Analysis of Spatiotemporal Inconsistencies of Trajectories with Planned and Reported Tasks

Felipe Pinto da Silva, Renato Fileto

PPGCC/INE, Federal University of Santa Catarina (UFSC)
PO BOX 476, 88040-900, Florianópolis-SC, BRAZIL

fpsilva98@gmail.com, r.fileto@ufsc.br

Abstract. *Trajectories automatically collected by sensors enable tracking moving objects that have to perform tasks in the geographic space. However, most existing methods that use such trajectories for analyzing moving objects behaviors do not use other relevant data sources. This paper proposes a method to detect and classify spatiotemporal inconsistencies of trajectories with both planned tasks and reported tasks. The classified spatiotemporal inconsistencies returned by the proposed method help to investigate possible misbehaviors of moving objects. This method has been implemented and evaluated with real data from a water supply company. It has helped to detect and investigate a variety of inconsistencies in these data.*

1. Introduction

Nowadays, a variety of mobile devices such as cell phones equipped with geographic positioning technologies (e.g., GPS, GSM) allow the automatic gathering of raw trajectories, i.e., temporally ordered sequences of spatiotemporal positions occupied by moving objects. Trajectories have more precise and detailed spatiotemporal information than traditional travel diaries, for example. Thus, they are a suitable source of information to analyze moving objects' behaviors.

Trajectories processing methods to investigate moving objects' behaviors while these objects (are supposed to) perform tasks in the geographical space have recently attracted the attention of some research groups. These methods have many applications, ranging from the management of itinerant teams doing public or commercial services (e.g., public works, maintenance in place, domiciliary inspection, in-home caregiving) to delivery services (e.g., conventional mail, pizza delivery). However, the analysis of data from other sources, such as tasks planning and travel diaries, along with trajectories allow the extraction of further relevant information about the tasks (e.g., type, goal), and other activities that may influence tasks execution performance (e.g., idle time).

This paper proposes a computational method to automatically detect and classify spatiotemporal inconsistencies of moving objects' trajectories with their planned tasks and tasks that they report to have done in specific spatiotemporal points. The inputs of the proposed method are: (i) raw trajectories of moving objects; (ii) the execution place and the expected duration of each planned task; and (iii) reports made by the moving objects themselves, with (possibly inaccurate) indications of the ending time, duration, and geographic coordinates of each executed task . The proposed method confronts the data from these three sources, and classifies each found inconsistency. The returned

spatiotemporal inconsistencies among the data of distinct nature and sources support the investigation of operational problems and possible misbehaviors of moving objects. The proposed method has been implemented in a prototype that uses state-of-the-art methods for spatiotemporal data processing. Experiments with real data about the activities of maintenance teams of a water supply company have shown that the proposed method is able to detect and help to investigate a wide variety of inconsistencies.

The rest of this paper is structured as follows. Section 2 provides the foundations necessary to understand the proposal. Section 3 describes the proposed method in detail. Section 4 reports experimental results. Section 5 reports some related work, and Section 6 concludes the paper, by enumerating contributions and themes for future work.

2. Foundations

The inputs of the proposed method are trajectories, planned tasks, and reported tasks. They are formally described in the following.

2.1. Trajectories

A raw trajectory is a sequence of spatiotemporal positions of a moving object obtained from some sensor (GPS, GSM, RFID, cameras). Definition 2.1 formalizes this concept.

Definition 2.1. *A **raw trajectory*** $\tau = (idMO, \underline{idTraj}, (x_1, y_1, t_1), \ldots, (x_n, y_n, t_n))$ *is a temporally ordered sequence of spatiotemporal positions of a moving object, where:*

- $idMO$ *is the identifier of the moving object;*
- $\underline{idTraj}$ *is the unique identifier of the trajectory* τ*;*
- *each* (x_i, y_i, t_i) *is a spatiotemporal position (point), representing that the object is at the spatial coordinates* (x_i, y_i) *in the instant* $t_i (1 \leq i \leq n)$*; and*
- $n > 0$ *is the number of spatiotemporal positions of* τ*.*

Raw trajectories must be preprocessed to eliminate inaccuracies caused by limitations of the sensing devices and problems of the trajectory gathering process [Yan *et al*. 2013]. Then, their filtered and possibly adjusted spatiotemporal positions can be structured in episodes [Mountain and Raper 2001], described by Definition 2.2.

Definition 2.2. Given a trajectory $\tau = (idMO, idTraj, (x_1, y_1, t_1), \ldots, (x_n, y_n, t_n))$, *an **episode*** $E = (\underline{idTraj, idE}, (x_{start}, y_{start}, t_{start}) \ldots (x_{end}, y_{end}, t_{end}))$ *is a sub-sequence of* τ $(1 \leq start \leq end \leq n)$ *that satisfies a given predicate, and is maximal in terms of its number of spatiotemporal points, where:*

- $\underline{idTraj}$ *is the trajectory identifier;*
- $\underline{idE}$ *is the episode identifier;*
- t_{start} *is the instant of the first spatiotemporal point of the episode; and*
- t_{end} *is the instant of the last spatiotemporal point of the episode.*

Episodes can be moves or stops, for example. The positions constituting an episode such as a stop can be determined by distinct predicates (e.g., being inside a place, having speed below a given threshold). [Spaccapietra *et al.* 2008] introduced the trajectory model based on intertwined stops and moves, and defines a semantic trajectory as a sequence of relevant locations visited by the moving object. Structured trajectories are time ordered sequences of episodes [Yan *et al*. 2013] (Definition 2.3).

Definition 2.3. *A **structured trajectory** is a timely ordered sequence of temporally disjoint episodes* $T_s = \{E_1, E_2, \ldots E_m\}$ $(m > 0)$.

Comprehensive reviews about concepts and techniques related with trajectories data processing (e.g., detecting relevant episodes and behaviors) can be found in [Parent *et al*. 2013] and [Pelekis and Theodoridis 2014]. A method for building structured trajectories by extracting stops on a given set o places is proposed in [Alvares *et al*. 2007]. This method is called IB-SMoT (Intersection-Based Stops and Moves of Trajectories). CB-SMoT (Cluster-Based SMoT) is a method for structuring trajectories based on speed [Palma *et al*. 2008]. It is able to identify stops by clustering adjacent positions in which the moving object is stationary or moves slowly.

In this paper we consider that a stop is necessary for a moving object to perform a task. This assumption is also used in [Huang *et al*. 2010], [Furletti *et al*. 2013] and [Clark & Doherty 2008]. Then, as we are interested in analyzing inconsistencies between stops, tasks planning and reported tasks, we use CB-SMoT to detect stops in any location, even locations where there is no reported or planned task.

2.2. Tasks

According to Huang, a task has location, initial instant, duration, and goal [Huang *et al*. 2010]. However, the proposed method does not impose rigid schedule. i.e., doesn't matter when a task should start, but the minimum and maximum expected duration.

Definition 2.4. *A **planned task** is a tuple* φ = (idMO,idTask,(x,y), minExpDuration,maxExpDuration,requestDate,maxCompletionDate), where:

- idMO is the identifier of moving object;
- idTask is the identifier of planned task;
- (x,y) are the spatial coordinates where the planned task should be performed;
- minExpDuration ≤ maxExpDuration are respectively the minimum and the maximum expected duration of the planned task;
- requestDate ≤ maxCompletionDate are respectively the request date and the maximum allowed completion date of the planned task execution.

Clark e Doherty [Clark & Doherty 2008] added feedback from users to analyze rescheduling tasks. These feedbacks only confirms the start execution and finish execution. Our method introduces the concept of reports made by the moving objects.

Definition 2.5. *A **reported task** is a tuple* ρ = (idMO,idTask,startTime,endTime, (x,y)), where

- idMO is the identifier of moving object;
- idTask is the identifier of planned and reported task;
- startTime is the starting time of the task execution;
- endTime is the end time of the task execution;
- (x,y) are the spatial coordinates from which the report was sent.

3. Analysis of Spatiotemporal Inconsistencies

This section describes the proposed method to analyze spatiotemporal inconsistencies of trajectories with planned and reported tasks. Section 3.1 presents an overview of the

method, while Sections 3.2 and 3.3 detail the method steps of inconsistencies detection and inconsistencies classification, respectively.

3.1. Overview

Figure 1 summarizes the proposed method. Its inputs are T, P, and R. T is a set of raw trajectories (Definition 2.1). P is a set of planned tasks (Definition 2.4). R is a set of reported tasks (Definition 2.5). Step 1 (Preprocessing and Extraction of Stop Episodes) starts by preprocessing each trajectory in T to generate a set of structured trajectories T_S (Definition 2.3). Each structured trajectory in T_S is a sequence of stops and moves (Definition 2.2). These episodes are extracted by using a cluster-based method such as CB-SMoT [Palma *et al*. 2008]. For simplicity and efficiency of the next step, the location of each stop is approximated by a single point, such as its centroid. The stops of all the structured trajectories of T_S are inserted in the relation S that is used as one of the inputs for step 2 (Detection of Inconsistencies).

Step 2 executes spatiotemporal SQL queries on sets of trajectory stops, planned tasks, and report tasks. The third and last step (Classification of Inconsistencies) helps to explain the inconsistencies found. Steps 2 and 3 are described in the following. The outputs of the proposed method are classified spatiotemporal inconsistencies.

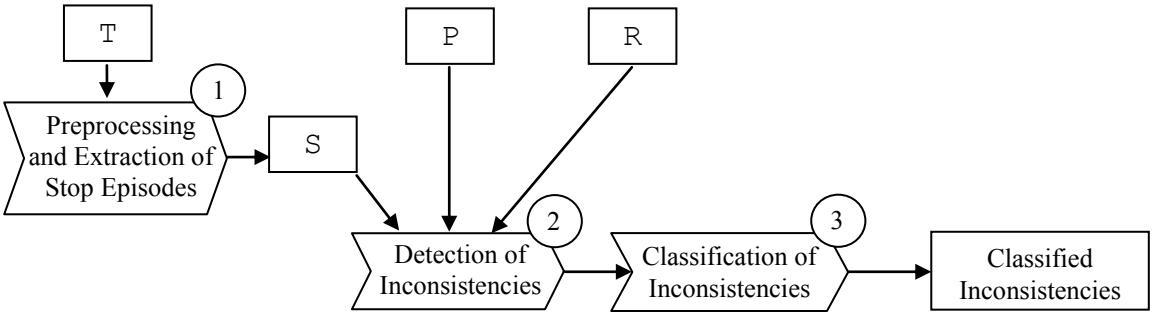

Figure 1. Data processing steps of the proposed method

3.2. Detection of Inconsistencies

The detection of inconsistencies is done by exploiting the relational database view whose query expression is presented in Figure 2. This query applies successive outer joins on the tables: planned tasks (P), reported tasks (R), and the stop episodes of the structured trajectories (S). All these datasets refer to the same set of moving objects during a certain time period (e.g., a month). R and S take part in more than one join in the query, with distinct join conditions. It is important to distinguish the attribute values (and *NULL* values) resulting from each outer join to have all the needed information to detect different kinds of inconsistencies in the final result (Res). Thus, we use R' and S', which are just the projections of the attributes of R and S, respectively, necessary for the respective join conditions. In other words:

$$R' = \pi_{idMO,startTime,endTime,x,y}(R)$$

$$S' = \pi_{idMO,startTime,endTime,x_c,y_c}(S)$$

The attributes of P, R, and S are as stated in Definitions 2.4, 2.5, and 2.2, respectively. The coordinates ($S.x_c$, $S.y_c$) can refer to the centroid of each stop episode,

for example. These coordinates are used instead of the whole subsequence of spatiotemporal points of each stop to represent it, for simplicity and efficiency.

The first operation of the query presented in Figure 2 (a) is a full outer join between `P` and `R`. The join condition is a conjunction of equalities of the attributes `idMO` and `idTask` of both tables(natural join condition), with the time compatibility of the planned task with the reported task (i.e., `P.requestDate` and `P.maxCompletionDate` less or equal `R.startTime` and `R.endTime`, respectively) This join enables the identification of planned tasks that do not match any reported task (i.e., planned tasks that were not realized or whose execution was not reported), and vice versa (i.e., reported tasks that do not correspond to any planned task).

The other outer joins in the query presented in Figure 2 (b, c, and d) have as join conditions the conjunction proximity of the coordinates of the respective spatiotemporal data (e.g., `distance((P.x,P.y),(R'.x,R'.y)) ≤ SThreshold_PR`), with moving object and time compatibility conditions. The spatial thresholds `SThreshold_PR`, `SThreshold_PT` and `SThreshold_RT` can be adjusted according to the dataset properties and application goals. Thus, one spatial point is considered close to another one if the distance between them is smaller than the respective spatial threshold.

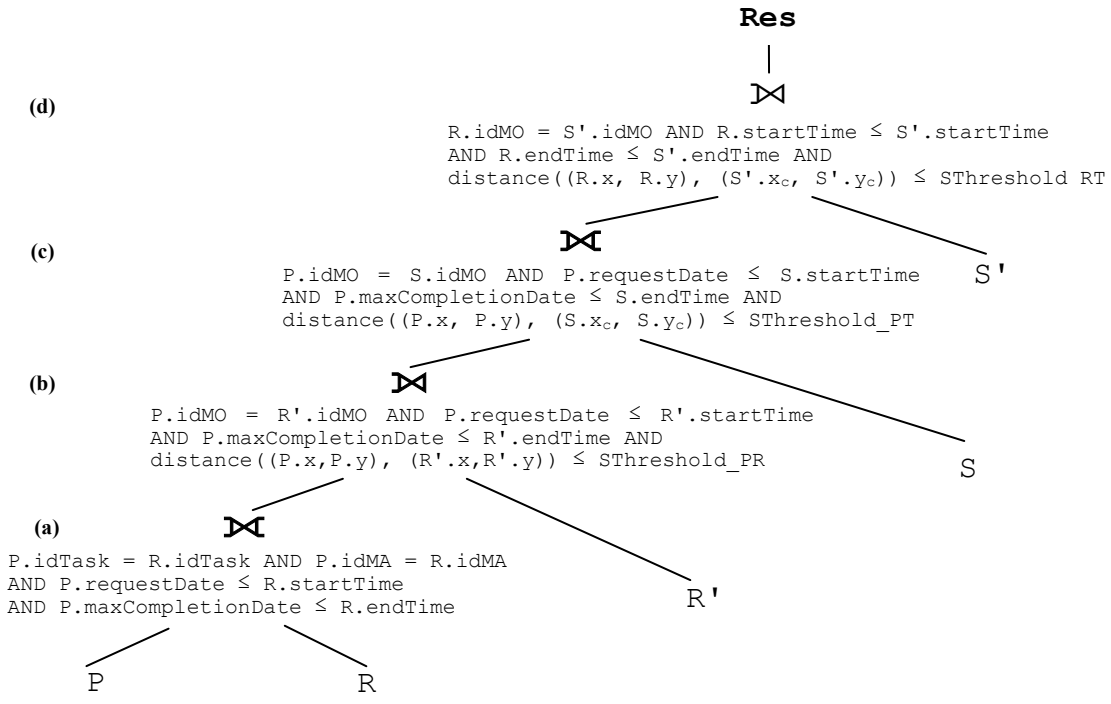

Figure 2. Definition of the view used for inconsistencies detection

The proposed method detects inconsistencies of each kind by posing a distinct query on the view `Res`. These queries filter tuples of `Res` presenting some partial matchings among planned tasks, reported tasks, and stop episodes referring to the same moving object. The planned task and the reported task must refer to the same `idTask`, as stated in Figure 2, to avoid combinatorial explosion. The considered partial matchings are detected by selection based on conjunctive conditions that exploit the values or values absence (*NULL* value) of some attributes of `P`, `R`, `R'`, `S`, and `S'`.

For example, if a reported task position is too far away from its corresponding planned task position, then `P.idTask` and `R.idTask` are not null and have the same value in `Res`, but `R'.idReport` is null, because `R'` has been joined with `P` by proximity to produce `Res`. Temporal inconsistencies, on the other hand, are detected by checking the temporal attributes in `Res`. For example, if a given moving object reports a task lasting longer than its maximum expected duration, then:

$$(R.endTime - R.startTime) > P.maxExpDuration$$

Figure 3 exemplifies the detection of planned tasks matching a reported task and a trajectory stop, but whose matching reported task duration is shorter than the minimum expected time to perform that task.

$$\Pi_{\text{distinct P.idTask}}\ \sigma_{\substack{\text{P.idTask is not null AND}\\ \text{R.idReport is not null AND}\\ \text{S.idStop is not null AND}\\ \text{R.idReport = R'.idReport AND}\\ \text{(R.endTime - R.startTime) > P.maxExpDuration}}}\ (\mathbf{Res})$$

Figure 3. Retrieving inconsistencies between planned duration and reported duration

Analogous queries can extract other kinds of inconsistencies by changing the restrictions on the attributes of `Res` used to filter its tuples. Figure 4 presents the extraction of planned and reported tasks whose report location is far from planned location. Moreover, the reported duration is greater than the maximum expected duration, and the moving object has not stopped close to the planned location.

$$\Pi_{\text{distinct P.idTask}}\ \sigma_{\substack{\text{P.idTask is not null AND}\\ \text{R.idReport is not null AND}\\ \text{S.idStop is not null AND}\\ \text{R'.idReport is null AND}\\ \text{P.maxExpDuration < (R.endTime - R.startTime)}}}\ (\mathbf{Res})$$

Figure 4. Retrieval of spatiotemporal inconsistencies on tasks location and duration

Figure 5 illustrates an inconsistency retrieved by the query presented in Figure 4. Let $\varphi \in P$ be a planned task, $\varphi.minExpDuration = 20$, and $\varphi.maxExpDuration = 30$, both in minutes. This task should be performed inside the region delimited by the blue circle labeled A. The red circles represent the centroids of two stops (s_1 , $s_2 \in S$) of the moving object assigned to execute φ. Consider $(s_1.endTime - s_1.startTime) = 20$ and $(s_2.endTime - s_2.startTime) = 10$. If the reported task $\rho \in R$ was done during the stop s_2, is less than `SThreshold_PT` meters away from s_2, and has the same `idTask` as that of φ, then there is a spatial inconsistency between φ and ρ, and a duration inconsistency between ρ and s_2.

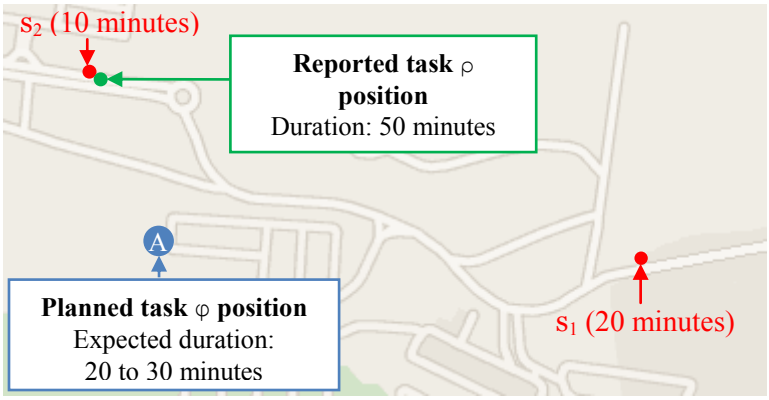

Figure 5. Detection of inconsistency in figure 4

3.3. Classification of Inconsistencies

The inconsistencies detected by the previous task are classified according with a conjunction of conditions that can hold or not on tuples of the view `Res`. These conditions can be disposed in a truth table for systematic assessment. They are expressed by possible *NULL* values on the attributes of planed tasks, reported tasks or trajectory episodes (`φ, ρ, ρ', s, s'`), which indicate lack of matching components for natural or similarity joins of the relational algebra expression that defines `Res` (Figure 2). The following temporal conditions are also considered in the truth table:

c_1: `ρ.duration ≤ φ.maxExpDuration`

c_2: `ρ.duration ≥ φ.minExpDuration`

c_3: `s.duration ≤ φ.maxExpDuration`

c_4: `s.duration ≥ φ.minExpDuration`

The duration of a reported task `ρ` or stop `s` can be determined as follows:

```
ρ.duration = (ρ.endTime – ρ.startTime)
s.duration = (s.endTime – s.startTime)
```

Thus, the truth table used to classify inconsistencies has 2^7 combinations of Boolean values, indicating if the attributes of `P`, `R`, `R'`, `S`, and `S'` exist (are not *NULL*), and if the temporal conditions c_1, c_2, c_3, and c_4 are satisfied. The number of lines of the fact table can be reduced by eliminating combinations of Boolean values that do not make sense (e.g., if attributes of `P` or `R` are *NULL* then the conditions c_1 and c_2 cannot be evaluated), and by considering only combinations of conditions that express relevant classes of inconsistencies (e.g., for a particular company).

Table 1 presents 10 classes of inconsistencies considered relevant by the water supply company of our case study to investigate possible misbehavior of its maintenance teams. The symbols ∃ and ∄ indicate if the attributes of `P`, `R`, `R'`, `S`, and `S'` are present or *NULL*, respectively, in the view `Res`. The columns c_1, c_2, c_3 and c_4 are checked (✓) if the respective temporal condition is satisfied, unchecked (✗) if they are satisfied, and marked as not possible to assess (-) otherwise. The class 1 refers to no inconsistency, i.e., the reported task and the stop positions are sufficiently close to that of the planned task, and their durations are within the expected time limits of the planned task. The

remaining classes presented in Table 1 refer to inconsistencies whose explanations appear in the last column.

Table 1. Some relevant classes of inconsistencies and their explanations

C	P.*	R.*	R'.*	S.*	S'.*	c_1	c_2	c_3	c_4	Explanation
1	∃	∃	∃	∃	∃	✓	✓	✓	✓	No inconsistency
2	∃	∃	∃	∃	∃	✓	×	✓	✓	Short reported task
3	∃	∃	∃	∃	∃	×	✓	✓	✓	Prolonged reported task
4	∃	∃	∄	∃	∃	×	✓	✓	×	Prolonged reported task, short stop, and wrong location of task report
5	∃	∃	∄	∄	∄	×	✓	-	-	Prolonged reported task, and planned task location not visited
6	∃	∄	∄	∃	∄	-	-	✓	×	Unreported task and planned location shortly visited
7	∃	∃	∄	∃	∄	×	✓	✓	✓	Prolonged reported task at planned location that is not visited.
8	∃	∄	∄	∄	∄	-	-	-	-	Task not performed
9	∄	∃	∄	∄	∄	-	-	-	-	Report of unknown task
10	∄	∄	∄	∃	∄	-	-	-	-	Idle stop

The inconsistency class 5 is detected by the query in Figure 4, and illustrated by task A of Figure 5. This inconsistency has no stops close to the planned task location. However, the reported task position is far away from its corresponding planned task position, and the reported duration is greater than the maximum expected duration. This characterizes a possible fraud.

4. Experiments

The proposed method has been implemented in Java, on top of a database managed with PostgreSQL 9.1.4 (64-bit) and PostGIS 2.0.3 r11132. The queries were performed in a server with an i7-2670QM 2.20GHz processor, 8Gb RAM, and 750Gb HD 7200 RPM.

We evaluated the implemented prototype of the proposed method on data of a water supply company. Several service requests must be served in a geographic region for each maintenance team, within given deadlines. Each team carries a mobile device running an application that collects its geographic position at every minute. Moreover, this application allows the team to report the execution of each task, supposedly from its execution location. The devices used by teams ensure 10 meters of accuracy. For these experiments, each maintenance team is considered a moving object, and each service request is considered a planned task. The spatiotemporal coordinates of raw trajectories and reported tasks were collected using the GPS sensors of the mobile devices carried by the maintenance teams. The planned tasks were extracted from a company's database, which have only the addresses where the tasks should be performed. Thus, we have used geocoding to infer the spatial coordinates of planned tasks.

The set of daily trajectories used in the experiments contains over 900 thousand spatiotemporal positions, of 11 maintenance teams, collected between January 2008 and September 2013. During this period those teams send 6,468 reported tasks, and there were 8,505 planned tasks for them.

The Google Geocoding API was used to infer the planned tasks' coordinates from the street addresses of the service requests. The CB-SMoT method [Palma *et al*. 2008] implemented in Weka-SMTP [Bogorny *et al*. 2011] was used to extract stops from raw trajectories with the following parameters: 300 seconds as minimum duration to consider a stop, and 0,3 m/s as maximum allowed speed in a stop. It has generated 8,779 stops episodes from the company database. The values of spatial thresholds specified in subsection 3.2 were: `SThreshold_PR` equals to 15 meters; `SThreshold_PT` equals to 20 meters; and `SThreshold_RT` equals to 15 meters.

4.2. Results

The proposed method detected 16,955 inconsistencies. Table 2 shows the total number of inconsistencies classified in each class described in Section 3.3. The 4 columns in the right of this table show the percentages of inconsistencies of each class (C) relative to: the number of inconsistencies found (%); the number of planned tasks (%P); the number of reported tasks (%R); and number of stop episodes (%S). Only 29 tasks showed no inconsistencies (class 1). Possibly, 24% of the tasks were not performed (class 8). Idle stops represent almost half of the inconsistencies found (class 10)

Table 2. Summary of the experimental results

C	Possible explanation	Count	%	%P	%R	%S
1	No inconsistency	29	0.17	0.34	0.45	0.33
2	Short reported task	160	0.96	1.88	2.47	1.82
3	Prolonged reported task	1	0.01	0.01	0.02	0.01
4	Prolonged reported task, short stop, and wrong location of task report	0	0	0	0	0
5	Prolonged reported task, and planned task location not visited	105	0.63	1.23	1.62	-
6	Unreported task and planned location shortly visited	19	0.11	0.22	-	0.22
7	Prolonged reported task at planned location that is not visited.	103	0.62	1.21	1.59	-
8	Task not performed	2,070	12.43	24.34	-	-
9	Report of unknown task	0	0	-	0	-
10	Idle stop	8,249	49.53	-	-	93.96
-	Other classes	6,219	37.34	-	-	-

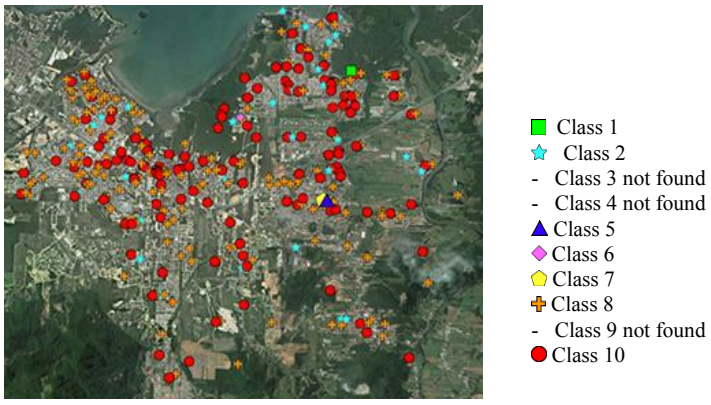

Figure 6. Geographical distribution of considered classes of inconsistencies

Figure 6 presents the geographical distribution of the inconsistencies of the classes presented in Table 2 found in a dataset collected between 1st and 30th June 2013. The cause of so many inconsistencies is currently under investigation yet.

These results illustrate how the proposed method allows the detection of a wider variety of inconsistencies than related work, by considering matching/non-matching trajectories, planned tasks, and reported tasks. In addition, our classification of the detected inconsistencies provides hints for their explanations.

5. Related Work

The work of [Raj *et al.* 2006] fuses data generated by GPS and other sensors to recognize activities performed by moving objects. The used sensors include a 3-axis accelerometer, microphones for recording speech and ambient sound, photo-transistors for measuring light conditions, temperature sensors, and barometric pressure sensors. The use of diverse sensors requires lots of configurations. In addition, the processing of multimodal data can be quite difficult and costly.

Trajectories are compared with a schedule of planned tasks in [Clark & Doherty 2008]. Their goal is to find rescheduled tasks by analyzing their execution locations (GPS coordinates), initial and final instants, planned tasks, and users' feedback. The findings are confirmed by the tasks executors via interviews. Their method does not use reported tasks, detects only temporal inconsistencies, and does not classify them.

The method for describing moving objects tasks and generating diaries introduced in [Huang *et al*. 2010] assumes that tasks along a trajectory occur at Points of Interest (PoIs), where the moving objects remain stationary for a while. According to them a task has a location (e.g., geographic coordinates), a starting time, duration, and a goal. Their method describes tasks according to categories of PoIs previously registered. Therefore, the quantity and the quality of the PoIs set are decisive for their method. They do not consider that a PoI can play different roles for different moving objects (e.g., a restaurant can be a lunch place for a moving object, and a job for another one).

The proposal of [Furletti *et al.* 2013] is similar to the one of [Huang *et al*. 2010], but instead of using PoIs previously registered, it uses PoIs from the Google Place API and OpenStreetMap. The type of a task is determined according to the type of location where the moving object stops to perform that task. Neither [Furletti *et al*. 2013] nor [Huang *et al*. 2010] use data of planned tasks or reported tasks.

Another novel method presented in [Zhenyu *et al*. 2012] generates a travel diary based on connections to Wi-Fi access points, and signal strength of mobile phones. Finally, the method proposed in [Yuan *et al*. 2013] constructs individual movement histories (similar to users' diaries) from a smart card transactions dataset. These works do not consider planned tasks. They rely just on the use of Wi-Fi access points and smart card transactions, respectively, to produce the travel diary information.

Table 3 compares related work. The columns "Trajectory", "Planned tasks" "User's reports", and "Other data sources" indicate if each method uses data from the respective sources. Notice that none of these related works classify spatiotemporal inconsistencies of trajectories with both planned tasks and reported tasks. For the best of our knowledge our proposal is the first one to consider all these data sources.

Consequently, our method is able to detect a wider variety of spatiotemporal inconsistencies than related proposals, and supports the investigation of the reasons for each detected inconsistency. In addition, our method does not impose a rigid schedule to start and finish tasks. It relies only on the set of tasks to be performed, and the expected position and duration of each task. Thus, it allows more flexibility for the moving objects to decide when to realize each task.

Table 3. Comparison to related works

Work	Trajectory	Planned tasks	User's reports	Other data sources	Classification of Inconsistencies
Clark & Doherty 2008	*Yes*	*Yes (Schedule)*	*No*	*No*	*No*
Furletti *et al.* 2013	*Yes*	*No*	*No*	*Google Place API and OpenStreetMap*	*No*
Huang *et al.* 2010	*Yes*	*No*	*No*	*Known PoI's*	*No*
Raj *et al.* 2006	*Yes*	*No*	*No*	*Sensors (accelerometer, microphones)*	*No*
Proposed method	***Yes***	***Yes***	***Yes***	***No***	***Yes***

6. Conclusions and Future Work

This paper introduced a computational method to detect and classify spatiotemporal inconsistencies of trajectories with planned and reported tasks. The main advantages of the proposed method are: (i) the use of 3 kinds of data (plans, tracks, and reports of moving objects performing tasks) to detect inconsistencies; (ii) capacity to detect a wider variety of spatiotemporal inconsistencies than state-of-the-art methods; (iii) higher flexibility for input data than related methods, by not using rigid schedules (with specific times for executing planned tasks); and (iv) several parameters to tune the method according to properties of the analyzed datasets. The returned spatiotemporal inconsistencies among data of distinct nature and sources help to investigate problems and possible misbehaviors of the moving objects (e.g., idle stops, frauds).

Future work comprises: (i) further validation of the proposed method, with distinct datasets and some ground true to asses results quality measures, such as precision and recall; (ii) evaluation of the performance and the scalability of the proposed method with bigger datasets; (iii) association of the geographical location of detected inconsistencies with geographic features of existing databases (e.g. OpenStreetMap) and linked data collections (e.g. LinkedGeoData) to better investigate the possible semantics of these inconsistencies; (iv) analysis of trajectory annotations and/or social media posts made by the moving objects to obtain additional information for explaining their behaviors (e.g., places and events visited, actions, and intentions).

Acknowledgments

Work supported by CNPq (grant 478634/2011-0), and FEESC. Thanks to the colleagues Andre S. Furtado, Juarez A. P. Sacenti, and Ricardo G. B. Nabo for their valuable help.

References

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

Bogorny, V., Avancini, H., de Paula, B. L., Kuplish, C.R., and Alvares, L. O. (2011). *Weka-STPM: a Software Architecture and Prototype for Semantic Trajectory Data Mining*. Transactions in GIS, 15:(2) 227-248

Clark, A. F., Doherty, S. T. (2008) *Use of GPS to automatically track activity re-scheduling decisions*. 8th Intl. Conf. on Survey Methods in Transport.

Furletti, B., Cintia, P., Renso, C., Spinsanti, L. (2013). *Inferring human activities from GPS tracks*. 2nd ACM SIGKDD Intl. Workshop on Urban Computing (UrbComp).

Huang, L., Li, Q., Yue, Y. (2010). *Activity identification from GPS trajectories using spatial temporal POI's attractiveness*. 2nd ACM SIGSPATIAL Intl. Workshop on Location Based Social Networks.

Mountain, D. and Raper, J. (2001). *Modelling human spatio-temporal behaviour: a challenge for location based services*. Intl. Conf. on Geocomputation, pages 24–26, Brisbane, Australia

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

Parent, C., Spaccapietra, S., Renso, C., Andrienko, G., Andrienko, N., Bogorny, V., Damiani, M. L., Gkoulalas-divanis, A., Macedo, J., Pelekis, N., Theodoridis, Y., Yan, Z. (2013). *Semantic trajectories modeling and analysis*. ACM Comp. Surveys, 45(4).

Pelekis, N., Theodoridis, Y. (2014). *Mobility Data Management and Exploration*. Springer, ISBN 978-1-4939-0391-7, pp. 1-298

Spaccapietra, S., Parent, C., Damiani, M.L., Macêdo, J.A., Porto, F., Vangenot, C. (2008). *A conceptual view on trajectories*. Data Knowl. Eng. 65(1): 126-146

Raj, A., Subramanya A., Dieter F., Bilmes, J. (2006). *Rao-Blackwellized particle filters for recognizing activities and spatial context from wearable sensors*. 10th Intl. Symp. on Experimental Robotics. Springer Verlag.

Yan, Z., Chakraborty, D., Parent, C., Spaccapietra, S., Aberer, K. (2013) *Semantic trajectories: Mobility data computation and annotation*. ACM TIST 4(3): 49.

Yuan, N.J., Wang, Y., Zhang, F., XIE, X., Sun., G. (2013). *Reconstructing Individual Mobility from Smart Card Transactions: A Space Alignment Approach*. IEEE 13th Intl. Conference on Data Mining, 877-886

Zhenyu C., Shuangquan W., Yiqiang C., Zhongtang Z., Mu L., (2012) *InferLoc: Calibration Free Based Location Inference for Temporal and Spatial Fine-Granularity Magnitude*. IEEE 15th Intl. Conf. on Comp. Science and Engineering