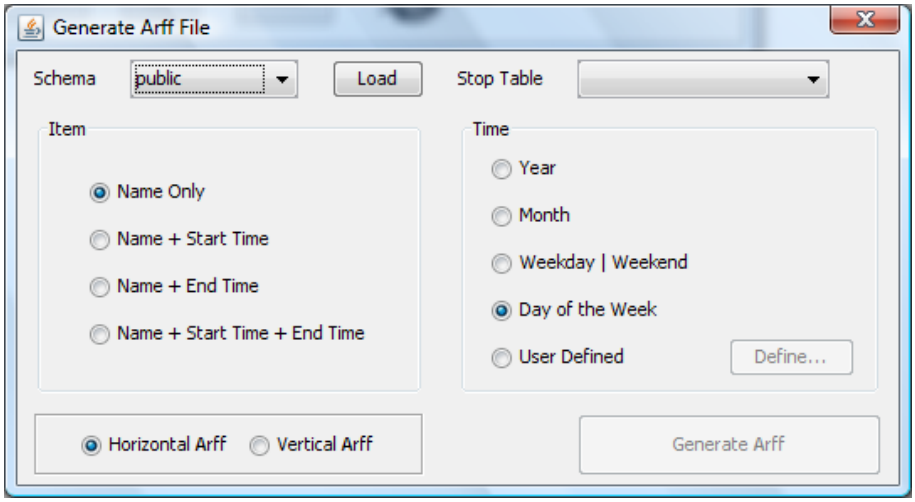

Figure 6. Generalization and Transformation

On the right side of the interface shown in Figure 4, the user can choose the most appropriate *method* to generate semantic trajectories (IB-SMoT or CB-SMoT) and define the respective parameters. The parameters for CB-SMOT are the maximal average speed to generate a cluster (MaxAvgSpeed), the minimal amount of time for a point to belong to a cluster (MinTime), and the maximal speed limit for a point to belong to a cluster (MaxSpeed). The output of the STPM module are two relations, stops and moves, with the structure described in previous sections. Stops and moves are computed by pressing the ok button. Stops are computed on the spatial granularity of instance or type. For instance, feature type granularity will label stops with the name of the relevant spatial feature type (e.g. Hotel), and the feature instance granularity will label the stops with the instance of the relevant spatial feature (e.g. IBIS Hotel). If stops are generated at the level of feature type, they cannot be converted to instance anymore, unless the user runs the stops computation algorithm again.

An important remark is that the user may want to extract stops several times, using different methods or changing the input parameters. Therefore, the name of the stops

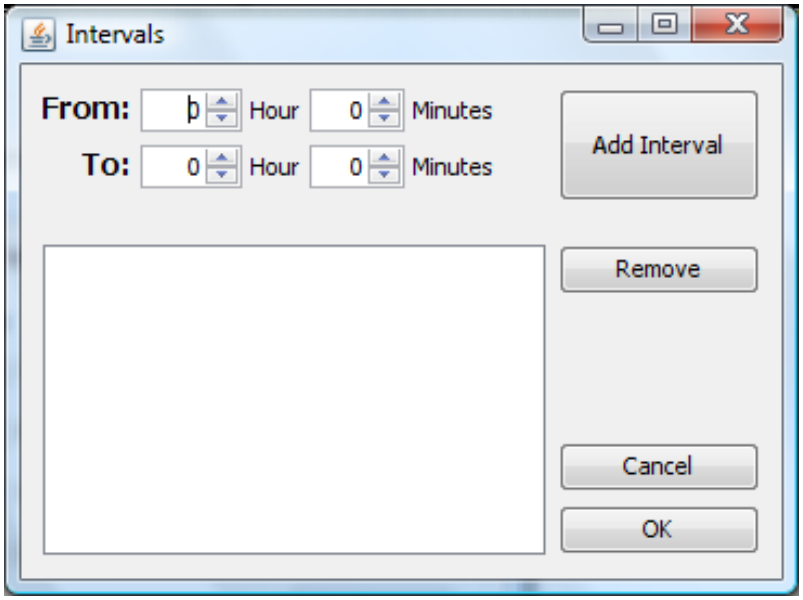

Figure 7. User Defined Time aggregation

relation is concatenated to the parameters defined for extracting the stops. For the method CB-SMOT, for instance, the stops relation will have on its name the maximal average speed, the minimal time (in seconds) and the maximal speed for a point to belong to a cluster. For example, the name stops_as_0_7_mt_300_ms_1_5 is a relation of stops computed with maximal average speed 0.7, minimal time 300 seconds and maximal speed limit 1.5. This avoids that for every stops computation the user has to manually change the table name in the database, and this name helps to remember the specified parameters for each experiment.

Once stops and moves are extracted, i.e., semantic information has been added to trajectories, the user can generate the arff file for trajectory pattern mining. This can be done by the *Generate ARFF FILE* on the main interface. This operation will activate the interface shown in Figure 6, where the user can specify different data granularities before generating the arff file. After choosing the stops table from where patterns will be extracted, the user can define the item to be mined. He/she can choose if the item to be mined will have only space information (name only) or both space and time information. In the last case he/she can still choose if the time information will include the begin, the end or both begin and end time of the stop. On the right side the user can choose the time granularity, pre-defined as year, month, weekday/weekend, days of the week, or user defined. For user defined time granularities the GUI shown in Figure 7 is provided. Finally, having defined the data granularity the user can generate the arff file either in the vertical or horizontal format, that will be stored into *weka/data* (Figure 6). Once the file is generated, the native data mining algorithms available in Weka can be applied.

A new module that we have implemented in Weka-STPM is an interface for the visualization of trajectories and spatial data (visualization button on Figure4). Figure 8 shows an example where the user has selected two relations from the database for visualization, a set of trajectories (trajetorias_parcial) and the districts of the city of Rio de Janeiro (lim_bairro).

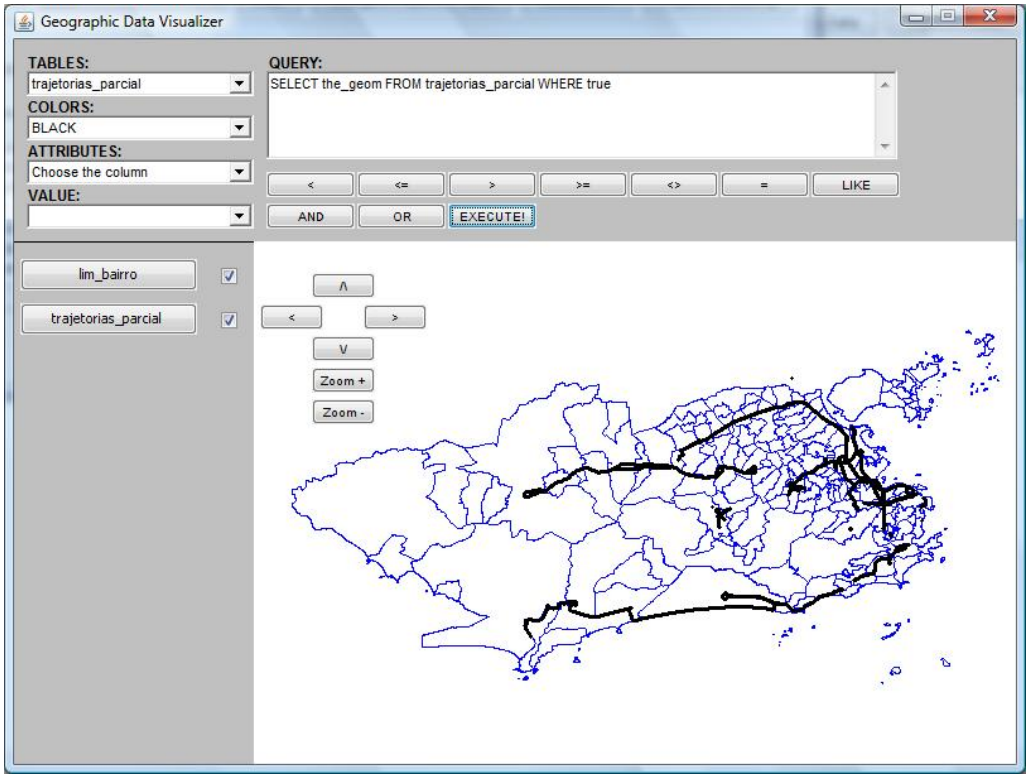

Figure 8. Spatial data visualization interface

At the visualization interface the user may choose the database relation to visualize on the left side. He can also choose the color for visualization. By simply selecting the relation and the color, its geometry will be shown in the center of the interface. For each selected relation, the user may want to filter the query. In case he may want to query by a specific attribute, he can choose the attribute of the relation in the combo and for the selected attribute, all possible values for that attribute are shown in the combo *value*. The user can build the query by choosing the attribute, using the logical operators and choosing the value of the selected attribute. Based on these parameters the interface will automatically build the SQL query.

-It is important to emphasize that the possible queries are simple SQL queries for data filtering, not for data mining. In this interface the user can also manually edit the SQL query to filter the geometric attributes. Figure 9 shows the same example as Figure 8, but filtering the trajectories that intersect the district Barra da Tijuca.

The SQL query that appears is the last one that has been typed, but the layers of data that appear on the map are related to all relations that are selected on the check box on the left side. For example, if the first query returned a set of trajectories, and the second query returned all districts, and both relations are marked on the check box, only the last query on districts will appear, but both layers are shown on the map.

The visualization interface allows also the user zoom in and out the map, as well as to mark in the check boxes on the left side to choose which relations should be plot on the map.

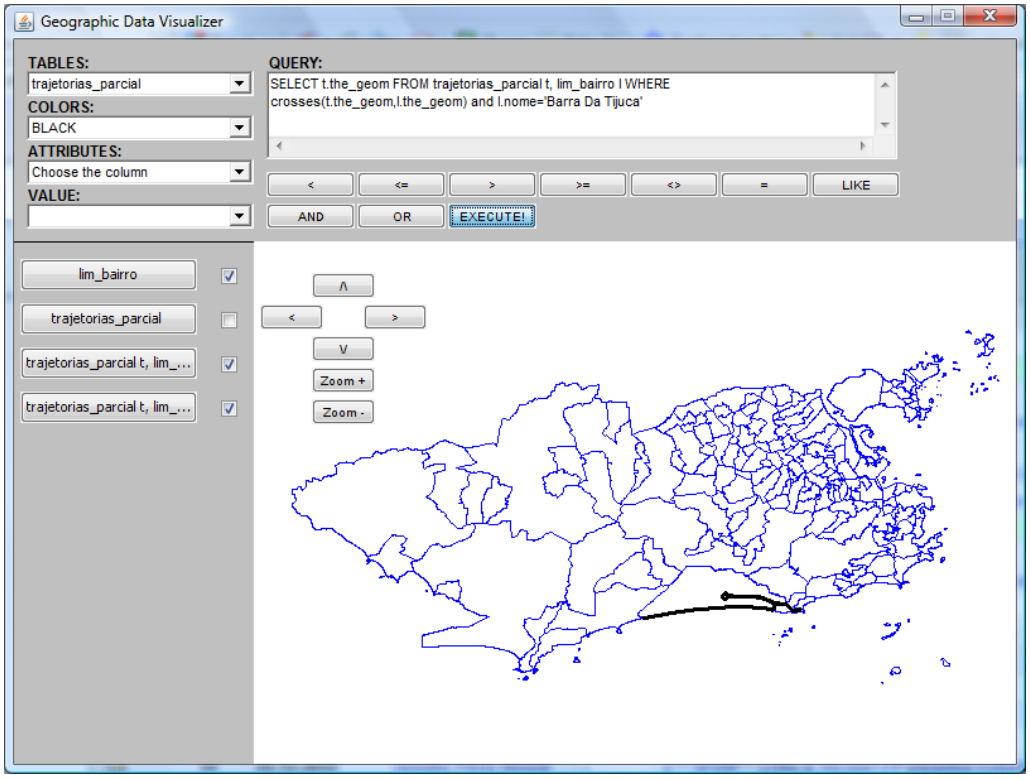

Figure 9. Spatial data visualization interface

5. Case Study: Extracting Multiple Level Semantic Trajectory Patterns with Weka-STPM

In order to show some results that can be obtained with the proposed software, we have tested the prototype with data stored in a Postgresql/PostGIS database. We show a case study with real car trajectory data. In order to understand the whole process we show an example of the output of the most important steps: the semantic enrichment (adding semantics), generalization (granularity transformation), the extracted patterns, and the graphical data visualization.

5.1. Application Scenario

In this section we describe an application scenario of a transportation application, using real trajectories, collected in the city of Rio de Janeiro. In this application the user is interested in extracting patterns from car trajectories to identify patterns of traffic jams, that are characterized as low speed regions. This dataset is composed by 2 thousand trajectories, having a total of around 7 million points. Figure 10 shows the same example of trajectory samples, where E and N are respectively the x and y coordinates, and tid and $time$ are respectively the trajectory identifier and the time attribute. Notice that as the x and y coordinates, the names of the attributes change from one dataset to another. The x and y coordinates are transformed to an attribute called the_geom when a shape file is converted to a database based on OGC standards, and our software will automatically generate the attributes TID and TIME in case their labels are different. The user just has to indicate which attribute is the trajectory identifier and which one represents time.

E	N	DATA	HORA	VELOCIDADE	DIRECAO	TIME	TID
680261,8491000000	7462602,1634000000	28 06 04	13 37 43	0,4000000000	021.5	2004-06-28 13:37:43	1
680262,1887000000	7462601,9748000000	28 06 04	13 37 45	0,4000000000	026.4	2004-06-28 13:37:45	1
680262,3597000000	7462601,9727000000	28 06 04	13 37 46	0,2000000000	024.6	2004-06-28 13:37:46	1
680262,3597000000	7462601,9727000000	28 06 04	13 37 47	0,2000000000	022.8	2004-06-28 13:37:47	1
680262,0156000000	7462601,7923000000	28 06 04	13 37 49	0,4000000000	019.0	2004-06-28 13:37:49	1
680262,1821000000	7462601,4211000000	28 06 04	13 37 51	0,2000000000	018.3	2004-06-28 13:37:51	1
680262,1821000000	7462601,4211000000	28 06 04	13 37 52	0,0000000000	018.5	2004-06-28 13:37:52	1
680262,1799000000	7462601,2365000000	28 06 04	13 37 53	0,0000000000	019.8	2004-06-28 13:37:53	1
680262,3464000000	7462600,8654000000	28 06 04	13 37 55	0,2000000000	023.5	2004-06-28 13:37:55	1
680262,3464000000	7462600,8654000000	28 06 04	13 37 57	0,0000000000	028.2	2004-06-28 13:37:57	1
686480,1560000000	7466272,1076000000	28 06 04	16 07 39	0,2000000000	043.2	2004-06-28 16:07:39	2
686480,2951000000	7466269,5217000000	28 06 04	16 07 41	0,6000000000	045.8	2004-06-28 16:07:41	2
686478,6855000000	7466263,8192000000	28 06 04	16 07 43	0,2000000000	047.4	2004-06-28 16:07:43	2

Figure 10. Raw trajectory sample

```
tid  | stopid | stop_name  | stop_gid |     start_time      |      end_time       |                the_geom
-----+--------+------------+----------+---------------------+---------------------+-----------------------------------------
2236 |      0 | 0_unknown  |        0 | 2004-08-04 18:10:18 | 2004-08-04 18:25:29 | POINT(667015.36217975 7464208.41043153)
2234 |      0 | 1_unknown  |        0 | 2004-07-20 18:35:58 | 2004-07-20 19:06:29 | POINT(679405.188285265 7469918.46315512)
2233 |      0 | 2_unknown  |        0 | 2004-07-20 15:39:33 | 2004-07-20 15:51:50 | POINT(680479.071827787 7466287.37028003)
2232 |      0 | 3_unknown  |        0 | 2005-05-23 08:04:08 | 2005-05-23 10:06:25 | POINT(673805.183177759 7455273.90615614)
1236 |      0 | 1_unknown  |        0 | 2005-02-21 17:20:42 | 2005-02-21 18:04:10 | POINT(681063.492700548 7469816.57366467)
1235 |      0 | 1_unknown  |        0 | 2005-02-18 16:39:03 | 2005-02-18 17:08:24 | POINT(681203.357858659 7469257.89224045)
1234 |      0 | 4_unknown  |        0 | 2005-02-17 15:33:33 | 2005-02-17 15:46:48 | POINT(686860.9302822 7463298.01652945)
1233 |      0 | 1_unknown  |        0 | 2005-02-16 07:47:22 | 2005-02-16 08:11:14 | POINT(680939.399430723 7469585.19999004)
1232 |      0 | 1_unknown  |        0 | 2005-02-15 16:45:57 | 2005-02-15 17:24:24 | POINT(681037.05860043 7469842.71077073)
1231 |      0 | 1_unknown  |        0 | 2005-02-14 07:37:42 | 2005-02-14 08:24:26 | POINT(679634.452558103 7471157.38290711)
```

Figure 11. Stops dataset

The user will run Weka and enter the Explorer interface, connect to the database and click on the button Trajectories to open the the STPM module. After loading the database relations on the button Load, the user can for instance, clean the trajectory dataset by removing a given number of points. This may be done by any of the cleaning methods.

So the next step for this application is to generate stops, with the method CB-SMOT, that is specific to find low speed regions.

5.2. Preprocessing Trajectories with CB-SMoT

For this experiment we considered as low speed regions those places where the speed was lower that 70% of the average speed of the trajectory (MaxAvgSpeed). For instance, if the average speed of the trajectory was 100 KM/h, low speed regions will correspond to subtrajectories with speed less than 70KM/H, for at least minTime, that we considered as 10 minutes (600 seconds). In this experiment we considered slums, hospitals and schools as the relevant spatial feature types that may be related to trajectory low speed regions (candidate stops).

After running the *adding semantics* module, a set of stops and moves has been generated. A partial set of stops, that correspond to a subset of trajectories, is shown in Figure 11. Notice that instead of sample points we have now more semantic information, knowing the low velocity parts of trajectories. The slow parts can correspond to regions that intersect relevant spatial features, like hospital surroundings, or if these regions do not intersect any spatial feature they are labeled as unknown stops. In Figure 11 we can notice that several trajectories have a common subtrajectory, at the same place and time interval, labeled as *1_unknown*. In this example we retrieved only the centroid of the stop, which is therefore a point. The real stored geometry of the stop is a line (set of points generating a subtrajectory).

Once the stops are generated, which is the step that adds semantics to trajectories and is the most time consuming step of the knowledge discovery process, the data are

ready for multiple-level mining.

5.3. Mining Patterns at Multiple Granularities

After adding semantics to trajectories with the method CB-SMOT, we transformed the data into different space and time granularities for then apply the algorithm in Weka for generating sequential patterns.

Figure 12 shows the result of a sequential pattern mining process on semantic trajectories (stops) considering two different granularities for the time dimension. In the first experiment (Figure 12-1) the time was at the user defined interval, characterizing rush hours ([07:00-07:59, 08:00-08:59, 17:00-17:59, and 18:00-18:59]. In this experiment we can observe that among patterns with one item, a traffic jam occurs around slum 321, in the morning (between 7:00 and 8:00) and in the afternoon (between 18:00 and 19:00). Another sequential pattern, but with two items, occurs at stops 0_unknown and 4_unknown, with support equal or higher that 5 %. This pattern occurs between 5 and 6 PM (between 17:00 and 18:00). At the same places (0_unknown and 4_unknown) the pattern repeats between the hours of 18:00 and 19:00, althought now with less support, for 8 trajectories.

With these patterns we know that low traffic occurs at the places 0_unknown and 4_unknown at the end of the day, but suppose that now we want to discover if these pattern repeats every day. For that, we have to change the time granularity to days of the week. After changing the time granularity and generating the arff file again, and running the sequential pattern mining method, we obtain the results shown in Figure 12-2, also generated considering 5% as minimum support. Notice that in this experiment the stops 0_unknown and 4_unknown are frequent again, but now for 8 trajectories and the time dimension now is Wednesday. So we can conclude that the pattern of low traffic occurs at stops 0_unknown and 4_unknown, between 17:00 and 19:00 on wednesday. This pattern is not frequent to any other days of the week, in this sequence, from 0_unknown to 4_unknown.

Now the user may be wondering where the stops 0_unknown and 4_unknown are located in space. He can then use the new visualization interface to query these patterns, as described in the following section.

5.4. Spatial Pattern Visualization

Figure 13 shows the graphical result of the stops 0_unknown and 4_unknown computed with the method CB-SMoT, and that represent two regions with low traffic movement. These places are located in two districts in the northern part of Rio de Janeiro, respectively the districts Vila Militar and a set of districts Penha, Penha Circular, Bras de Pina and Cordovil. In this version of the prototype the output of the query must be a geometric object.

Finally, Figure 14 shows the result of all stops computed with the method CB-SMOT, the set of trajectories on which stops were computed and the districts of Rio de Janeiro, so that the user may visualize all low speed regions over the set of trajectories.

These patterns are visualized with the new pattern visualization module, only available in this version of our prototype.

```
(1) item=nameStart,    spaceG=instance,
    timeG=[07:00-07:59,08:00-08:59,09:00-09:59,17:00-17:59,18:00-18:59, 19:00-20:00]
Scheme:      weka.associations.TrajectorySequentialPattern -S 0.05
Large Sequences of Length 2
        (0_unknown_17:0-17:59, 4_unknown_17:0-17:59)  Support: 11 trajs
        (0_unknown_17:0-17:59, 3_unknown_18:0-18:59)  Support: 9 trajs
        (0_unknown_18:0-18:59, 4_unknown_18:0-18:59)  Support: 8 trajs
        (2_unknown_7:0-7:59, 0_unknown_8:0-8:59)      Support: 8 trajs
        (1_unknown_7:0-7:59, 0_unknown_8:0-8:59)      Support: 10 trajs
Large Sequences of Length 1
        (1_unknown_18:0-18:59)      Support: 10 trajs
        (321_slum_18:0-18:59)       Support: 8 trajs
        (3_unknown_8:0-8:59)        Support: 8 trajs
        (2_unknown_8:0-8:59)        Support: 14 trajs
        (9_unknown_8:0-8:59)        Support: 13 trajs
        (1_unknown_8:0-8:59)        Support: 11 trajs
        (0_unknown_16:0-16:59)      Support: 9 trajs
        (13_unknown_7:0-7:59)       Support: 8 trajs
        (321_slum_7:0-7:59)  Support: 8 trajs
        (3_unknown_17:0-17:59)      Support: 8 trajs
        (13_unknown_6:0-6:59)       Support: 8 trajs

(2) item=nameStart,    spaceG=instance,    timeG=[weekday]
Scheme:      weka.associations.TrajectorySequentialPattern -S 0.05
Large Sequences of Length 2
      (0_unknown_wednesday,4_unknown_wednesday)    Support: 8 trajs
      (1_unknown_wednesday,0_unknown_wednesday)    Support: 8 trajs
      (1_unknown_friday,0_unknown_friday)          Support: 9 trajs
      (2_unknown_friday,0_unknown_friday)          Support: 10 trajs
Large Sequences of Length 1
      (4_unknown_thursday) Support: 11 trajs
      (3_unknown_tuesday)  Support: 9 trajs
      (1_unknown_monday)   Support: 10 trajs
      (4_unknown_tuesday)  Support: 9 trajs
      (2_unknown_tuesday)  Support: 9 trajs
      (321_slum_friday)    Support: 10 trajs
      (3_unknown_monday)   Support: 11 trajs
      (3_unknown_thursday) Support: 11 trajs
```

Figure 12. Sequential pattern output at different time granularities

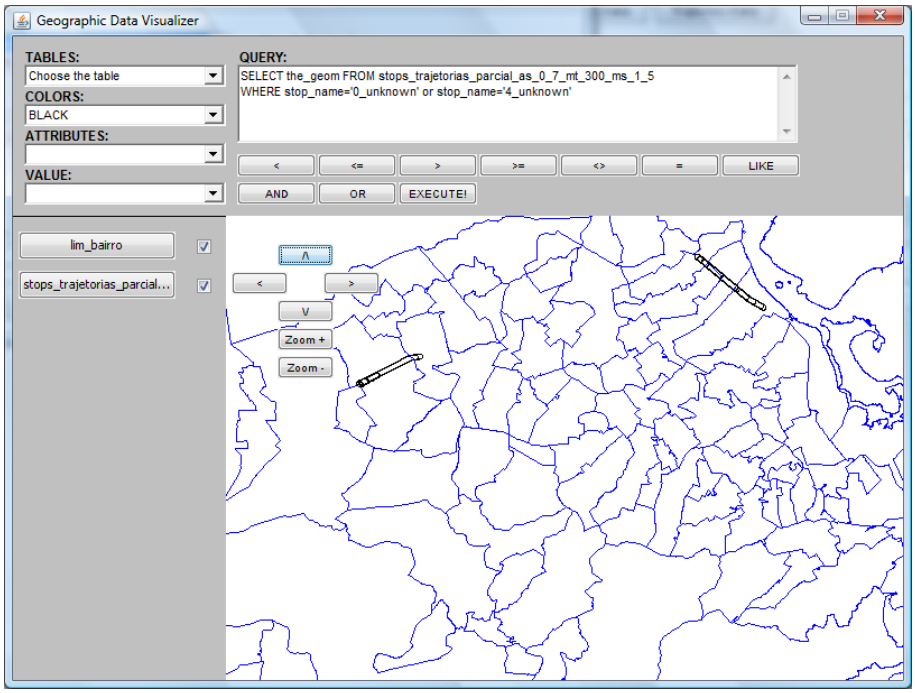

Figure 13. Frequent sequential stops

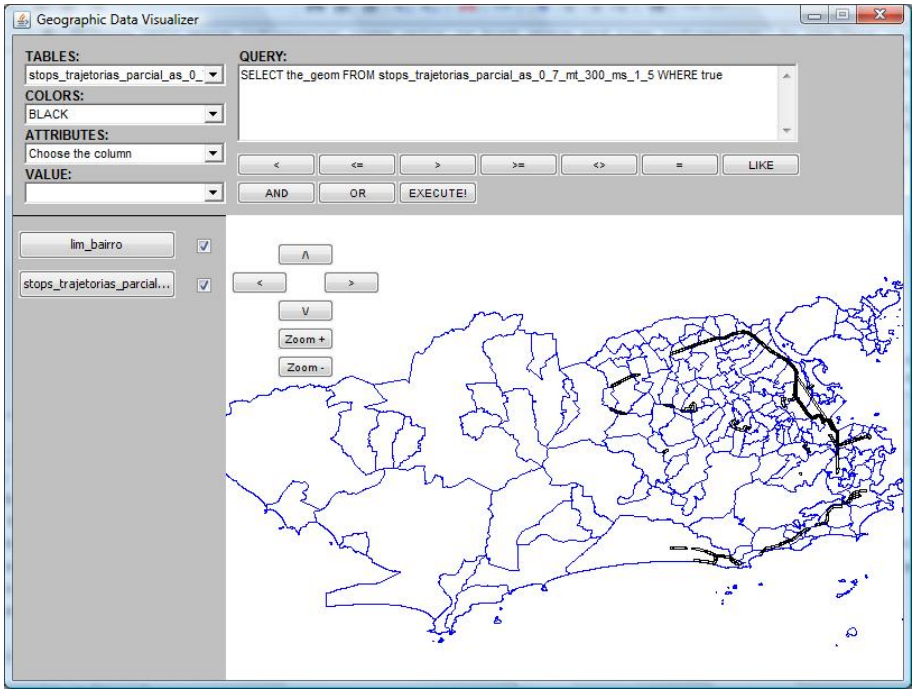

Figure 14. Low speed regions in the city of Rio de Janeiro

6. Conclusions and Future Work

The data generated by mobile devices are raw data that, by their nature, are very difficult to understand and to use for decision making processes. To extract patterns from these data is not a trivial task from the user point view. One reason is the fact that these data have no semantics/meaning associated, being very difficult to query, to interpret, or to visually identify patterns. Another reason is the fact that although several methods have been developed for trajectory pattern mining, these methods are not implemented in a toolkit, and no tool is available to help the user to automatically or semi-automatically perform the complete knowledge discovery process.

The main contribution of this paper can be summarized in a software architecture, that can be easily extended, and a software prototype that implements several pieces that are necessary to semantic trajectory data mining. It is the first trajectory data mining software prototype, implemented in an open source tool, and that covers the steps of data preprocessing, data mining, and data and pattern visualization.

The proposed approach is general enough to cover different application domains. Two methods are available to enrich trajectories with domain-specific geographic information. In this paper we extend the method CB-SMOT to semantically enrich trajectories using maximal speed and speed limit to identify the low speed regions in trajectories, while in [Palma et al. 2008] low speed regions were computed using a distance function as in the method DBSCAN. We improved this method to increase the usability for the user.

Trajectory enriched data can be transformed into both space an time different granularities, what is a fundamental task in the data mining process, and that is almost impossible to be done manually.

With Weka-STPM the user is able to automatically add semantic geographic information to trajectories, in a preprocessing step, and then exploit several algorithms available in Weka for data mining. Since the semantic enriched trajectory data are stored in a database, this allows the user to exploit the data with spatial and non-spatial queries as well.

A very interesting aspect that is provided by the tool is the visualization interface, similar to ArcGIS and QuantumGIS, although not as sophisticated as, but it allows the user to graphically visualize the data, as well as to perform visual SQL queries from this interface. It allows the overlay of several layers of spatial and spatio-temporal information.

As far as we know, this is the first data mining toolkit that provides such visualization interface.

As future works we are implementing new trajectory data mining methods to extract behavior patterns, considering both the geometric properties of trajectories and domain knowledge provided by the user in a knowledge base. We are also improving the spatial data visualization interface, to make it more flexible. Last but not least, we are evaluating the usability of the proposed tool.

Acknowledgments

The authors would like to thank both CNPq and FAPESC for the financial support of this research and all students that contributed for this tool.

References

Alvares, L. O., Bogorny, V., Kuijpers, B., de Macedo, J. A. F., Moelans, B., and Vaisman, A. (2007). A model for enriching trajectories with semantic geographical information. In *ACM-GIS*, pages 162–169, New York, NY, USA. ACM Press.

Alvares, L. O., Palma, A. T., Oliveira, G., and Bogorny, V. (2010). Weka-stpm: From trajectory samples to semantic trajectories. In *Workshop on Open Source Code*, pages 1–6. SBC.

Baglioni, M., de Macêdo, J. A. F., Renso, C., Trasarti, R., and Wachowicz, M. (2009). Towards semantic interpretation of movement behavior. In Sester, M., Bernard, L., and Paelke, V., editors, *AGILE Conf.*, Lecture Notes in Geoinformation and Cartography, pages 271–288. Springer.

Bogorny, V. (2010). Weka-stpm download. www.inf.ufsc.br/ vania/software.html.

Bogorny, V., Kuijpers, B., and Alvares, L. O. (2009). St-dmql: A semantic trajectory data mining query language. *International Journal of Geographical Information Science*, 23:1245–1276.

Bogorny, V., Palma, A. T., Engel, P., and Alvares, L. O. (2006). Weka-gdpm: Integrating classical data mining toolkit to geographic information systems. In *WAAMD Workshop*, pages 9–16. SBC.

Cao, H., Mamoulis, N., and Cheung, D. W. (2006). Discovery of collocation episodes in spatiotemporal data. In *ICDM*, pages 823–827. IEEE Computer Society.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In Simoudis, E., Han, J., and Fayyad, U. M., editors, *Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press.

Frank, E., Hall, M. A., Holmes, G., Kirkby, R., Pfahringer, B., Witten, I. H., and Trigg, L. (2005). Weka - a machine learning workbench for data mining. In Maimon, O. and Rokach, L., editors, *The Data Mining and Knowledge Discovery Handbook*, pages 1305–1314. Springer.

Gudmundsson, J. and van Kreveld, M. J. (2006). Computing longest duration flocks in trajectory data. In de By, R. A. and Nittel, S., editors, *GIS*, pages 35–42. ACM Press.

Janowicz, K., Raubal, M., Schwering, A., and Kuhn, W. (2008). Semantic similarity measurement and geospatial applications. *T. GIS*, 12(6):651–659.

Laube, P., Imfeld, S., and Weibel, R. (2005). Discovering relative motion patterns in groups of moving point objects. *International Journal of Geographical Information Science*, 19(6):639–668.

Lee, J., Han, J., and Whang, K. Y. (2007). Trajectory clustering: A partition-and-group framework. In *SCM SIGMOD International Conference on Management Data (SIGMOD'07)*, Beijing, China.

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

Lee, J.-G., Han, J., Li, X., and Gonzalez, H. (2008b). *raClass*: trajectory classification using hierarchical region-based and trajectory-based clustering. *PVLDB*, 1(1):1081–1094.

Li, Y., Han, J., and Yang, J. (2004). Clustering moving objects. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 617–622, New York, NY, USA. ACM Press.

Li, Z., Lee, J.-G., Li, X., and Han, J. (2010). Incremental clustering for trajectories. In Kitagawa, H., Ishikawa, Y., Li, Q., and Watanabe, C., editors, *DASFAA (2)*, volume 5982 of *Lecture Notes in Computer Science*, pages 32–46. Springer.

Nanni, M. and Pedreschi, D. (2006). Time-focused clustering of trajectories of moving objects. *J. Intell. Inf. Syst.*, 27(3):267–289.

OGC (2008). Opengis standards and specifications: Topic 5: Features. Available at: http://portal.opengeospatial.org/modules/admin/license_agreement.php?suppressHeaders=0&access_lice Accessed in November 2010.

Palma, A. T., Bogorny, V., and Alvares, L. O. (2008). A clustering-based approach for discovering interesting places in trajectories. In *ACMSAC*, pages 863–868, New York, NY, USA. ACM Press.

Pelekis, N., Theodoridis, Y., Vosinakis, S., and Panayiotopoulos, T. (2006). Hermes - a framework for location-based data management. In Ioannidis, Y. E., Scholl, M. H., Schmidt, J. W., Matthes, F., Hatzopoulos, M., Böhm, K., Kemper, A., Grust, T., and Böhm, C., editors, *EDBT*, volume 3896 of *Lecture Notes in Computer Science*, pages 1130–1134. Springer.

Spaccapietra, S., Parent, C., Damiani, M. L., de Macedo, J. A., Porto, F., and Vangenot, C. (2008). A conceptual view on trajectories. *Data and Knowledge Engineering*, 65(1):126–146.

Trasarti, R., Rinzivillo, S., Pinelli, F., Nanni, M., Monreale, A., Renso, C., Pedreschi, D., and Giannotti, F. (2010). Exploring real mobility data with m-atlas. In Balcázar, J. L., Bonchi, F., Gionis, A., and Sebag, M., editors, *ECML/PKDD (3)*, volume 6323 of *Lecture Notes in Computer Science*, pages 624–627. Springer.

Tsoukatos, I. and Gunopulos, D. (2001). Efficient mining of spatiotemporal patterns. In Jensen, C. S., Schneider, M., Seeger, B., and Tsotras, V. J., editors, *SSTD*, volume 2121 of *Lecture Notes in Computer Science*, pages 425–442. Springer.

van Hage, W. R., Wielemaker, J., and Schreiber, G. (2010). The space package: Tight integration between space and semantics. *T. GIS*, 14(2):131–146.

Verhein, F. and Chawla, S. (2006). Mining spatio-temporal association rules, sources, sinks, stationary regions and thoroughfares in object mobility databases. In Lee, M.-

L., Tan, K.-L., and Wuwongse, V., editors, *DASFAA*, volume 3882 of *Lecture Notes in Computer Science*, pages 187–201. Springer.