Individual and Group Activity Recognition in Moving Object Trajectories

Marco Aurelio Beber[1], Carlos Andres Ferrero[2], Renato Fileto[1], Vania Bogorny[1]

[1] Programa de Pós-Graduação em Ciência da Computação
Universidade Federal de Santa Catarina, Brazil
marco.beber@posgrad.ufsc.br,{r.fileto, vania.bogorny}@ufsc.br
[2] Instituto Federal de Santa Catarina, Brazil
andres.ferrero@ifsc.edu.br

Abstract. The knowledge about which activities people do at certain places is useful for several application domains. Existing works for activity recognition from trajectory data assume that only one activity is performed at each place, and do not identify the objects involved in each activity. We claim that several activities may be performed at certain places, such as shopping centers, universities, and others. In this paper, we propose a new method to recognize multiple activities performed at a place by integrating GPS trajectories and social media data, labeling trajectories with activities and the individuals involved in each activity. Experiments show that the proposed solution achieved good results in labeling and recognizing both individual and group activities.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data Mining*; *Spatial Databases and GIS*

Keywords: Activity Inference from Twitter Data, Activity Trajectory, Group Activity Inference, Semantic Trajectories, Social Networks, Trajectory Activity Recognition

1. INTRODUCTION

We are living the era of big data, where individuals are constantly leaving traces of their movements and their activities. Even though we are not fully aware of it, we are being tracked everyday. Our spatiotemporal traces can be delineated as *moving object trajectories*. A raw trajectory is a temporally ordered sequence of spatial positions, which do not present explicit semantics.

A raw trajectory can be split in stops and moves, where stops are the most important parts of a trajectory, and that can be enriched with more semantic information, such as the name of the place visited by the object [Spaccapietra et al. 2008]. Bogorny et al. [2014] proposes the Constant model, with the most important aspects for semantically describing trajectory segments, such as transportation means, environmental conditions, activities, and goals associated with these segments. Fileto et al. [2015] proposes the Baquara ontological framework for semantically enriching trajectories with linked open data. Although the previous works propose a model to represent important aspects related to trajectories, they do not present a solution for individual and group activity inference.

Several recent works address semantic trajectory data analysis for different purposes such as estimating the attractiveness of places [Furtado et al. 2013], semantic outlier discovery [de Aquino et al. 2013], user profile inference [de Alencar et al. 2015], etc. However, only a few works have focused on activity recognition, and even less on group activity inference. While it might be easy to discover visited places in many situations, determining the activities performed at these places and who are the individuals involved in these activities is not a trivial task. There is no unique association be-

This work was supported by the Brazilian agencies CAPES and CNPQ.

tween each visited place, called *Point of Interest* (POI or POI type) and the activities that can be performed at this POI. Several activities may be performed at a single POI, and there is a wide range of possibilities that vary in number and nature. For instance, at a shopping mall one could be eating, purchasing, working, socializing or even watching a movie.

Although there are a few activities that could be a priori labeled as individual or group activities, such as socializing, that is normally performed in groups, most activities can be either performed alone or in group, such as studying, working, walking, etc. There is no unique way to infer human activities from movement data, and it is very challenging, since a group of individuals who do not know each other can still by chance be performing the same activity at the same place.

Some works in the literature are related to human activity recognition using different types of data. For instance, Liu et al. [2012], Weerkamp et al. [2012], and Zhu et al. [2016] perform activity recognition in textual data extracted from social media, such as tweets and tips from Foursquare, but not GPS trajectory data, which is the focus of this article. The few works on GPS trajectory data that recognize activities, such as [Furletti et al. 2013] and [Njoo et al. 2015] ignore an important aspect, that more than one activity can be performed at each POI. Their main limitation is the assignment of only one activity at a POI. In addition, they rely on specialists to manually label POIs (or POI types) with activities that can happen at these POIs (or POI types). Another limitation is the dependence of an annotated trajectory dataset to generate a classification model. Furthermore, the few works on GPS trajectory data that recognize group activities, such as [Hirano and Maekawa 2013; Gordon et al. 2014; Bourbia et al. 2016] also have a limitation, they consider that the group performing the activity is already known, or that the group is defined by a buddy-list containing all individuals.

In this paper we propose a method to integrate GPS trajectories with social media data and census data in order to enrich semantic trajectories represented as stops with activities. We extend our previous work [Beber et al. 2016] to recognize multiple activities that can be performed at certain POIs, and propose a method to discover if these activities are performed in groups. The input of the method is a set of semantic trajectories represented as stops and the output is this set annotated with activities and all individuals participating in each activity.

In summary, we make the following contributions: (i) we present an algorithm to associate multiple activities to trajectory data; and (ii) propose the algorithm G-Activity to infer activities performed by groups of individuals, based on trajectory encounters and relationships.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 presents the basic definitions. Section 4 describes our proposal for individual and group activity recognition. Section 5 presents the experimental evaluation, and finally, Section 6 presents the conclusions.

2. RELATED WORK

There are different works in the literature related to human activity recognition using different types of data, such as social media and GPS trajectories. The works based on social media focus on text classification, and extract features from text and POIs to build classifiers for activity recognition. For instance, Liu et al. [2012] builds a classifier over tweets in order to predict the POI type of tweets linked to Foursquare. Although this work does not recognize activities, it recognizes POI types that can be related to different activities. Weerkamp et al. [2012] proposed an approach to predict the popular activities that will happen in a future time window, such as tonight, tomorrow, and next week, by using a future time-window and keywords related to activities. Zhu et al. [2016] builds a multi-label classifier using tweets manually annotated with activities in order to predict up to three activities. To build the classifier, it considers the tweet text, the tweet posting time, the POI type from Foursquare and POI name from Foursquare.

On the other hand, only a few works try to recognize activities on GPS trajectories. Moreno et al. [2010] proposed an algorithm that given a set of stops, uses a predefined set of rules that considers the minimum time and maximum speed to infer the goal of the movement. However, the set of rules has to be defined by a specialist, and the matching process is based on movement aspects of the goal, such as its minimum time and its maximum speed. Therefore, it ignores important aspects such as the place and the time of the goal. This paper, instead, focuses on recognizing activities, and we do not depend on a specialist, since we extract the knowledge from Twitter in order to build a knowledge base that describes the activities that can be performed at a POI. Furletti et al. [2013] proposed a method for activity recognition where a set of activities is manually defined for POI types, and given a trajectory, it finds the stops and matches the POI type of the stop with the manually defined activities. Our work on the other hand considers that multiple activities can happen at each POI and computes the similarity of the trajectory and the activities in the knowledge base in order to infer activities. Reumers et al. [2013] uses a dataset of semantic trajectories annotated with activities and proposes to infer activities using a decision-tree based model. To build the tree, it uses the start time, duration and activity of each stop, but does not consider the place where the activity happened and depends on the annotated trajectory data to build the model. Kim et al. [2014] builds a classification model to recognize groups of activities, as for instance, home, work and transportation. It uses spatial regions annotated with the frequency of time, duration and frequency of the activities. However, it does not infer activities, just groups of activities, and it also depends on the annotated trajectory data to build the model. Njoo et al. [2015] also manually defines an activity for each POI type, and using a dataset of semantic trajectories annotated with activities it builds classifiers with trajectories from the same moving object, to represent the routine of the object. However, if the moving object goes to a place that was not previously seen, it matches the POI type with the manually defined activity.

Regarding group activity recognition, a few works use trajectory and sensor data. For instance, Hirano and Maekawa [2013] use bluetooth data and a buddy-list to identify individuals performing activities together. If the bluetooth signals of the individuals are near each other, and they are present in each other's buddy-list, then they are in a group activity. The problem with this approach is that it is limited to the individuals in the buddy-list. For instance, a man introducing his girlfriend to his parents at a dinner party. The girlfriend is not in the parents' buddy-list, only in the buddy-list of the man, so she will not be considered as a member of the family group activity. We solve this problem by considering the indirect connection in encounters, which means that if people are performing the same activity together and share a common connection, they are in the same group activity. Bourbia et al. [2016] uses several sensors to collect data about how people interact with the environment and uses a set of rules to identify group activities. The sensor data are used to access an ontology that defines the relationships between sensor data and their corresponding actions. These actions are combined in the form of logic rules to describe group activities. Our approach does not use rules to identify group activities. Instead, we find encounters of individuals that have a strong relationship.

Overall, our work is different from the previous approaches since we do not depend on a specialist to build the knowledge base, instead, we extract the knowledge from Twitter and census data. We also consider that multiple activities can take place at each POI, and we propose an algorithm to match trajectories with the knowledge base to infer activities. Furthermore, we consider the indirect connection in encounters to recognize group activities.

3. BASIC DEFINITIONS

In this section we present the main definitions to understand our proposal. We start with the concept of semantic trajectory, which has several definitions in the literature, such as [Bogorny et al. 2014] and [Spaccapietra et al. 2008]. In this work we simplify the concept of semantic trajectory considering only stops that can be associated with POIs. The stops are the most relevant parts of a trajectory and

represent the part where the object stayed for a minimal amount of time and where he/she probably performed an activity. Definition 3.1 shows the formal definition of semantic trajectory.

Definition 3.1 ***Semantic Trajectory***. A semantic trajectory $S = \langle s_0, s_1, ..., s_n \rangle$ is a sequence of stops, where the ith stop is a tuple $s_i = (x_i, y_i, startTime_i, endTime_i, poi_i)$, with x_i and y_i being the spatial coordinates of the stop at the POI poi_i, starting at $startTime_i$ and finishing at $endTime_i$.

A stop of a trajectory occurs at a place, called Point of Interest (POI), given in Definition 3.2.

Definition 3.2 ***Point of Interest***. A point of interest is a tuple $poi = (type, x, y, ot, ct)$ where $type$ is the type of the POI (e.g. Restaurant, Gym, University, Shopping Mall), x and y its location, and ot and ct are, respectively, the opening and closing hours of the point of interest.

The moving object stops at POIs to perform one or more activities. For instance, a student stops (stays) at a university to study. A worker stops at a restaurant to work as a cook, while clients stop at the restaurant to eat. Furthermore, the activity can be performed alone or by a group of people. The definition of activity is given in Definition 3.3.

Definition 3.3 ***Activity***. An activity is a tuple $a = (act.startTime, act.endTime, label, P)$ where $act.startTime$ and $act.endTime$ are respectively, the start and end times of the activity $label$ and P is the set of moving objects that performed the activity.

We extend the definition of semantic trajectory to cope with activities, and call it activity trajectory, which is given in Definition 3.4.

Definition 3.4 ***Activity Trajectory***. An activity trajectory $T = \langle t_0, t_1, ..., t_n \rangle$ is a sequence of tuples $t_i = (s_i, A_i)$, with s_i being a stop and $A_i = \{a_0, a_1, ..., a_n\}$ being the set of activities performed at s_i.

Different activities can happen at the same POI type. In order to know which are the activities that happen at each POI type we extract information from social media data, such as the frequency that each activity happens at a POI type and the average time it is performed, resulting in what we call a *POI Type Profile*, given in Definition 3.5.

Definition 3.5 ***POI Type Profile***. A POI type profile is a tuple $pro = (POItype, act, meanTime, sdTime, meanDuration, sdDuration, frequency)$, where $meanTime$ is the mean time of the observed occurrences of the activity act at the POI type $POItype$; $sdTime$ is the standard deviation of this time, $meanDuration$ is the mean duration time of act at $POItype$, $sdDuration$ is the standard deviation of this duration, and $frequency$ is the frequency of act at $POItype$ relative to the total number of activity occurrences observed at $POItype$.

In the following section we present the proposed approach for activity recognition.

4. ACTIVITY RECOGNITION FROM MOVEMENT DATA AND SOCIAL NETWORKS

Our approach to infer activities has three main steps. The first one is to build an activity knowledge base in the form of POI type profiles, which is extracted from Twitter, Foursquare and census data. The second step is to infer individual activities in trajectories by matching the stops of the moving object with the POI type profiles in the knowledge base. The third step is to infer group activities, by computing the encounters of the moving objects at the same POIs and their relationship degree. These steps are described in the following sections.

4.1 Building the Knowledge Base

The knowledge base is a representation of the distribution of activity time and duration that happen at each POI type. It contains the following information: POI type, activity name, average duration, standard deviation of the duration, average time, standard deviation of activity time and relative frequency. Table I shows an example of the knowledge base for the POI Type *Shopping Mall*. For instance, based on social media data analysis it is possible to measure that the average time that people are watching movies at a *Shopping Mall* is at 16.35 (16h21min), with a standard deviation of 3.12 (03h07min). The average time (duration) that people stay at a *Shopping Mall* watching a movie is 2.27 (02h16min), with a standard deviation of 0.77 (46min), and the relative frequency of the activity watching movies at a *Shopping Mall* is 10%.

Each attribute is described as follows: (i) POI Type is extracted from Foursquare, and it is present in the tweet to show where the activity happened; (ii) Activity Name is the activity we want to infer; (iii) Average Duration is the average time people spend doing an activity at a POI type. We can extract this information from any set or taxonomy of activities; (iv) Duration Std Dev is the standard deviation of the average duration. This information can be extracted from any set or taxonomy of activities; (v) Average Time is the post time of the activity at the POI type. This information is extracted from the tweets; (vi) Time Std Dev is the standard deviation of the average time. This information is extracted from the tweets; and finally (vii) Relative Frequency is the proportion of tweets of an activity that happened at the POI type.

Algorithm 1 describes how we build the POI type profiles knowledge base. The inputs of this algorithm are a set of georeferenced tweets annotated with activities (more details are given in Section 5.1), and a census dataset, which consists of activity diaries containing for each activity the place where it was performed (POI), and the time spent performing the activity. We use the census dataset as complementary data for each activity, because only the posting time of each tweet is not enough to infer activities.

This algorithm iterates the set of tweets (lines 4 to 8), extracting the posting time of the tweets to an auxiliary structure organized by POI types and activities (line 5). Then, for each tweet it adds one to the frequency of the respective POI type and activity in the same auxiliary data structure (line 6), and adds one to the frequency regardless of the activity (line 7) in order to obtain the relative frequency of the activities at each POI type. After that, it iterates the auxiliary structure A (lines 9 to 18) and obtains the POI type and activity of each instance (lines 10 and 11). After that, it calls the method $C.getDuration$, which has a list of durations for each POI type/activity, to get the activity duration list filtered by POI type and activity (line 12), and calculates the mean and standard deviation of the durations (lines 13 and 14). Then, it calculates the mean and standard deviation of the tweet posting times (lines 15 and 16), and the relative frequency of the activity for the respective POI type (line 17). Finally, it returns the POI type profiles as the knowledge base K (line 19). This algorithm iterates only once the set of tweets and the auxiliary dictionary A of POI types and activities. So the complexity is $O(n_d + n_t * n_a)$, where n_d is the number of tweets, n_t is the number of unique POI types, and n_a is the number of unique activities.

Table I: Knowledge Base for POI Type *Shopping Mall*

POI Type	Activity Name	Avg Time	Time Sd	Avg Duration	Duration Sd	Relative Frequency
Shopping Mall	Consumer Purchases	13.98	3.57	0.74	0.86	0.17
Shopping Mall	Socializing, Relaxing, and Leisure	14.57	4.00	0.78	0.99	0.45
Shopping Mall	Eating & Drinking	14.17	3.52	0.63	0.48	0.28
Shopping Mall	Movies	16.35	3.12	2.27	0.77	0.10

Algorithm 1 Knowledge-Base Builder

Require:
D // corpus of tweets
C // census dataset
1: K = empty dictionary;
2: A = empty dictionary;
3: T = empty dictionary;
4: **for each** $tweet$ in D **do**
5: $A[tweet.poi.type, tweet.act].time.append(tweet.time)$;
6: $A[tweet.poi.type, tweet.act].frequency \mathrel{+}= 1$;
7: $T[tweet.poi.type].frequency \mathrel{+}= 1$;
8: **end for**
9: **for** $i = 0; i < A.size(); i = i + 1$ **do**
10: $ptype = A.getPOIType(i)$;
11: $act = A.getActivity(i)$;
12: $duration\_list = C.getDuration(ptype, act)$;
13: $K[ptype, act].meanDuration = mean(duration\_list)$;
14: $K[ptype, act].sdDuration = sd(duration\_list)$;
15: $K[ptype, act].meanTime = mean(A[ptype, act].time)$;
16: $K[ptype, act].sdTime = sd(A[ptype, act].time)$;
17: $K[ptype, act].frequency = A[ptype, act].frequency \ / \ T[ptype].frequency$;
18: **end for**
19: **return** K;

In the next section we describe the algorithm T-Activity and show how to infer activities using the knowledge base.

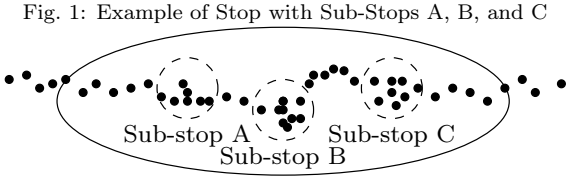
4.2 Individual Activity Recognition

The moving object can perform multiple activities at a POI, so the question is how to distinguish, for instance, if a person is purchasing items at a store, eating at a restaurant or watching a movie at a cinema, when we only know that this person is in a shopping center? In addition, even when this person is in a smaller POI, such as a cafe, there are many possible activities that this person can be doing there, such as eating, having just a drink, reading, socializing, working, and so on. Thus, it is necessary first to determine sub-stops in smaller places that can be inside a bigger place such as a shopping center, and then, for each sub-stop, determine the activity performed.

In order to identify multiple activities at the same POI, we propose to split each stop at a big place into sub-stops, i.e. in smaller places contained in the bigger place. This can be done by using the method to detect sub-stops proposed by Moreno et al. [2010]. For example, let us consider Figure 1 as a trajectory of a student that has a stop at a university. Inside this stop, he has a sub-stop at a classroom (A), a sub-stop at the laboratory (B) and a sub-stop at the university cafeteria (C). By using the concept of sub-stops we can identify more than one activity at each stop.

The speed of the movement of an object inside a shopping mall is more similar than the movement speed outside, where the object is moving between places. The methods to compute stops, such as CB-SMoT [Palma et al. 2008] are not able to distinguish different sub-stops inside a single place as a shopping mall. To solve this problem, Moreno et al. [2010] proposed a method to find sub-stops inside

Fig. 1: Example of Stop with Sub-Stops A, B, and C



stops, which considers the points inside each stop as a trajectory and runs the algorithm CB-SMoT a second over these points. The formal definition of sub-stop is given in Definition 4.1.

Definition 4.1 ***Sub-Stop***. A sub-stop is a tuple $sub = (s, t.startTime, t.endTime, x, y)$ where sub is the sub-stop that happens inside the stop s, starting at $t.startTime$ and finishing at $t.endTime$, with x and y being the coordinates of the centroid of the sub-stop.

In order to recognize activities, the objective is to find the activity in the POI type profiles that is more similar to the sub-stops in the trajectories. Therefore, we compute the time and duration similarity between the POI type profiles and the sub-stops. The time similarity is computed in Equation 1, where K is the POI type profile, st is the sub-stop start time, $meanTime$ is the average time the activity starts in the POI type profile and $K_{sdTime}/K_{meanTime}$ is the variation coefficient of the time in the POI type profile.

$$TimeSim(K, st) = 1 - \left| \frac{K_{meanTime} - st}{K_{meanTime}} \right| * \frac{K_{sdTime}}{K_{meanTime}} \tag{1}$$

The idea behind this formula is to obtain the similarity between the activity which can be performed at the POI type and the sub-stop. We consider the similarity as the inverse of the distance between the start time of the sub-stop and the average time in the profile. Also, we use the coefficient of variation to penalize the similarity score, such that as the standard deviation grows in relation to the mean, the similarity score decreases.

The similarity of the duration of the sub-stop and each activity in the POI type profile is computed using Equation 2, where K is the POI type profile, d is the sub-stop duration, $meanDur$ is the average duration of the activity in the POI type profile and $K_{sdDur}/K_{meanDur}$ is the variation coefficient of the duration in the POI type profile.

$$DurationSim(K, d) = 1 - \left| \frac{K_{meanDur} - d}{K_{meanDur}} \right| * \frac{K_{sdDur}}{K_{meanDur}} \tag{2}$$

We consider the similarity as the inverse of the distance between the duration of the sub-stop and the average duration in the profile. We also use the coefficient of variation $K_{sdDur}/K_{meanDur}$, to penalize the similarity score, which means that as the standard deviation grows in relation to the mean, the similarity score decreases, helping us to distinguish activities with similar durations.

If the score between the activities of the same POI is too similar, the time and duration cannot describe which activity happened. Therefore, our model also considers the frequency of the activities to help us to distinguish between activities. However, using the frequency of the activities can lead to activities that cannot be inferred if the frequency is too low. This is seen in Equation 3, where K is a POI type profile, $K_{frequency}$ is the relative frequency of the activity at the POI type, d is the sub-stop duration, st is the sub-stop starting time, and $DurationSim$ and $TimeSim$ are respectively, the duration similarity function and the time similarity function.

$$ModelSim(K, d, st) = K_{frequency} * DurationSim(K, d) * TimeSim(K, st) \tag{3}$$

Algorithm 2, called T-Activity, describes the activity recognition. It receives as input a semantic trajectory S, a knowledge base in the form of POI Type profiles K, a radius $radius$ that is used to create a buffer at the centroid of each sub-stop to check if the sub-stops intersect each other, and the parameters of the algorithm ESD [Furtado and Bogorny 2017], to find the sub-stops in the same way as proposed by Moreno et al. [2010].

Algorithm 2 T-Activity

Require:
S // semantic trajectory
K // set of POI Type Profiles
$radius$ // radius to intersect sub-stop centroids
$ratio, minTime$ // ESD parameters
1: $T = computeSubStops(S, ratio, minTime)$;
2: **for each** *stop* in T **do**
3: **for each** *sub* in *stop* **do**
4: $area = buffer(sub.x, sub.y, radius)$;
5: $near\_substops = intersect(area, stop)$;
6: $duration = sum(near\_substops.getDuration())$;
7: $freq = getFrequency(K, sub.s.poi.type)$;
8: $score\_time = getSimTime(K, sub.s.poi.type, sub.st)$;
9: $score\_duration = getSimDuration(K, sub.s.poi.type, duration)$;
10: $ranked\_activities = getRankedActivities(freq, score\_time, score\_duration)$;
11: $sub\_act = max(ranked\_activities)$;
12: $stop.A.append(sub\_act)$;
13: **end for**
14: **end for**
15: **return** T

The algorithm starts by initializing the activity trajectory T, computing the sub-stops by calling the function *computeSubStops*. It calls the algorithm ESD, which is a clustering algorithm based on time and distance variation between neighbor points of the trajectory. It evaluates the trajectory points and finds clusters where neighbor points are within a given distance and the time difference between the initial point and last point of the stop is greater than a given threshold. If no sub-stop is found, it considers the whole stop as the sub-stop (line 1). After that, it iterates the stops and sub-stops of the trajectory (lines 2 to 14). Then, for each sub-stop, it creates a buffer with a radius of size *radius* from the sub-stop centroid (line 4) and calls the method *intersect* to find all sub-stops that intersect this buffer (line 5). The algorithm groups the near sub-stops because we are interested in the whole time that was spent performing the activity. For instance, an individual works for 08h00min, but it leaves the company for lunch. By using sub-stops, we would find two sub-stops of 04h00min, hence we group them to account for one activity and sum their duration, resulting in 08h00min. After that, the algorithm computes the similarity between the sub-stop *sub* and the activities in the knowledge base K. First, it gets the set of relative frequencies for the activities at the POI type of the stop from the knowledge base K (line 7). Then it computes the set of time similarity scores for the activities at the POI type of the stop using Equation 1 (line 8). Then it computes the set of duration similarity scores for the activities at the POI type of the stop using Equation 2 (line 9). Then, it computes the score of the set of the activities by multiplying the activity scores of each set using Equation 3 (line 10), and selects the activity with the highest score (line 11). Finally, it appends the activity with the highest score (line 12), and returns the activity trajectory (line 15). The complexity of this algorithm is $O(n_s * n_{sub}^2)$, where n_s is the number of stops and n_{sub} is the number of sub-stops. Also, as we execute algorithm ESD outside the loop, it adds $O(n_p)$ to the complexity, where n_p is the number of points of the stops. In the following section we describe the proposed approach for group activity recognition.

4.3 Group Activity Recognition

People perform different activities at different types of places, alone or in groups. The problem to determine group activities is that people that are together not necessarily are performing group activities. They can be close just by chance in many situations. For instance, people having dinner at a restaurant are not necessarily interacting with each other. Proximity is not an option to solve this problem, because data with sufficient precision is usually not available, and people can share tables

or be close just due to limited space in a POI. Therefore, in this work a *group activity* is an activity that is performed by a group of people that have a certain relationship degree and that are at the same place, at the same time.

In order to identify groups of individuals with potential to be doing something together, we identify encounters, which are two or more moving objects meeting at the same place, at the same time, and for a minimum amount of time. We can do that by comparing trajectories and checking if the respective moving objects are near to each other at the same time and stay together for a certain time. For instance, when students meet at the classroom, when people are commuting by public transport, and so on. In this article we use the concept of encounter defined in [Santos et al. 2015].

Definition 4.2 ***Encounter***. Let $R_1 = \langle p_1, p_2, p_3, ..., p_n \rangle$ and $R_2 = \langle q_1, q_2, q_3, ..., q_m \rangle$ be two trajectories. Let $straj_1 = \langle p_a, p_{a+1}, ..., p_{a+u} \rangle$ and $straj_2 = \langle q_b, q_{b+1}, ..., q_{b+v} \rangle$ be two subtrajectories of R_1 and R_2 respectively. R_1 and R_2 have an encounter at two maximal subtrajectories $straj_1$ and $straj_2$ w.r.t a spatial distance Δ_d, a temporal tolerance Δ_t and a minimum duration $minTime$ IIF the following conditions hold:

—$\forall p_i \in straj_1, \exists q_j \in straj_2 |$ spatialDist(p_i, q_j) $< \Delta_d \wedge$ temporalDist(p_i, q_j) $< \Delta_t$
—$\forall q_j \in straj_2, \exists p_i \in straj_1 |$ spatialDist(q_j, p_i) $< \Delta_d \wedge$ temporalDist(q_j, p_i) $< \Delta_t$
—(min(p_{a+u}.t,q_{b+v}.t) - max(p_a.t,q_b.t)) > minTime

where the functions $spatialDist()$ and $temporalDist()$ compute, respectively, the Euclidean distance and the temporal distance between the points p_i and q_j.

Although encounters can be used to infer group activities, the fact that objects stay together in space and time does not mean that they are doing a joint activity. For instance, objects which are together in a park are not necessarily doing joint activities and may not even know each other. In this work we assume that a group activity involves people that have a relationship and/or are indirectly connected by a common relationship in an encounter. Therefore, we use the relationship degree of individuals, proposed in [Santos 2016], to infer group activities. In this article we use the algorithm MORE++ [Santos 2016] to compute the relationship degree by weighting scores based on the number of encounters objects have at different days, the number of different places where objects meet, and the amount of time people spend together. The formal definition is given in Definition 4.3.

Definition 4.3 ***Relationship Degree***. Let DB $= e_1, e_2, ..., e_n$ be a set of encounters w.r.t Δ_d, Δ_t and $minTime$ of all sets of moving objects in a trajectory database. Let $E(o_i, o_j)$ denote the set of all encounters between o_i and o_j. The relationship degree between o_i and o_j is computed as:

$$R(o_1, o_2) = R_f(o_1, o_2).w_{R_f} + R_d(o_1, o_2).w_{R_d} + R_a(o_1, o_2).w_{R_a} \quad (4)$$

where $w_{R_f} + w_{R_d} + w_{R_a} = 1$ and R_f, R_d and R_a being respectively, the relationship based on the frequency of the encounters, the relationship based on the duration of the encounters and the relationship based on the number of different areas of the encounters, with each of these functions representing the proportion of days, duration, and different areas the objects had encounters.

The relationship degree is a number between zero and one, and the higher the number (closer to one) the stronger is the relationship. Lower relationship degrees may reveal individuals that had encounters by chance and may not necessarily have a relationship. Therefore, we use the relationship degree as an indicator whether people know each other or not, which is the information we need to recognize group activities. We select only strong relationships, and represent them as the edges of a graph, where the vertices represent the moving objects, as shown in Figure 2.

Table II: Example for Area Relationship Degree

Oid	Rel. Degree
A,B	3/5 = 0.6

Oid	Rel. Degree
B,C	3/5 = 0.6

Oid	Rel. Degree
A,B,C,D,E,F	1/5 = 0.2

To better understand why weak relationships are not considered, let us observe Figure 3, that shows a group of individuals A, B, C, D, E, F having an encounter at a restaurant. Looking at the figure we can notice that although all objects are at the restaurant, having an encounter, they are not doing a group activity, since they are at different tables. To discover if the objects are in fact doing joint activities we analyze all previous encounters of all objects, giving us a higher probability that the individuals did not meet by chance. For instance, let us consider that objects in the encounter A, B, C, D, E, F also had the following encounters: $e_1 = (A, B)$ at the cafeteria, $e_2 = (B, C)$ at the university, $e_3 = (A, B)$ at the mall, and $e_4 = (B, C)$ at work. For this example, we compute the relationship degree as the number of different places the individuals met, dividing these numbers by the maximum number of encounters of an individual, which is object B with 5 encounters, resulting in the scores showed in Table II. For objects that have strong relationships, i.e., the relationship degree is higher than a given threshold, suppose 0.5 in this example, we build a graph. By this graph we exclude individuals D, E, F, as the weaker relationship degree indicates that they met by chance.

Going back to our previous example, although individuals A, C have a weak relationship because they only met once at the restaurant in Figure 3, we still cannot exclude them from the group activity inference because they are connected by a common individual. Filtering strong relationships increases the probability that people that are together are performing a joint activity, but it does not mean that people with weak relationships cannot perform activities in group. For those cases, we check individuals that are indirectly connected through the relationship graph, as given in Definition 4.4.

Definition 4.4 ***Encounter Indirect Connection***. Let oid_1, oid_2 and oid_3 be moving objects in the encounter E. To represent the indirect connection between the individuals in the encounter, we adopt the concept of transitivity, which states:

—$oid_1, oid_2, oid_3 \in E : hasConnection(G, oid_1, oid_2) \wedge hasConnection(G, oid_2, oid_3) \Rightarrow hasConnection(G, oid_1, oid_3)$

where the function hasConnection computes a breadth-first search in the graph G to check if oid_1, oid_3 share the common connection oid_2.

Fig. 2: Strong Relationship Degrees (A,B and B,C)

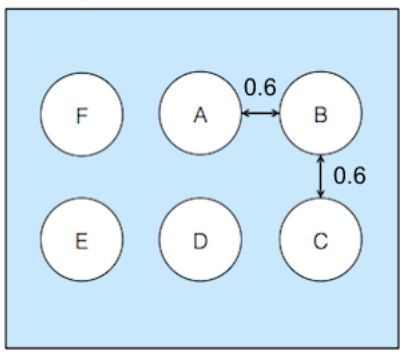
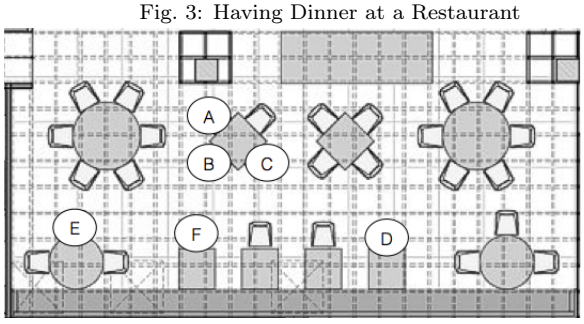


Fig. 3: Having Dinner at a Restaurant



Source: http://santosesantosarquitetura.com.br

Algorithm 3 Filter Encounters

```
Require:
    G // relationship graph
    E // encounter
 1: FilteredEncounters = empty list;
 2: index = 0;
 3: individuals = E.oids; //hashset
 4: vertices = G.vertices[individuals];
 5: for each vertex in vertices do
 6:     edges = vertex.edges;
 7:     if FilteredEncounters.size() == 0 then
 8:         Group = new empty list;
 9:         FilteredEncounters.add(Group)
10:         FilteredEncounters[index].add(vertex);
11:     else
12:         if vertex not in FilteredEncounters[index] then
13:             index = index + 1;
14:             Group = new empty list;
15:             FilteredEncounters.add(Group)
16:             FilteredEncounters[index].add(vertex);
17:         end if
18:     end if
19:     for edge in edges do
20:         if edge in individuals then
21:             FilteredEncounters[index].add(edge);
22:         end if
23:     end for
24: end for
25: return FilteredEncounters;
```

Algorithm 4 G-Activity

```
Require:
    T // activity trajectories
    G // relationship graph
    E // set of encounters
 1: for each encounter in E do
 2:     filEnc = filterEncounters(G, encounter);
 3:     for each traj in T[encounter.trajectories] do
 4:         for each filtered in filEnc do
 5:             if traj.oid in filtered.oids then
 6:                 traj.P.add(filtered.oids);
 7:             end if
 8:         end for
 9:     end for
10: end for
```

Having described the definitions, Algorithm 3 shows how to use the graph to filter encounters to find group activities. It receives as input the graph representing the relationship degrees and an encounter to analyze. It starts by initializing an empty set of filtered encounters (line 1), the index of the filtered encounter (line 2), the individuals involved in the encounter (line 3) and then filtering the vertices that represent the individuals (line 4). After that, it iterates the vertices of the graph G (line 5 to 24). It extracts the edges of the vertex (line 6) and checks whether it is the first iteration or not (line 7). If so, it creates a list to store the individuals that have a relationship (line 8), and adds the vertex (individual) to this group (line 9 and 10).

After that it iterates the edges (relationships) of the vertex (individual) (line 19 to 23) and checks if each edge is from any individual in the encounter E (line 20). If this is the case, it adds the edge to the filtered encounter (line 21). For the next iterations of the vertices, it checks whether the vertex was already filtered out (line 12). If not, it creates a new list that represents a group of individuals that have a relationship (line 14), and adds the vertex (individual) to this group (lines 15 and 16). Finally, it returns the set of filtered encounters (line 25). As an example, if the input is an encounter of one couple plus 4 people that do not have a strong relationship, the output will be just the couple, as they have a strong relationship. As this algorithm iterates the vertices of the graph G once and all the edges of the graph G once it has a complexity of $O(n_v + n_e)$, where n_v is the number of vertices of the relationship graph G and n_e is the total number of edges from the vertices that are iterated. The idea of the algorithm is similar to a Breadth-First Search (BFS), however, as the relationship graph is represented by a centered vertex connected to several other vertices (similar to a star), we perform several BFS, searching only one level for each vertex and checking if the discovered edges are in the encounter E. Also, as we represent the individuals of the encounter E as a hashset, we maintain a linear behavior.

Finally, algorithm 4, called G-Activity, describes how we recognize group activities. It has as input a set of activity trajectories T, a relationship graph G and a set of encounters E. This algorithm is the most expensive to compute, as it has to compute the algorithm *filterEncounters* for each encounter in the set E (lines 1 and 2). It iterates the trajectories and matches them with the filtered encounters (lines 3 and 4). Then it checks if the moving object is part of the filtered encounter, and adds all individuals of the encounter as participants of the group activity (lines 5 and 6). As an example, if the input is a set of three trajectories from one encounter, where two individuals know each other, the algorithm will annotate two trajectories with two participants (activity performed in group) and one trajectory with just one participant (activity performed alone). The time complexity of the algorithm is $O(n_e * (n_g + n_t * n_f))$, where n_e is the total number of encounters, n_g is the time complexity of the algorithm 3, which iterates the graph G in linear time, n_t is the number of activity trajectories T and n_f is the number of filtered encounters.

5. EXPERIMENTAL EVALUATION

In this chapter we evaluate the proposed approach with three different datasets, two GPS trajectory datasets and one synthetic trajectory dataset generated over activity diaries extracted from census data. The GPS trajectories were collected in Florianópolis, Brazil, and Pisa, Italy, and are annotated with activities. The dataset generated from census data is the North American census, which contains the activities that a person performs during a day, the place the activity is performed and for how long. For building the knowledge base we use a dataset of georeferenced tweets generated from Foursquare collected for the period of 14/09/2010 to 11/05/2015. Table III shows the list of activities an its frequency in each dataset.

We start the experimental evaluation presenting the knowledge base construction in Section 5.1 and the evaluation of T-Activity and G-Activity in the remaining sections.

5.1 Building the Knowledge Base

We gathered tweets from the Twitter Public Streaming API to feed the knowledge base. We selected 137,509 georeferenced tweets generated from Foursquare from 14/09/2010 to 11/05/2015. Took the ones written in Portuguese, with coordinates inside Brazil's bounding box, and with at least 3 words. To build the knowledge base, the first step was to extract the POI information from Foursquare, using the Venue Search API. We did that by looking the Venue at the coordinates present in the tweet. As a result we had the tweet text, the tweet time, the POI type and the POI name. Then, for this experiment we filtered the tweets by the following POI types: Restaurant, Gym, Supermarket, University and Shopping Mall, which are the existing types in the trajectory datasets. The result was a corpus with 45,209 tweets.

Table III: Dataset Statistics

Activity	Twitter Dataset	Florianópolis Dataset	Pisa Dataset	Census Dataset
Socializing, Relaxing, and Leisure	295	3	33	344
Eating & Drinking	548	38	3	1315
Education	170	53	0	342
Consumer Purchases	222	21	21	3033
Sports, Exercise, and Recreation	226	13	0	209

Activity	Twitter Dataset	Florianópolis Dataset	Pisa Dataset	Census Dataset
Movies	31	0	0	0
Working	0	0	29	0
Waiting the Bus	0	0	8	0
Healthcare	0	0	1	0
Total	**1492**	**128**	**95**	**5243**

Table IV: Entire Knowledge Base

POI Type	Activity Name	Avg Time (hrs)	Time Std Dev	Avg Duration (hrs)	Duration Std Dev	Rel. Freq. (%)
Shopping Mall	Consumer Purchases	13.98	3.57	0.74	0.86	0.17
Shopping Mall	Socializing, Relaxing, and Leisure	14.57	4.00	0.78	0.99	0.45
Shopping Mall	Eating & Drinking	14.17	3.52	0.63	0.48	0.28
Shopping Mall	Movies	16.35	3.12	2.27	0.77	0.10
Supermarket	Consumer Purchases	15.12	4.52	0.69	0.52	0.73
Supermarket	Eating & Drinking	14.50	5.17	0.54	0.47	0.27
Restaurant	Eating & Drinking	12.24	6.83	1.00	0.62	0.78
Restaurant	Socializing, Relaxing, and Leisure	14.60	6.59	1.47	1.34	0.18
Restaurant	Consumer Purchases	14.51	5.28	0.16	0.19	0.04
University	Socializing, Relaxing, and Leisure	14.52	5.69	0.73	0.88	0.02
University	Education	13.38	5.37	3.16	1.90	0.96
University	Eating & Drinking	13.96	4.51	0.51	0.30	0.02
Gym	Sports, Exercise, and Recreation	13.25	5.98	0.99	0.67	1.00

To identify the activities from tweets, we follow the method proposed by Zhu et al. in [Zhu et al. 2016], which consists in building a classification model to assign each tweet to an activity. We randomly selected a sample of tweets stratified by POI type, with the size determined by a confidence level of 95% and a confidence interval of 5%, and manually classify each one to an activity present in the ATUS taxonomy accordingly to the text of the tweet. We tried to use verbs to identify activities, but the results were not satisfactory, since the amount of geo-referenced tweets containing verbs was insignificant. Therefore we ignored the grammatical class of the words and adopted the metric TF-IDF to weight the words.

Having the annotated tweets, we build a classification model considering the following features: (i) POI Type: as each tweet is georeferenced to a Foursquare POI, we extract the POI type and construct a matrix containing 11 binary features, each one representing a general POI type; (ii) Tweet Text: by extracting the most relevant uni-grams and bi-grams weighted by TF-IDF, we obtain 9394 features from the text; (iii) POI Name: the same way we extract features from the tweet text we extract the POI name, as some POI names can be indicative of activities (e.g. Japanese Restaurant, Fourth Street Market); Posting Time: we chunk the tweet posting time by hour of the day to construct a matrix of 24 features. We combine the previous features in a matrix and build a SVM linear model with L1-regularization for feature selection and squared hinge loss. We evaluate the tweet classification model by splitting the manually labeled dataset into training and testing data, with a proportion of 70/30, obtaining an accuracy of 78.57%, and average precision of 83.33%, and an average recall of 75.83%. After building and evaluating the classification model, we classify the remaining tweets. In addition, we run Algorithm 1 to extract the mean and standard deviation of time and duration and also the relative frequency of the activities to store in the knowledge base. Table IV shows the knowledge base generated for this experiment, where the majority of the POI types have multiple activities. For instance, *Shopping Mall* has four different activities, where the most common activity is *Socializing, Relaxing, and Leisure*. It is important to notice that the average time is an approximation of multiple distributions, which explains the high standard deviations of time.

5.2 Dataset Description

The Florianopolis GPS trajectory dataset was collected from 14/04/2016 to 08/06/2016 by 8 participants in the city of Florianópolis, Brazil. As we can see in Table III, this dataset contains 21 *Consumer Purchases* that happened at supermarkets and malls, 38 *Eating & Drinking* that happened at restaurants and universities, 13 *Sports, Exercise and Recreation* that happened at gyms, 53 *Education* that happened at universities and 3 *Socializing, Relaxing and Leisure* that happened at universities and restaurants. In addition, the dataset is composed of 59 trajectories, 100 stops and 128 sub-stops labeled by the participants.

The Pisa dataset has 38 trajectories collected during 14 days by 7 users and are manually annotated with stops, individual activities and group activities. Although this is a small dataset to validate our approach, it is important since there are not many ground truth trajectory datasets for activity recognition. It contains 95 stops, where 29 happened at *Home*, 29 at *University/Research Institute*, 21 at *Market*, 8 at *Bus Stop*, 4 at *Bar*, 1 at *Hospital*, and 3 at *Bakery*.

Considering the difficulty for obtaining a large semantic trajectory dataset with ground truth, we use a dataset generated from the ATUS dataset [Shelley 2005]. This dataset consists of activity diaries, where each diary corresponds to the semantic trajectory of a resident of the United States of America. The diary contains the activity, the place where the activity was performed (POI), and the start and end times of the activity. From this dataset we selected all households that have activity entries with more than 5 days, resulting in 41 households with 2440 stops and 5243 sub-stops. In order to identify sub-stops, we considered as stop every POI where an activity is performed and, as we have multiple entries at the same POI, we consider them as sub-stops. Table III shows the distribution of activities in this dataset, which has 3033 *Consumer Purchases*, 1315 *Eating & Drinking*, 209 *Sports, Exercise and Recreation*, 342 *Education* and 344 *Socializing, Relaxing and Leisure*.

5.3 Evaluation of T-Activity

To evaluate T-Activity we perform experiments in two different datasets. First we evaluate it on the Florianópolis dataset, where we consider as input the POI type profiles in Table IV and set the parameter *radius* to 10 meters. Figure 4 shows the f1-score in comparison to Furletti et al. [2013], Reumers et al. [2013], and Njoo et al. [2015].

On analyzing the f1-score we can see that T-Activity presents the best result. In addition, Furletti and Njoo have the same score as our approach where the main activity is the only activity at the POI, such as *Consumer Purchases* for *Supermarket* and *Sports, Exercise and Recreation* for *Gym*, otherwise it has a lower score. Reumers on the other hand, only considers the duration and start time of the activities, and as some activities have similar start times and durations the f1-score is affected. It is important to notice that our work is the only one able to recognize the activity *Socializing, Relaxing and Leisure* as it is not the main activity of any place.

In addition, on considering the difficulty for obtaining a semantic trajectory dataset with ground truth, we evaluate T-Activity with a dataset generated from the ATUS dataset [Shelley 2005], and considered as input the POI Type Profiles in Table IV, generated from the Twitter dataset. Figure 5 shows the f1-score in comparison to Furletti et al. [2013], Reumers et al. [2013], and Njoo et al. [2015].

Analyzing the f1-score we can see that our work outperforms the existing works for almost all activities. The work of Furletti and Njoo have a lower score for almost all classes. This happens because each POI type is matched to an exclusive activity, and our method considers the similarity of the activities along the relative frequency of the activities. However, considering the relative frequency

Fig. 4: Comparison of f1-Score at Florianopolis Dataset

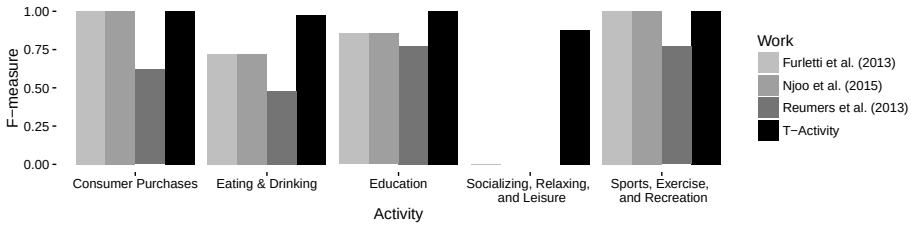


Fig. 5: Comparison of f1-Score at Census Dataset

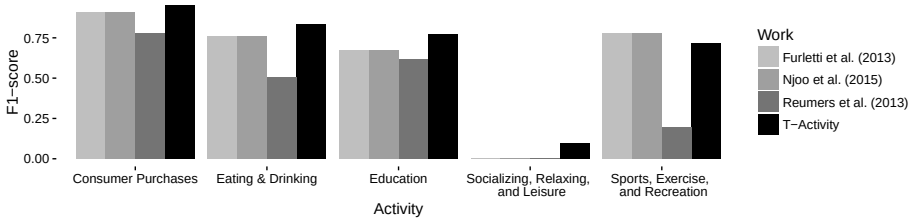


Table V: Results for Group Activities at Pisa Dataset

Participants	Activity			
	Socializing	Working	Purchasing	Waiting the Bus
1, 2	8/8	2/2	2/2	0/0
1, 3	2/2	4/4	0/0	0/0
2, 3	6/6	4/4	0/0	0/0
1, 2, 3	9/9	12/12	6/6	6/6
5, 6, 7	3/3	3/3	0/3	0/0

Table VI: Relationships at Pisa Dataset

Object Id	Relationship Degree
1,2,3	0.407
1,2	0.788
1,3	0.467
2,3,4	0.284
2,3	0.536
5,6,7	0.479

is problematic for activities that are too similar and have a low frequency, such as *Socializing, Relaxing and Leisure*. On the other hand, as Reumers does not consider the POI type, the f1-score has the lowest result for all classes.

5.4 Evaluation of G-Activity

To evaluate G-Activity we perform experiments on the Pisa dataset. As the stops are annotated with activities that happened in groups, we analyze the accuracy of our method for group activity recognition. In total, 27 out of 28 encounters were correctly detected in this dataset using the algorithm *BeingTogether* with parameters Δ_t = 300sec (time tolerance), Δ_d = 50m (distance threshold), and $minTime$ = 10 minutes (minimal time duration for an encounter). Table VI shows the relationship degree of the individuals, where each row represents the relationship of a group with a score ranging from 0 to 1, with 0 being the weakest relationship and 1 the strongest. We computed the relationship degrees using algorithm MORE++ with parameters $w_{R_d} = 0$ (weight for feature duration), $w_{R_a} = 0.5$ (weight for feature area), and $w_{R_f} = 0.5$ (weight for feature frequency). We ignored the feature duration due to some gaps in the dataset affecting the score.

As input to *G-Activity*, we use the Pisa dataset, the set of detected encounters, the relationship degree in Table VI and a relationship threshold $\Delta_r = 0.5$ to prune weak relationships. Table V summarizes the output of *G-Activity* to show the groups that performed activities together and which activities they performed in comparison to the ground-truth, where each row represents a group that performed activities together, one or more times. In this table we can see that the activities *Socializing*, *Working* and *Waiting the Bus* were all correctly identified, while activity *Purchasing* had 8 out of 11 instances correctly identified. In fact, only 3 *Purchasing* activities from participants *5,6,7*

were not identified. This happened because one of their encounters was not found by the algorithm *BeingTogether*, resulting in activities that could not be found by *G-Activity*. Overall, *G-Activity* correctly found 67 out of 70 activities performed in groups, resulting in an accuracy of 95.71%.

6. CONCLUSION

In this article we proposed a method for activity recognition in moving object trajectories, extracting the possible activities from tweets posted at POI types and matching these activities with the stops of moving object trajectories. Indeed, we proposed a new method for group activity recognition, identifying encounters and checking whether the participants have relationships with each other and are performing the same activity. Even though activity recognition is broadly performed with different types of data, infer individual and group activities in moving object trajectories is not a trivial task. As future work, we will go deeper in the activity analysis, defining and recognizing unusual activities, using Gaussian Mixture Models to calculate the statistics for time and duration in the knowledge base and improving the score functions by using a probability density function instead of the frequency.

REFERENCES

Beber, M. A., Ferrero, C. A., Fileto, R., and Bogorny, V. Towards Activity Recognition in Moving Object Trajectories from Twitter Data. In *Brazilian Symposium on Geoinformatics*. Campos do Jordão, Brazil, pp. 68–79, 2016.

Bogorny, V., Renso, C., Aquino, A. R., Lucca Siqueira, F., and Alvares, L. O. CONSTAnT: a conceptual data model for semantic trajectories of moving objects. *Transactions in GIS* 18 (1): 66–88, 2014.

Bourbia, A. L., Son, H., Shin, B., Kim, T., Lee, D., and Hyun, S. J. Temporal Dependency Rule Learning Based Group Activity Recognition in Smart Spaces. In *Compsac*. Atlanta, USA, pp. 658–663, 2016.

de Alencar, L. A., Alvares, L. O., Renso, C., Raffaetà, A., and Bogorny, V. A Rule-based Method for Discovering Trajectory Profiles. In *International Conference on Software Engineering & Knowledge Engineering*. Pittsburgh, USA, pp. 244–249, 2015.

de Aquino, A. R., Alvares, L. O., Renso, C., and Bogorny, V. Towards Semantic Trajectory Outlier Detection. In *Brazilian Symposium on Geoinformatics*. Campos do Jordão, Brazil, pp. 115–126, 2013.

Fileto, R., May, C., Renso, C., Pelekis, N., Klein, D., and Theodoridis, Y. The Baquara^2 Knowledge-based Framework for Semantic Enrichment and Analysis of Movement Data. *Data & Knowledge Engineering* 98 (1): 104–122, 2015.

Furletti, B., Cintia, P., Renso, C., and Spinsanti, L. Inferring Human Activities from GPS Tracks. In *SIGKDD International Workshop on Urban Computing*. New York, USA, pp. 5:1–5:8, 2013.

Furtado, A. S. and Bogorny, V. Efficient and Accurate Discovery of Stops in Trajectory Data. Tech. rep., Universidade Federal de Santa Catarina, Departamento de Informatica e Estatistica, Florianópolis, Brazil, 2017.

Furtado, A. S., Fileto, R., and Renso, C. Assessing the Attractiveness of Places with Movement Data. *Journal of Information and Data Management* 4 (2): 124–133, 2013.

Gordon, D., Scholz, M., and Beigl, M. Group Activity Recognition using Belief Propagation for Wearable Devices. In *International Symposium on Wearable Computers*. Seattle, USA, pp. 3–10, 2014.

Hirano, T. and Maekawa, T. A Hybrid Unsupervised/Supervised Model for Group Activity Recognition. In *International Symposium on Wearable Computers*. Zurich, Switzerland, pp. 21–24, 2013.

Kim, Y., Pereira, F. C., Zhao, F., Ghorpade, A., Zegras, P. C., and Ben-Akiva, M. Activity Recognition for a Smartphone Based Travel Survey Based on Cross-user History Data. In *International Conference on Pattern Recognition*. Stockholm, Sweden, pp. 432–437, 2014.

Liu, H., Luo, B., and Lee, D. Location Type Classification Using Tweet Content. In *ICMLA, 2012*. Boca Raton, USA, pp. 232–237, 2012.

Moreno, B., Times, V. C., Renso, C., and Bogorny, V. Looking Inside the Stops of Trajectories of Moving Objects. In *Brazilian Symposium on Geoinformatics*. Campos do Jordão, Brazil, pp. 9–20, 2010.

Njoo, G. S., Ruan, X. W., Hsu, K. W., and Peng, W. C. A Fusion-based Approach for User Activities Recognition on Smart Phones. In *Data Science and Advanced Analytics*. Paris, France, pp. 1–10, 2015.

Palma, A. T., Bogorny, V., Kuijpers, B., and Alvares, L. O. A Clustering-based Approach for Discovering Interesting Places in Trajectories. In *Symposium On Applied Computing*. Ceara, Brazil, pp. 863–868, 2008.

Reumers, S., Liu, F., Janssens, D., Cools, M., and Wets, G. Semantic Annotation of Global Positioning System Traces: activity type inference. *Journal of the Transportation Research Board* 2383 (1): 35–43, 2013.

Santos, A. A. *Towards Moving Objects Relationship Inference from Encounter Patterns*. M.S. thesis, Universidade Federal de Santa Catarina, Florianópolis, Brazil, 2016.

Santos, A. A., Furtado, A. A., Alvares, L. O., Pelekis, N., and Bogorny, V. Inferring Relationships from Trajectory Data. In *Brazilian Symposium on Geoinformatics*. Campos do Jordão, Brazil, pp. 68–79, 2015.

Shelley, K. J. Developing the American Time Use Survey Activity Classification System. *Monthly Lab. Rev.* 128 (1): 1–3, 2005.

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

Weerkamp, W., Rijke, M. d., et al. Activity Prediction: a twitter-based exploration. In *Time-aware Information Access Workshop*. Portland, USA, pp. 1–4, 2012.

Zhu, Z., Blanke, U., and Tröster, G. Pervasive and Mobile Computing. *Pervasive and Mobile Computing* 26 (1): 103 – 120, 2016.