<http://www.diva-portal.org>

Postprint

This is the accepted version of a paper published in *Journal of Location Based Services*. This paper has been peer-reviewed but does not include the final publisher proof-corrections or journal pagination.

Citation for the original published paper (version of record):

Prelipcean, A., Gidofalvi, G., Susilo, Y. (2014)

Mobility Collector.

Journal of Location Based Services, 8(4): 229-255

<http://dx.doi.org/10.1080/17489725.2014.973917>

Access to the published version may require subscription.

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-161529>

To appear in the *Journal of Location Based Services*
Vol. 00, No. 00, Month 20XX, 1–28

THIS IS A COPY OF THE PRE-PRINT MANUSCRIPT.

THE PUBLISHED VERSION:

Prelipcean, A.C., Gidófalvi, G. and Susilo, Y.O. (2014) Mobility Collector. *Journal of Location Based Services* ahead-of-print (2014): 1-27 (DOI:10.1080/17489725.2014.973917), found at http://www.tandfonline.com/doi/full/10.1080/17489725.2014.973917#.VG8S0HW9_0p

Preferred citation:

Prelipcean, A.C., Gidófalvi, G. and Susilo, Y.O. (2014) Mobility Collector. *Journal of Location Based Services* ahead-of-print (2014): 1-27 (DOI:10.1080/17489725.2014.973917)

To appear in the *Journal of Location Based Services*
Vol. 00, No. 00, Month 20XX, 2–28

Mobility Collector

Adrian C. Prelipcean^{a*}, Győző Gidófalvi^a and Yusak O. Susilo^b

^a*KTH Royal Institute of Technology, School of Architecture and Built Environment,
Department of Geoinformatics,*

^b*KTH Royal Institute of Technology, Centre for Transport Studies, Department of
Transport Science*

(Received 00 Month 20XX; final version received 00 Month 20XX; accepted 00 Month 20XX)

Despite the availability of mobile positioning technologies and scientists' interests in tracking, modeling, and predicting the movements of individuals and populations, these technologies are seldom efficiently used. The continuous changes in mobile positioning and other sensor technologies overburden scientists that are interested in data collection with the task of developing, implementing and testing tracking algorithms and their efficiency in terms of battery consumption. To this extent, this paper proposes an adaptive, battery conscious tracking algorithm that collects trajectory data fused with accelerometer data and presents *Mobility Collector*, which is a prototype platform that, using the tracking algorithm, can produce highly configurable, off-the-shelf, multi-user tracking systems suitable for research purposes. The applicability of the tracking system is tested within the transport science domain by collecting labeled movement traces and related motion data, i.e., accelerometer data and derived information (number of steps and other useful movement features based on temporal aggregates of the raw readings) to develop and evaluate a method that automatically classifies the transportation mode of users with a 90.8% prediction accuracy.

Keywords: movement; location tracking; data collection; location awareness; smartphones; battery conscious; location and accelerometer data fusion;

1. Introduction

Location-based technologies have radically changed the landscape of research, the way consumers use computing in their every day life, and the focus of applications. Consequently, the decrease of the desktop environment usage accompanied by the increase of the number of purchased mobile devices in the last several years (StatCounter 2014) shifted the main focus of commercial companies towards new concepts such as location-awareness or location-adaptive algorithms. These trends require a better understanding and usage of context in applications and / or services. To extract aspects of this context that refer to the mobility / movement of the user, effective tracking methods are needed.

The rigid, classical tracking option allows for either a high frequency data collection (with high battery consumption) or a low frequency data collection (with low battery consumption) without taking into consideration the dynamic nature of movement (i.e., wide variation in speed and movement direction of objects), neither of which fully meets the requirements of common tracking applications in transport science, Geographic Information Systems (GIS), or Location Based Services

---

*Corresponding author. Email: acpr@kth.se

(LBS). In a transport science environment, dense data is needed to be collected over longer periods of time (e.g., 12-17 hours). In a GIS environment, accurate and up-to-date locations are needed to provide users with fresh and contextualized information. Finally, in a LBS environment, the spatial and temporal aspect of data are relevant, but in some settings the temporal one might have precedence over the spatial one and vice versa. For example, a LBS that provides the bus schedule for predefined bus stations relies on the temporal aspect of data and an LBS service that alerts an user when he/she is close to a point of interest (e.g., has entered the perimeter of a geofence) relies on the spatial aspect of data. All these features are not achievable with the current tracking options, which slows down progress in both scientific and industrial environments.

One research discipline that can benefit from using an object's position together with acceleration traces - whose raw readings temporal aggregation can provide the number of steps and other useful movement features - is transportation science. There are several applications of positional data in the transportation science environment, such as: (1) studying the movement of groups of users within a city or a larger area to verify whether the spatial syntax of that place is relevant for the current transportation network status (Ratti 2004), (2) predicting movement of masses or individuals (Ashbrook and Starner 2002; Bachman, Borgelt and Gidófalvi 2013; Gidófalvi and Dong 2012; Gidófalvi et al. 2011), and (3) automatically inferring a user's transportation mode (Zheng et al. 2008). However, the applications depend on the amount of information that can be extracted from a data set and the information content is directly linked to the data collection process.

Data collection is essential to capture and understand individual and aggregated activity-travel patterns and behavioral mechanisms that underlie the decisions that can provide important information for various research and governmental agencies (e.g., planning, management, transport, etc.). The knowledge that is generated from data collection is used to estimate various urban and transport planning models for: (1) traffic forecasting (Herrera et al. 2010; Bar-Gera 2007), (2) transport and land use interactions (Verburg, Overmars and Witte 2004; Li and Yeh 2004; Xiao et al. 2006) and (3) the impact of changes of transportation infrastructure and policy to the stability and variability of people activity-travel patterns (Stopher et al. 2008; Li et al. 2004; Kawasaki and Axhausen 2009). In fact, the success of transportation policy implementation depends on an accurate description and prediction of aggregate flows as well as the disaggregate travel behavior of individuals. This requires suitable and reliable data collection tools to capture people's complex activity-travel engagements in both space and time.

The traditional data collection methods that capture these engagements use paper surveys that are error prone, time consuming, expensive and they burden the study participants. In transportation science, the traditional paper survey practice often results in a significant number of under reported short trips (which are mostly walks and non-routine trips), low response rates and paper-and-pencil method imposes a time lag / delay between the data collection process and the data entry. These shortcomings have been reported in the literature, e.g., Stopher, Bullock and Horst (2003), Wolf, Oliveira and Thompson (2003) and Asakura and Hato (2009).

Advances in mobile computing (i.e., new operating systems and programming frameworks), communication (i.e., the adoption of the third and fourth generation of mobile telecommunication standards and the increasing coverage of wireless networks), positioning (i.e., the ubiquity of network-based, GPS and WiFi positioning) and embedded sensors (i.e., accelerometer, gyroscope, magnetometer, etc.) can offer an alternative to the traditional data collection methods. Namely, the new technologies allow for the creation of mobile applications that, with the user's explicit

consent, can continuously collect location traces of the user and other information about the user's mobility. However, the successful usage of new technologies has been limited by: (1) the default tracking method provided by the open source mobile operating systems, which is not power efficient and does not automatically adjust to the movement dynamics of the user, (2) the absence of a generic, configurable, open-source trajectory collector and annotator, (3) the steep learning curve in mobile application development that is imposed by the continuously evolving technologies, and (4) the users' reluctance to share their private data.

To contribute to the uptake of tracking technologies in consensual mobility studies or in any other application that needs to legitimately track the movement / mobility of the user to provide value-added services to the user, the present paper focuses on the first three barriers. Namely, the paper proposes two new tracking methods, it presents an algorithm for battery conscious tracking, it defines two types of annotations, and proposes *Mobility Collector*, which is a framework that produces highly configurable, battery conscious tracking and annotating systems. The utility of the Mobility Collector platform is the focus of the case study, where Mobility Collector is used to generate a system that collects equally distributed positional data fused with accelerometer readings from multiple users. The data are used to train a classifier that automatically detects a user's transportation mode.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 provides necessary preliminaries and formalism. Section 4 presents the novel tracking options, the battery conscious algorithm and the annotating options. Section 5 presents the platform that is developed to provide a highly configurable tracking and annotating system. Finally, Section 6 presents the case study and Section 7 concludes and presents future work.

2. Related work

The related work is divided into two: Section 2.1 discusses research that has addressed the problem of battery conscious location tracking algorithms, and Section 2.2 discusses automatic mode detection, which is relevant for the case study that is presented in Section 6.

2.1 *Battery conscious location tracking*

Although previous approaches have considered adaptive, battery conscious location tracking, they have not utilized the optimization opportunities to their full extent. New approaches (Ben-Abdesslem, Phillips and Henderson 2009; Kim et al. 2010; Lee, Yoon and Han 2012) improve a device's battery life by using the GPS receiver together with an accelerometer. These approaches use the accelerometer to detect the motion of objects and duty cycle the location receivers for battery efficiency. Ben-Abdesslem, Phillips and Henderson (2009) use a location tracking method that switches between GPS and WiFi receivers to use the most optimal sensor in terms of battery consumption. Kim et al. (2010) and Lee, Yoon and Han (2012) use the WiFi receivers to obtain a location whenever it can not be obtained by a GPS receiver. Previous methods use static sampling parameters and a battery preserving algorithm that only affects the periods where no movement is present or when insufficient satellites are visible to obtain a GPS fix. Alternatively, researchers have also considered adaptive approaches. Namely, Zhuang, Kim and Singh (2010) consider a variation of adaptive sampling in case of low battery levels where the

location tracking starts collecting locations at a lower rate, but this approach is a last-resort option to lower the battery consumption. The tracking platform presented in this paper, Mobility Collector, uses the GPS receiver to obtain locations and the battery efficiency depends on the user's movement (i.e., it detects when the user is still and stops receiving locations) and on the user's dynamics (i.e., use the optimal tracking frequency considering the user's dynamically changing speed).

It is worth mentioning the generic points of interest (POIs) based tracking approach proposed by Ravindranath et al. (2012). In this case, a set of POIs are declared and the location listening service uses the lowest energy consuming sensor for approximate locations and switches to a more energy-intensive sensor when the POI is closer than the maximum possible error of the sensor. While this can provide great battery optimization for services that depend on a set of POIs, it is not suitable for location tracking, where the main purpose is to deliver precise geometry aspects (i.e., length, speed) for all parts / segments of trips. Compared to this approach, Mobility Collector does not take into account a set of POIs, it uses the GPS receiver only, and it is tailored for location tracking.

Funf (Aharony et al. 2011) is an open sensing framework that focuses on collecting, uploading and configuring data signals available via mobile phones. Although it provides useful information, it does so by individually collecting data from different sensors. Furthermore, its focus is on collecting various sensor readings for a general purpose and not on collecting data suitable for understanding an user's / object's mobility, whereas Mobility Collector focuses on collecting mobility data.

Jensen and Pakalnis (2007) present a new tracking system that guarantees accuracies at a low update and communication cost. To reduce the aforementioned costs, the client builds a model that predicts its future locations and sends it to the server. For the duration of the tracking frame, the client continuously receives locations and compares every location with the model's predicted location. When the predicted location differs from the received location by more than a tracking accuracy distance threshold, the client sends the actual location to the server, together with an updated prediction model. Mobility Collector uses a linear model similar to the vector-based tracking model that predicts the needed frequency to obtain equally spaced locations for varying speed levels. However, the Mobility Collector client adapts its sampling frequency according to the object's speed and it starts receiving locations at the predicted frequency, and not at fixed intervals.

2.2 *Transportation mode detection*

The process of automating the transportation mode detection of a particular user or a groups of users has shifted from classical approaches to modern ones but some shortcomings are still affecting the process. The classical approaches involved GPS receivers (Wolf, Guensler and Bachman 2001) or custom built devices such as a GPS receiver and an accelerometer but the deployment to a large set of users is limited due to the fact that such a device does not constitute an every-day item and users do not see an immediate / alternative benefit in their usage and hence often leave them behind on their travels. The modern approaches tend to use to existing devices that are already established as every-day items ,i.e., smartphones, and that have the potential for offering and analyzing potentially useful data. From the approaches that use smartphones and their embedded sensors, Zheng et al. (2008) and Stenneth et al. (2011) use the GPS receiver, Schüssler, Montini and Dobler (2011) and Reddy et al. (2010) use the GPS receiver together with the accelerometer. Reddy et al. (2010) also makes use of the network based position.

Although great results have been achieved in automatic mode detection, the studied approaches usually make use of external geospatial data (e.g., Stenneth et al. (2011) uses a transportation network and the real time position of the buses, Schüssler, Montini and Dobler (2011) use the location of public transportation station), or make use of post-processing and do not deliver the results in real-time (Zheng et al. 2008; Schüssler, Montini and Dobler 2011).

In general, it is almost always easier to differentiate between fewer classes, especially when the characteristics of the classes are very similar, as it is the case for positional motion characteristics of different motorized transportation modes. More often than not, approaches trade the potential utility of the classification (i.e., number of distinguished modes) for accuracy. Reddy et al. (2010) distinguishes between five classes with a precision of 93.6% (stationary, walking, running, bicycling and motorized), Schüssler, Montini and Dobler (2011) distinguishes six classes with a precision of 83% (walking, bike, car, urban public transport and rail) and Stenneth et al. (2011) distinguishes between five classes with a precision of 93.5% (stationary, walking, bicycling, bus, car and aboveground train).

The approach proposed in this paper offers a 90.8% classification precision for seven classes (walking, bicycling, bus, car, train, subway and ferry) by using information specific to a smartphone's sensors (i.e., accelerometer and GPS receiver) and the user's habits (i.e., the most common transportation mode of the user).

3. Preliminaries

This section provides the necessary preliminaries, formalism and terminology used in the remainder of the paper.

A moving object's position is obtained by a positioning service ,e.g., from a GPS receiver, usually at a very short period of time after the service is enabled. The time at which the location receiver is enabled depends either on a series of events (event-driven observations) or on an object's movement (location tracking).

In both cases, the output is a sequence of locations $L = \langle l_1, l_2, \dots, l_n \rangle$, where $l_i = (x, y, t)$. Let l_i denote the i th location in L , $l_i.x$ and $l_i.y$ represent its coordinates (planar or geographical) and $l_i.t$ denote the time at which it was recorded. The location sequence is defined as follows:

$$L = \langle l_i^{x,y,t} : l_{i+1}.t > l_i.t, \forall 0 < i < n \rangle \quad (1)$$

The sequence of locations has an information content / entropy (Chaitin 1982), which measures the amount of information contained by the sequence (the information content is zero when the sequence contains redundant information, i.e., a sequence that only contains duplicates of a location instance).

Because of storage limitations and / or data relevancy, it is common to use constraints when recording measurements and group them together, according to spatial or temporal criteria. The sequence of locations that are grouped together is a maximal contiguous subsequence L' of L such that every two consecutive locations in L' satisfy a temporal constraint (see Equation 2, where Δt_{min} represents the temporal constraint), or a spatial constraint (see Equation 3, where Δd_{min} represents the spatial constraint and $d[l_i.(x, y), l_j.(x, y)]$ represents the displacement between l_i and l_j) or any logical combination between the two constraints.

$$L'_t = \langle l_i^{x,y,t} : l_{i+1}.t * \text{foofoo} \times -l_i.t \geq \Delta t_{min}, \forall 0 < i < n \rangle \quad (2)$$

$$L'_d = \langle l_i^{x,y,t} : d[l_i.(x,y), l_{i+1}.(x,y)] \geq \Delta d_{min}, \forall 0 < i < n \rangle \quad (3)$$

Location tracking is a variation of sampling that aims at the maximization of the information content of a sequence of locations. It systematically obtains an object's location while maintaining a relationship between any two consecutive locations, $f(l_{i+1}, l_i) = \top$.

$$L^e = \langle l_i^{x,y,t} : f(l_{i+1}, l_i) = \top, \forall 0 < i < n \rangle \quad (4)$$

4. Battery efficient optimal mobility tracking

4.1 Location tracking

Location tracking extracts a subset of GPS-observable locations out of the infinite set of actual locations that describe an object's whereabouts at any given time within the tracking time frame. The most common type of tracking is user tracking, where a user carries a device (which is usually a smartphone) that can record the user's location. These devices run on different operate systems and have different hardware components. This paper focuses on devices running the Android mobile operating system because: (1) it is available for multiple types of devices (e.g., smartphones, smart-watches, etc.), (2) multiple phones manufacturers support Android, and (3) it is an open source operating system, with most of its component available for analysis and modification.

For services similar to location tracking, Android offers an application programming interface (API) of a location service that allows for periodic updates of a smartphone's geographical location.

4.1.1 Default Android tracking implementation

The default Android implementation¹ of a location tracking method depends on two main parameters, namely the minimum distance between two consecutive received locations, *min_Dist* and the minimum time between two consecutive received locations, *min_Time*. In the terminology used by this paper, *min_Time* corresponds to Δt_{min} , and *min_Dist* corresponds to Δd_{min} . In Android's implementation, the GPS receiver is enabled, it waits until it receives a location, it compares the distance from the current location to the previous location with Δd_{min} and it broadcasts the current location when the distance is larger, then waits for Δt_{min} milliseconds and reiterates this process until it is stopped. The sequence of locations, which is the result of the Android's default tracking implementation, is represented by the following equation:

$$L_e^{And} = \langle l_i^{x,y,t} : l_{i+1}^t - l_i^t \geq \Delta t_{min} \wedge d[l_i.(x,y), l_{i+1}.(x,y)] \geq \Delta d_{min}, \forall 0 < i < n \rangle \quad (5)$$

Setting up the Δt_{min} parameter to a value other than 0 and Δd_{min} to 0, the output of the tracking session is a subset of locations that are equally distributed in time - this type of tracking will be referred to as *equitime tracking* method. Similarly, setting Δt_{min} to 0 and Δd_{min} to a value other than 0, the output of the tracking session is a subset of locations that are equally spaced - this type of

---

¹The explanation for the Android implementation used in this paper is based on the authors' understanding of the explanation at Android's Official Page for Developers [http://developer.android.com/reference/android/location/LocationManager.html#requestLocationUpdates\(long,float,android.location.Criteria,android.app.PendingIntent\)](http://developer.android.com/reference/android/location/LocationManager.html#requestLocationUpdates(long,float,android.location.Criteria,android.app.PendingIntent)), as accessed on 29.03.2014.

tracking will be referred to as *equidistance tracking*. Both options track an object's movement - for the remainder of this paper an object synonymously refers to a smartphone that is carried by a user, unless it is explicitly stated otherwise.

4.1.2 Considerations on optimal tracking

Although the movement of an object simultaneously takes place in both the spatial and temporal dimensions and measurements along these dimensions are inherently related to one another through the velocity of the object, an equidistance sample is preferred over an equitime sample for two reasons. First, the equidistance sample has more utility in applications that make use of information from the spatial domain, for example 2D mapping applications that project the 2.5D (2D + time) sample onto the 2D space domain. Second, without assuming more than that the velocity of the object is changing dynamically, an equidistance sample maximizes the information content (entropy) of a fixed size sample for a fixed period and thereby equivalently maximizes the utility of the sample in a problem setting where the aim is to analyze the motion of the object primarily from a 2D / displacement perspective.

4.1.3 Equitime tracking

The equitime tracking method enables the GPS receiver, it waits until a location is sampled and then disables the receiver for Δt_{min} seconds, until it is manually stopped. The condition referred to in Equation 4 is represented by $l_{i+1}^t - l_i^t \geq \Delta t_{min} \wedge l_{i+1}^t - l_i^t < \Delta t_{min} + \epsilon_t$, where ϵ_t represents the non-deterministic time difference between the time at which the GPS receiver is enabled to obtain a location and the time at which the GPS receiver obtains a location. The equation that describes the sequence of sampled locations is:

$$L_t^e = \langle l_i^{x,y,t} : l_{i+1}^t - l_i^t \geq \Delta t_{min} \wedge l_{i+1}^t - l_i^t < \Delta t_{min} + \epsilon_t, \forall 0 < i < n \rangle \quad (6)$$

Compared to the sequence of locations presented in Equation 2, L_t' , which theoretically describes a sample form L that satisfies the temporal criterion according to its parameter Δt_{min} , the sequence of locations in Equation 6, L_t^e , defines a sample that is the outcome of a temporally constrained sampling process (tracking).

4.1.4 Equidistance tracking

Equidistance tracking is obtained in Android's implementation when Δt_{min} is set to 0 and Δd_{min} is set to the value which represents the desired distance between any two consecutive locations. However, if Δt_{min} is equal to 0, the GPS receiver is always enabled, which drains the battery life (this aspect is discussed in Section 4.2). This paper proposes a new approach to equidistance tracking that efficiently uses the smartphone's battery.

The proposed equidistance tracking algorithm dynamically adjusts the tracking frequency, Δt_{min} , according to an object's speed to obtain equally spaced locations every Δd_{min} meters. The condition between any two consecutive locations referred to in Equation 4 is represented by $d[l_{i+1}^{x,y}, l_i^{x,y}] \geq \Delta d_{min} \wedge d[l_{i+1}^{x,y}, l_i^{x,y}] < \Delta d_{min} + \epsilon_d$, where ϵ_d is the distance equivalent of ϵ_t , namely the distance the object traveled in ϵ_t , given its speed. The equation that describes the sequence of measurements obtained by an equidistance tracking session is:

$$L_d^e = \langle l_i^{x,y,t} : d[l_{i+1}^{x,y}, l_i^{x,y}] \geq \Delta d_{min} \wedge d[l_{i+1}^{x,y}, l_i^{x,y}] < \Delta d_{min} + \epsilon_d, \forall 0 < i < n \rangle \quad (7)$$

Compared to the sequence of locations presented in Equation 3, L_d' , which theo-

retically describes a sample form L that satisfies the spatial criterion according to its parameter Δd_{min} , the sequence of locations in Equation 7, L_d^e , defines a sample that is the outcome of a spatially constrained sampling process (tracking).

The proposed implementation uses an immediate history of previous locations to determine the maximum speed of the object and it re-registers the location receiver with a new frequency that is adequate to obtain locations spaced at Δd_{min} while considering the speed, as shown in Algorithm 1.

Data: LocationListener li , List<Location> L , Δd_{min}
Result: r_f

```

1  $l = li.getLocation();$ 
2 if  $length[L] = li.req_{size}$  then
3    $L \leftarrow L \setminus head(L);$ 
4    $L \leftarrow L \cup l;$ 
5    $speed \leftarrow sum(L.speed)/length[L];$ 
6    $p_f \leftarrow speed/\Delta d_{min};$ 
7   if  $|p_f - li.\Delta d_{min}| > min\_freq$  then
8      $li.\Delta d_{min} \leftarrow p_f;$ 
9      $li.req_{size} \leftarrow ct_{size}^{\Delta d_{min}};$ 
10  end
11 else
12    $L \leftarrow L \cup l;$ 
13 end

```

Algorithm 1: Method to adjust the frequency for equidistance tracking

Whenever a new location is broadcasted to the receiver, the algorithm presented in Algorithm 1 is called. There are 3 arguments for this algorithm: the location receiver, li (when it gets initialized, the values of $li.\Delta t_{min}$ and $li.req_{size}$, which represents the number of locations in the immediate history for the current value of Δd_{min} , are set to default values, i.e., $li.\Delta t_{min} = 30$ and $li.req_{size} = 3$), the location list L , which holds the recent history of the recorded locations and the equidistance sampling parameter, Δd_{min} . When li receives a new location, l , (Line 1) it is added to L . If L has a sufficient history of locations (Line 2), the algorithm first computes the average speed between consecutive locations stored in L (Line 5) and it estimates a frequency p_f that can be used by li to sample equally spaced locations. If the absolute value of the difference between the current frequency, $li.\Delta t_{min}$, and p_f is greater than a threshold value, min_freq , (Line 7) li updates its frequency to p_f (Line 8) and the required size value changes according to p_f (Line 9). This allows for a dynamic collection method which provides regularly spaced data and improves the battery life.

However, adapting $li.\Delta t_{min}$ to the object's current speed has one disadvantage, namely $li.\Delta t_{min}$ can, in theory, be set to ∞ when the history of immediate locations suggests no movement, thus permanently disabling location tracking. To avoid this scenario, one can either use a threshold value that represents the maximum value of $li.\Delta t_{min}$ or use a method to link li to a component which measures the user's motion (i.e., an accelerometer). The latter option is preferred and it is presented in Section 4.2.

4.1.5 A comparison between equitime and equidistance tracking

The equitime tracking option can be used for obtaining information regarding an object's mobility and it is common practice to set the receiver's frequency to high

Figure 1. Comparison between equidistance and equitime tracking. The figure on the left represents sampling output for fixed parameter settings ($\Delta t_{min}, \Delta d_{min}$) for various constant speeds. For equitime sampling, the region left to the vertical dashed lines is associated with oversampling and (questionable) battery effectiveness and the region under the horizontal dashed line is associated with undersampling and inferior information content. The figure on the right represents sampling output of different sampling methods in the case of dynamically changing speed.

values (i.e., obtaining a location every second) to maximize the information content. However, the equitime tracking method, implemented by setting the sampling frequency of the default tracking method high, oversamples, which implies that only a subset of the location sequence can be used to obtain information (i.e., obtaining an object's location every second when an object is stationary for long periods of time), as shown in Figure 1. While the information overload can be dealt with by using a spatial constraint, obtaining samples comes with a battery cost. Similarly, the equitime tracking method, implemented by setting the sampling frequency of the default tracking method low, undersamples, which implies that the method fails to collect all the samples that can be used to derive movement information.

The equidistance tracking option dynamically adjusts the tracking frequency and it avoids undersampling and oversampling, thus maximizing the information content without trading extra battery life for it. However, in order to use equidistance tracking efficiently, Δd_{min} has to be set according to the spatial scale of the studied phenomena. Considering an object that moves at a varying speed (the right part of Figure 1), an equitime tracking either fails to detect the fluctuations in speed (that occur in intervals smaller than Δt_{min}) or detects the fluctuations in speed with a high cost in battery life. The equidistance approach detects the fluctuations in speed (if Δd_{min} is set to an appropriate value that matches the spatial scale of the target phenomenon) while optimally utilizing the battery (low battery consumption at low speed levels).

4.2 Battery considerations

This section discusses the battery consumption of a smartphone when using different sensors at different frequencies and introduces an algorithm that balances the workload between different sensors to perform location tracking in a battery conscious manner.

Table 1. Battery consumption of sensors

Sensors	Discharge rate (mAh)
No sampling load (i.e., the cost of running the operating system)	17
Accelerometer – one read every 200 milliseconds	18
GPS – one read every 30 seconds	47
GPS – one read every 15 seconds	65
GPS – one read every second	84
GPS – no visible satellites (independent of frequency)	106

4.2.1 Battery consumption

The accelerometer is more efficient than the GPS receiver and its battery consumption² is negligible when it is compared to the cost of the device running without a sampling load, as shown in Table 1. As expected, the battery consumption of the GPS receiver increases directly proportional with the frequency at which it receives locations. However, when the GPS receiver is enabled but it cannot obtain a fix because there are not enough visible satellites³, it exhaustively searches for satellites until it obtains a fix, independent of the frequency at which the receiver is set. The GPS receiver's battery consumption when no satellites are visible results in the steepest decline of the device's battery level.

The equitime tracking option has a constant battery consumption when it is used (if enough satellites are visible), which is driven by the frequency at which it tries to obtain locations. However, the equidistance tracking option dynamically adjusts the frequency at which a GPS receiver tries to obtain locations and this causes lower battery consumption when an object moves at a low speed than when the object is moving at a (considerably) higher speed. If an equitime tracking option with a high frequency is used (i.e., once every second), its battery consumption will be higher than using an equidistance tracking option (in the worst case scenario, the object is always moving at a high speed, which corresponds to receiving a location per second, and the equidistance tracking option will consume as much battery as an equitime tracking option).

4.2.2 Battery saving

As discussed in Section 4.1.4, an equidistance tracking option can potentially permanently disable the GPS receiver if the object stops moving. To avoid this, a new method is suggested, which periodically listens to the low cost accelerometer and detects the motion component of movement that often co-occurs with the positional component. Klepeis, Tsang and Behar (1996) showed that people spend most of their time (approximately 90%) indoors, in which case there is no need to collect new locations, as they offer redundant information (i.e., the object is in the building). However, when leaving a building, the proposed algorithm detects motion (i.e., the user takes the smartphone and leaves the building, in which case the accelerometer detects that the user is moving) and can re-enable the GPS receiver to start listening for locations. Furthermore, the battery consumption indoors (i.e., when there are not enough satellites) is considerably more intense than in any other case (as presented in Section 4.2.1). A new method is suggested that, on one hand, wakes up the location listener when the accelerometer detects movement and, on the other hand, it minimizes the amount of time the GPS receiver is enabled when it can not obtain a location fix.

²The measurements for battery consumption were performed on an HTC One X+ smartphone and the device's screen was turned off for the duration of the measurements because its battery consumption dwarfs that of any sensor.

³At least four visible satellites are needed to obtain a GPS fix.

The proposed power saving algorithm links an alarm to the object / user's context in two different ways. This paper employs the terminology used in the Android's official developer's web page⁴ :

- the *positional component* of the user's context, which represents the displacement (Euclidean distance) between two consecutive locations of an object that can be detected by the GPS receiver (measured in a reference frame, such as the World Geodetic System, i.e., WGS84), and
- the *motion component* of the user's context, which represents the manner in which an object moves that can be detected by an accelerometer (measured in an inertial three-axis reference system relative to the device).

The power saving algorithm has two segments: a position segment, which obtains locations from the GPS receiver (high battery consumption) and analyzes the positional component of the object's movement, and a motion segment, which obtains movement traces from the accelerometer sensor (negligible battery consumption) and analyzes the motion component of the object's movement. The pseudocode for the suggested approach for an automatic algorithm that is used for improving battery life is presented in Algorithm 2.

First, on Line 1 the algorithm enters its spatial segment and a location listener, li_{loc} , is registered (Line 2). li_{loc} tries to receive locations at a t_{loc} interval until it gets unregistered. An alarm is then scheduled for a time, t_p , which is the sum between the current time and a time constant, t_s (Line 3). As long as the location listener receives locations, the alarm is canceled and rescheduled for t_p (Lines 4-10). If li_{loc} does not receive any locations until t_p , the algorithm unregisters li_{loc} (Line 11), which causes the algorithm to enter its motion segment.

Then, a loop that collects information regarding the object's motion every t_m seconds is launched (Lines 13-31). An accelerometer listener li_{acc} , which registers accelerometer reads at a t_{acc} interval, gets registered and an alarm, is scheduled for a time t_p , which is the sum between the current time and a time constant, t_m (Line 15). An empty list, L_a , is used to collect the accelerometer values recorded in the t_m interval. After t_p , li_{acc} is unregistered and the average of the accelerometer values stored in L_a is compared to a threshold value, min_acc . If the average is larger than min_acc , the accelerometer reads suggest movement and this causes the algorithm to re-enter its positional segment (Line 25 and Line 2). If the average is smaller than min_acc , the algorithm waits for t_m seconds (Line 27 and Line 28) and reiterates through the motion segment (Line 17).

---

⁴The terminology is derived from http://developer.android.com/guide/topics/sensors/sensors_overview.html, as accessed on 29.03.2014

```

Data: Service  $s$ , Listener<Location>  $li_{loc}$ , Listener<Accelerometer>  $li_{acc}$ ,
List<Accelerometer Values>  $L_a$ ,  $t_s$ ,  $t_m$ ,  $t_{acc}$ ,  $t_{loc}$ 
1 while  $isRunning(s)$  do
    /* positional segment of the battery saving alarm */
2    register( $li_{loc}$ );
3     $t_p = now() + t_s$ ;
4    while  $now() > t_p$  do
        /* positional segment of the alarm gets cancelled and
           rescheduled as long as new locations are broadcasted */
5         $li_{loc}.requestLocation()$ ;
6        if  $locationReceived(li_{loc})$  then
7             $t_p = now() + t_s$ ;
8            wait( $t_{loc}$ );
9        end
10    end
    /* left the positional segment of the alarm */
11    unregister( $li_{loc}$ );
12     $isNotMoving \leftarrow true$ ;
13    while  $isNotMoving$  do
        /* motion segment of the battery saving alarm */
14        register( $li_{acc}$ );
15         $t_p = now() + t_m$ ;
16         $L_a \leftarrow L_a.removeAll()$ ;
17        while  $now() > t_p$  do
18             $acc = li_{acc}.getAccelerometerReads()$ ;
19             $L_a \leftarrow L_a \cup acc$ ;
20            wait( $now() + t_{acc}$ );
21        end
22        unregister( $li_a$ );
23         $isMoving \leftarrow (avg(L_a) > min\_acc)$ ;
24        if  $isMoving$  then
25             $isNotMoving \leftarrow false$ ;
26        else
27             $t_p = now() + t_m$ ;
28            wait( $t_p$ );
29        end
30    end
31 end

```

Algorithm 2: Method to duty cycle the battery consumption of a device

4.2.3 Effects on battery life

Consider, for simplicity, that the battery consumption due to a tracking service is limited only to the effect of the GPS receiver on the battery life (thus ignoring any other external factors such as screen and GSM induced battery consumption). The battery consumption C can be described by Equation 8, where $t_{GPS}^{f=a}$ represents the amount of time the GPS receiver was enabled and received locations (i.e., enough satellites were visible) at a fixed frequency $f=a$, $d_{GPS}^{f=a}$ represents the discharge rate that is associated with receiving locations at the frequency $f=a$, $t_{GPS}^{f=\infty}$ represents the amount of time the GPS spent failing to obtain the object's location (due to insufficient visible satellites) and $d_{GPS}^{f=\infty}$ represents the discharge rate that is

associated with these failures:

$$C = t_{GPS}^{f=a} * d_{GPS}^{f=a} + t_{GPS}^{f=\infty} * d_{GPS}^{f=\infty}. \quad (8)$$

The purpose of using Algorithm 2 is to minimize the amount of time the GPS is enabled to avoid trying to obtain a location fix when it is not possible to obtain it. This operation is done by leveraging $t_{GPS}^{f=\infty}$ and $t_{GPS}^{f=a}$ with t_{acc} , as shown in Equation 9, where t_{acc} represents the amount of time the GPS receiver is disabled because it cannot obtain a fix and is replaced by the accelerometer and d_{acc} represents the battery discharge rate that is associated with using the accelerometer:

$$C = t_{GPS}^{f=a} * d_{GPS}^{f=a} + (t_{GPS}^{f=\infty} - t_{acc}) * d_{GPS}^{f=\infty} + t_{acc} * d_{acc}. \quad (9)$$

Furthermore, the equidistance tracking affects - Algorithm 1 - the battery life because it dynamically adjusts the frequency at which the GPS receiver tries to acquire location fixes, which results in Equation 10, where F is a set that contains the frequencies that were predicted by the equidistance algorithm:

$$C = \sum_{f_i \in F} t_{GPS}^{f_i} * d_{GPS}^{f_i} + t_{GPS}^{f=\infty} * d_{GPS}^{f=\infty}. \quad (10)$$

Combining the two strategies presented in Equation 9 and 10, the effect on battery life is given by Equation 11:

$$C = \sum_{f_i \in F} t_{GPS}^{f_i} * d_{GPS}^{f_i} + (t_{GPS}^{f=\infty} - t_{acc}) * d_{GPS}^{f=\infty} + t_{acc} * d_{acc}. \quad (11)$$

Because of the complexity and dynamic nature of an individual's movement, it is only possible to make conclusive remarks about the savings when one empirically tests with a set of users, each carrying for several days three identical copies of a smartphone at all times that run the two components of the battery saving algorithm and the battery saving algorithm as a whole (given by Algorithm 2, Algorithm 1, and both algorithms combined, respectively), and measures the battery consumption of each per user. One could calculate the average savings of the strategies over all the smartphones carried by each of the users during the extended duration of the test period. While this analysis is beyond the scope of this paper, several remarks can be made referring to the effectivity of each algorithm. First, Algorithm 1 provides savings when the users speed during movement is not constant. Second, Algorithm 2 provides savings when the user is stationary (which often happens to correlate with the user being in indoor environments). Finally, empirical measurements conducted with a single user during the course of an average day suggest that the battery savings effectivity of Algorithm 2 is higher than Algorithm 1, primarily because it is applicable for a longer period of time (as illustrated in Figure 2). However as Algorithm 1 and 2 can naturally be combined to implement the disjunction of the two strategies and since the applicability periods of Algorithm 1 and 2 are nearly mutually exclusive, the combined algorithm results in an even higher applicability and hence battery savings efficiency.

Figure 2. Comparison between battery consumption strategies. The image presents 30% remaining battery percentage after “Mobility Collector” has been running for 13.5 hours, during which the GPS was enabled for approximately 2.5 hours. The steepness of the slope that describes the battery discharge rate varies according to the frequency at which the GPS receiver acquires locations. The period of time when Algorithm 1 was active is drawn with red. The discharge rate within the rectangle to the left is due to equidistance tracking for a period where the most time was spent traveling at walking speed (one location per thirty seconds, with a discharge rate of 5% of battery capacity every hour) and the discharge rate within the rectangle to the right is due to equidistance tracking for a period where the most time was spent traveling at an average bus speed (one location per five seconds with a discharge rate of 13% of battery capacity every hour). The stationary periods identified by Algorithm 2 are drawn in blue and correspond with the gradual battery discharge slope.

4.3 Annotating system

For continuous variables, including variables with interval and ratio scales, the variables, the linking, and the calculation of the linked variables are often performed automatically. For discrete / finite variables, namely variables with nominal and ordinal scales, the linking can be performed manually⁵. The manual linking is hereby referred to as annotation and this paper proposes a new system suitable for manually annotating data via an easy to use interface. The system records the annotated data together with either the time of annotation (temporal annotation) or with the location at which the data was annotated (spatial annotation).

Consider Figure 3, where A_i represent the annotations, T_i represent the items annotated by temporal annotations and S_i represent the items annotated by spatial annotations. The temporal annotation is inserted immediately (T_1 is annotated at t_1^A and it is relevant until T_2 is annotated at t_2^A) and the spatial annotation is inserted when it can be associated with a location (A_1 is recorded at t_1^A but S_1 is annotated at t_1^L and it is relevant until S_2 is annotated at t_3^L). In the spatial annotation case, S_1 corresponds to L_1 and L_2 , and S_2 corresponds to L_3 . However, if multiple annotations are recorded in between two consecutive locations, only the

⁵Linking of discrete / finite variables can also be done automatically by converting continuous variables via thresholding.

Figure 3. An annotating system for spatial and temporal annotations. Elements denoted with T_i represent data annotated by temporal annotations and the elements denoted by S_i represent data annotated by spatial annotations. The arrows are specific to spatial annotations, the direction of the filled arrows denotes the locations associated with the annotation the arrows starts from and the direction of the empty arrows denotes the location at which one annotation's period stops and the consecutive annotation's period starts. If one arrow is directed from an annotation to another annotation, the first annotation is ignored.

most recent annotation is inserted⁶ in the database as a spatial annotation (A_3 is performed before A_4 , but it is not associated with a location and it is not inserted in the database) and S_4 , which corresponds to L_4 , is annotated as A_4 and A_3 is missed. If spatial annotations are used, every location has an annotation.

5. Mobility Collector

Mobility Collector is a highly configurable, open source, battery conscious, mobile tracking and annotating framework that is specifically designed for research purposes. The framework generates a mobile application that collects data and a database that centralizes the collected data.

5.1 Sensor Fusion

The battery saving algorithm presented in Section 4.2.2 uses the accelerometer to detect movement in a time window by aggregating the accelerometer reads. If this information is fused with the sequence of locations, it increases the information content of the sequence. The process of combining data derived from different sensors is defined as sensor fusion. Developments towards a framework for sensor fusion have been made (Llinas and Hall 1998) and it has proven to be useful in different research areas such as intelligence analysis (Jenkins et al. 2011) or indoor localization (Martin et al. 2010).

Mobility Collector is a framework that can be configured to collect trajectory (a location measurement sequence) and categorical (annotations) data suitable for different research purposes. The framework focuses on offering a system that can capture the mobility / movement of objects. This is done by fusing together trajectories with accelerometer data. However, the accelerometer measures acceleration values at a very high frequency (usually, once every 200 milliseconds) and this makes it difficult to store every accelerometer read in a database. To overcome

⁶This paper focuses on inserting one spatial annotation per location to offer spatially discrete data that can be used for training real-time location-based classifiers, but an implementation in which multiple spatial annotations correspond to a location is possible.

Figure 4. System Architecture

this, Mobility Collector uses a list to temporarily store all the accelerometer reads between two consecutive locations, l_{i-1} and l_i . When l_i is received, the accelerometer values in the list are aggregated and the following features are inferred: (1) the mean, minimum, maximum and the standard deviation values of every accelerometer read per axis, (2) the number of steps, which is computed by an algorithm that is a variant of the approach proposed by Mladenov and Mock (2009), and (3) whether the object is moving or stationary. The summary is fused with l_i and inserted in a database. Mobility Collector offers the option to collect either fused accelerometer data with trajectories or collect stand-alone trajectories.

5.2 System Architecture

Mobility Collector can be configured either as a client-server architecture or as a stand alone client. The stand alone client case is useful in any mobile application that does not intend to share the movement related data with a central server but intends to use it to provide useful functionality to the user based on the locally collected or incrementally processed movement information. The client-server architecture can be used to seamlessly collect and centralize data from multiple clients to a database server. The out-of-the-box implementation of the client-server architecture is that of a thin client, where the client is used to collect data without performing any advanced operations (e.g., detecting the frequent places of the user). However, if an administrator decides that the client should perform advanced operations, the source code can be extended programmatic. In this case, a thick client architecture will provide the needed functionality. Both thick and thin clients are presented in Jing, Helal and Elmagarmid (1999).

The Mobility Collector using a client-server architecture is presented in Figure 4. The client is an Android mobile device that is equipped with a GPS receiver, an accelerometer, permanent data storage and it is able to process data and communicate to a remote server. A user interacts with the Android device for authentication tasks (i.e., logging in, registering), to toggle the Mobility Collector on / off or to annotate the data (if annotating is enabled). A server contains a Java servlet which facilitates the interaction between multiple clients and the server's database. All

Figure 5 shows the Mobility Collector web form, divided into two parts. Part (a) is the 'First Part' and contains an 'About section' with a text area (highlighted with a green box and a green checkmark), 'System Architecture' (Client-Server), 'Sampling type' (Equidistance), 'Sampling time(seconds)' (30), 'Sampling distance(meters)' (50), 'Accuracy constraint' (0), and 'Accelerometer' (Yes). Part (b) is the 'Second Part' and contains 'Servlet URL' (http://kth.se), 'Trajectory annotation' (Point-based), 'Enable Power-Saving' (Yes), 'Auto Upload' (True), 'Upload Frequency' (1), 'Annotation Items' (a list: Walk, Bicycle, Car, Bus, Subway, Train, Ferry), and a 'Submit Form' button.

Figure 5. Mobility Collector web form used for the case study

the collected data are sent to the server in numerical and/or string format and the server's database does not need to be spatially enabled.

5.3 Mobility Collector Generator

The Mobility Collector generator is a web page where researchers can fill in a form (see Figure 5) that generates their desired configuration. The form has two different sections, one for details regarding the researcher that intends to use the generator and one for details regarding the configuration of the system, which is shown in Figure 5(a) and 5(b).

When using the Mobility Collector generator, a researcher has the option to specify the type of tracking (equitime or equidistance), their respective parameters (Δt_{min} or Δd_{min}) and the architecture of the system (client only or client-server). Furthermore, the options of fusing locations with accelerometer summaries, using the battery saving algorithm, automatically upload the data to a server are present. Finally, Mobility Collector supports the specification of an annotating system and the type (spatial or temporal) and schema (i.e., annotation labels) of the annotations, and an "About section" for the mobile application can be introduced.

The Mobility Collector then generates a mobile application for Android devices (in the form of an "apk" file) and emails it to the researcher. For a client-server architecture, the email also contains a link to the source code that is hosted on Github⁷, which is a web-based hosting service, along with the instructions on how to configure the server. The current implementation delivers a script that can be used on OpenShift⁸ to automatically deploy and configure the servlet and the database (with its scheme). If used on a personal server, the servlet deployment is not automatic and the database connection has to be configured manually.

⁷Github can be accessed at <https://github.com/>.

⁸OpenShift can be accessed at <https://www.openshift.com/>.

Figure 6. Annotation menu for Mobility Collector where the selected annotation is walking.

Figure 7. Number of instances per class

6. Case study and results

6.1 Case study setup

The purpose of the case study is to illustrate the effectiveness and ease of use of the Mobility Collector platform. Mobility Collector provides (trajectories fused with accelerometer) data that is used to train classifiers to automatically detect the mode of transportation of the user.

To achieve this, a Mobility Collector was configured by choosing a client-server architecture, an equidistance tracking option with sampling parameters ($\Delta t_{min} = 30s$, which is the initial sampling rate, and $\Delta d_{min} = 50m$, which is the equidistance tracking parameter) with a suitable annotation scheme (*Walk*, *Bicycle*, *Car*, *Bus*, *Subway*, *Train* and *Ferry*) for transport mode collection. The completed form is shown in Figure 5 and the generated mobile application is shown in Figure 6(a). To achieve a more appealing and usable user interface, the source of the generated mobile application has been slightly modified to include buttons with icons that correspond with the transport modes (Figure 6(b)). For a period of 10 days, 11 users have used the Mobility Collector to annotate their daily movement. The resulting data set contains 14203 data instances (Figure 7 shows the frequency distribution of the modes of the collected locations).

Participants annotated most of the data as either car or train because longer distances are covered by these transportation means. The participants did not travel long distances by bicycle because either bicycling was not a preferred transportation mean in the selected group or the weather⁹ prevented them from using bicycles. Although long distances are traveled by subway, the GPS lacks signal underground and can not receive locations, thus leading to few instances annotated as subway.

6.2 Automatic transportation mode classification

The collected data, which consists of positional and motion features, is used by a window based processing approach to calculate statistical aggregates of a sequence of previous measurements. The processed data is used to derive a vector for each location measurement and a set of different classifiers were trained on the labeled data set using a K-fold cross validation (Kohavi 1995) approach to obtain training samples.

6.2.1 Features used by the classification

Contrary to other studies that also use data that is not specific to a smartphone's sensors (Stenneth et al. 2011; Schüssler, Montini and Dobler 2011), the herein proposed classifiers use data that are obtained only from a smartphone's sensors and that are accessible to the smartphone in real time without needing to contact a server for additional information.

The features derived from the collected data are presented in Table 2, where $hist\_summary(feature, n)$ represents the average, minimum and maximum values of a feature for a window that contains the current and the preceding $n-1$ measurements (the current case study uses $n = 5$). The *distCheck* is a boolean indicator variable that is set to true when the distance to the previous measured location is larger than the sampling distance, *min_Dist*, which is a helpful feature to detect changes in transportation modes. The *numSteps* records the number of steps that is computed from the accelerometer values recorded in between consecutive locations measurements and is useful for differentiation between walking and bicycling.

To distinguish between different user patterns, the *userId*, which represents a unique identifier for every participant, is used in the classification. This increases the accuracy of the classification by 6.2% because some participants exclusively and systematically use some and not other transportation modes, e.g., only one user used a ferry and only one user used a bicycle. While this is a substantial and significant increase in accuracy for the given data set, using this feature restricts using the learned classifiers to detect the transportation mode of new users. However, for a large user dataset, with a large data set to train the classifiers on, using *userId* would make the accuracy gain for using this feature negligible.

6.2.2 Tested classifiers

The WEKA (Hall et al. 2009) machine learning software suite is used in the case study for the following reasons. First, it offers multiple types of classifiers. Second, it is a widely used tool by the scientific community. Finally, WEKA is open source and the classifiers are implemented as extensions written in Java, whose source code as a whole or in part can easily be integrated into Mobility Collector. The classifiers used in this case study are: (1) Bayesian Network (**BN**), (2) Support

---

⁹The case study was performed in November, in Stockholm, Sweden.

Table 2. List of features

Parameter name	Information
speed	the current speed of the user
distCheck	a boolean indicator of whether $dist(l_i, l_{i-1}) > \Delta d$
hist_summary(speed,4)	statistical summary of speed
accMax	the maximum accelerometer value
accMean	the average of the accelerometer values
accStdDev	the standard deviation of the accelerometer values
numSteps	the number of steps
hist_summary(accStdDev,4)	statistical summary accStdDev
hist_summary(numSteps,4)	statistical summary of numSteps
userId	a unique identifier for the user

Table 3. Classification precision and recall results per class and the weighted average of the overall precision and recall

Mode	BN		SVM		DT		RT		RF	
	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
Car	84.7%	78.5%	90.6%	38.1%	80.7%	78.2%	84.6%	84.9%	91.6%	87.2%
Train	82.3%	82.8%	60.2%	97.0%	79.8%	86.5%	84.7%	84.3%	87.8%	92.1%
Walk	92.9%	91.0%	89.9%	96.3%	92.1%	92.1%	93.3%	93.0%	93.0%	97.3%
Subway	10.1%	33.7%	33.3%	2.2%	22.6%	7.9%	35.8%	38.2%	59.5%	24.7%
Bus	69.6%	84.1%	90.3%	91.1%	89.1%	76.8%	87.6%	88.2%	92.8%	92.0%
Bike	51.2%	55.9%	0.0%	0.0%	84.3%	36.4%	59.8%	56.8%	86.5%	54.2%
Ferry	99.3%	99.3%	94.4%	99.2%	97.5%	95.5%	98.1%	98.8%	99.7%	99.3%
Avg.	84.4%	83.1%	79.2%	74.0%	83.4%	83.5%	86.5%	86.5%	90.8%	90.9%

Vector Machine (**SVM**), (3) Decision Tree (**DT**), (4) Random Tree (**RT**) and (5) Random Forest (**RF**).

6.2.3 Automatic mode detection results and discussion

The classifiers are trained and tested on different datasets and specific performance measures are computed. A K-fold ($K = 10$) cross validation partitions the data sample into K equal sized subsamples, the classifiers retains one subsample as a validation set and it uses the other $K - 1$ as a training set. This process is repeated K times until each of the subsamples is used as a validation set. After training each classifier, a summary provides its *precision*, which represents the percentage of correct positive predictions, and its *recall*, which represents the percentage of positive cases the classifier made (Ting 2010). In Table 3, the precision and recall of the classifiers are presented.

6.2.3.1 Inter-class analysis of classification errors.

All classifiers except SVM, which has a precision of 79.2%, achieve a precision of over 80%. RF, with a 90.8% precision and a 90.9% recall, outperforms the rest of the classifiers. For RF, the classes have the precision under 90% either because they have a low recall value (e.g. subway, due to the inability to receive GPS signal underground, and bike, due to the low usage of this transportation mean when it is compared to the others) or because they are difficult to distinguish since they exhibit similar characteristics (i.e., when traveling by car or train, the smartphones record similar accelerometer patterns). Table 4 contains the confusion matrix of the RF classifier.

RF confuses car with train because they have common characteristics (i.e., high speed, low values for the accelerometer reads or the users that travel by car also travel by train). RF misclassified 7% of the instances of walking as other transportation modes because of the training set and the data collection method. Every location has an annotation and when the users annotate a change (e.g., from car

Table 4. Confusion Matrix for RF

Class	Car ^p	Train ^p	Walk ^p	Subway ^p	Bus ^p	Bike ^p	Ferry ^p
Car ^a	4480	611	40	2	4	0	0
Train ^a	369	4464	10	0	1	1	0
Walk ^a	30	5	2408	3	19	7	2
Subway ^a	2	0	32	22	31	2	0
Bus ^a	5	1	49	9	735	0	0
Bike ^a	1	1	49	1	2	64	0
Ferry ^a	4	0	1	0	0	0	736

^a Actual Class^p Predicted Class

Figure 8. Spatial distribution of the correctly classified instances (a) and classification errors (b).

to walk) in between two consecutive locations, l_i and l_{i+1} , the new transportation mode will be associated to l_{i+1} . This affects the classification because l_{i+1} contains the summaries of the accelerometer values that are not specific to either the first mode (previous annotation) or the second mode (new annotation). Furthermore, although the classifier accurately distinguishes bus from car, the number of data instances annotated as bus (799) is small when compared to those annotated as car (5137). Also, the data set has to contain more instances annotated as classes that have a low precision and recall (i.e., subway and bicycle) to improve the classifier's prediction accuracy. The spatial distribution of the correctly classified instances by the **RF** classifier is illustrated in Figure 8(a) and the spatial distribution of the classification errors is illustrated in Figure 8(b).

To the authors' knowledge, this is the highest transportation mode classification accuracy that uses smartphone-specific data only (i.e., locations and accelerometer values) and considers 7 transportation modes. This is of particular interest in LBS because the classifier can be deployed to smartphones, which can use it to detect the transportation mode and to provide fresh and relevant information to the user in real time.

Figure 9. An overview of the sequentially singular (a) and regular (b) classification errors produced by the **RF** classifier. Areas such as those within the blue rectangle contain mostly sequentially singular classification errors, whereas areas similar to the one denoted by the red rectangle contain mostly sequentially regular classification errors.

6.2.3.2 Sequential analysis of classification errors.

Studying the sequential context of the classification errors can offer more insights on whether the accuracy of the presented classifiers can be improved and can identify by which means it can be improved. To this extent, this paper identifies two types of errors that can be identified by analyzing the number of classification errors within a sliding window, namely:

- *Sequentially singular classification errors*, which are singular misclassification instances within the sliding window and can be eliminated by a low-pass filter, and
- *Sequentially regular classification errors*, which are systematic errors that form the majority of instances within the sliding window and therefore cannot be eliminated by a low-pass filter.

The sequentially regular and singular classification errors identified when using a sliding window which contains five classification instances (the current instance, the previous two instances, and the following two instances) are illustrated in Figure 9. Out of the identified errors, approximately 39.5% were identified as sequentially singular classification errors, which can be eliminated by a low-pass filter, and 61.5% were identified as sequentially regular classification error. The identified percentages of the sequential analysis depend on the length of the sliding window but the study of the associated precision and recall trade off of a model that uses this low-pass filter is not part of the current paper and is planned to be part of future research.

6.2.3.3 Geocontextual analysis of classification errors.

While 39.5% of the errors could be eliminated by a low-pass filter, analyzing the geographical context of the misclassified instances can offer further insight on what type of areas are associated with sequentially regular classification errors. The analysis can include studying the proximity of the sequentially regular classification errors to transportation stations, tunnels, railways, intersection, traffic lights, stop signs, transportation routes, etc. This paper considers the proximity to trans-

Figure 10. Spatially discretized relative misclassification density (a) compared to spatially discretized average GPS accuracy (b).

portation stations and the proximity to transportation routes for the geocontextual analysis of the classification errors.

From the identified sequentially regular classification errors, 16.8% are within a 50 meters buffer of a transportation station, and 31.2% are within a 100 meters buffer of a transportation station. Similarly, 82.1% of the sequentially regular classification errors are within a buffer of 50 meters of a transportation route, and 86.3% of the sequentially regular classification errors were within an 100 meters buffer of a transportation route. The transportation stations were all found to be within 50 meters of a transportation line and the proximity analysis to either transportation lines *OR* transportation stations did not provide any meaningful result. Considering the aforementioned results, embedding the geocontextual information to a classifier and asses the accuracy gain (or loss) of this additional information is planed to be part of future research.

Another factor that could impact the misclassification rate is the level of built up environment, which affects the GPS receiver's localization accuracy. To study this, the Stockholm area has been discretized by using a rectangular grid, where each grid cell is a square with its side equal to 500 meters. For each cell, the average accuracy of recorded locations was computed and the result is shown in Figure 10(a). The GPS receiver seems to acquire locations more inaccurately within central Stockholm and along the railway. However, in the southern region, even if the area is densely built up, the GPS receiver accurately acquires locations. While this information offers an overview of the localization accuracy, it does not directly explain the misclassification error.

To analyze the misclassification error, a new measure m_e is introduced that describes the relative misclassification density per grid cell, which is defined by Equation 12, where n_m represents the number of misclassified instances within a grid cell and n_t represents the total number of instances within the same grid cell:

$$m_e = n_m / n_t. \quad (12)$$

The relative misclassification density for the study area is illustrated in Figure 10(b). There is a higher misclassification density along the railways and highways than in the highly built up areas. This is mainly due to the confusion between car

and train, due to similar features (i.e., high speed values and low values for the accelerometer measurements). The grid cells with a higher relative misclassification density than their neighboring cells coincide with the places where the users transition between different traveling modes, which is expected because the locations where transition occurs are not guaranteed to be recorded at the considered spatial scale. Finally, there seems to be no direct link between the accuracy of the GPS readings and the relative misclassification density but proving this statement is planned to be part of future research.

7. Conclusions and future work

This paper introduces two new tracking options (equitime and equidistance tracking), advocates the use of the optimal equidistance tracking, proposes a new approach to battery efficiency and introduces Mobility Collector, which is a new research-oriented tracking platform that can produce highly configurable tracking systems. The case study illustrates how Mobility Collector can be used by underlying the necessary steps to achieve a system that can detect the transportation mode of users with a precision of 90.8% for seven considered transportation means: walking, bicycling, bus, car, train, subway and ferry.

The planned research direction that will be providing improvements to the herein proposed travel mode detection method is based on the insights that were gained in the sequential and geocontextual analysis of the method's misclassification errors. Similarly, further work could be focused on distinguishing between the car driver and car passengers by using data that is collected and / or derived from smartphones. However, in this case, the positional component of the user's context coincides for both passengers and the driver and the motion component might not be sufficient to discriminate between the two entities. This problem might be solvable by using additional data sources, e.g., (1) detecting the type of interaction with the smartphone and measuring the amount of interaction (continuous interaction when it is used as a navigation tool by the driver or not at all, because the smartphone could be idling vs. intermittently used by passengers who are not required to pay attention to the road), or (2) identifying the car used by the traveler and taking into account the continuity of car trips (the traveler that uses a particular car most often is, most of the time, the driver of the car).

The case study did not consider "stationarity" as a transportation mode. Even though stoppage and transition periods are important in the transportation science, this case study focuses on inferring the transportation mode of a user in real time and does not consider "stationarity" periods, which can be identified after they occur. Namely, due to the equidistance distribution of the collected data, "stationarity" periods can be identified as unexpectedly long temporal gaps between two consecutive recorded locations¹⁰.

The Mobility Collector framework is an open-source project and developers can contribute to it as they see fit. Some of the aspects the authors wish to further explore are: (1) extend the data collection to cover all the smartphone's sensors (e.g., magnetometer, thermometer, etc.), (2) develop more efficient battery saving methods, (3) use an optimization algorithm for finding the trade-off between power consumption and data frequency, as opposed to the currently used trial-and-error

---

¹⁰The equitime tracking approach can provide data suitable for the detection of stationarity periods in real time by identifying when the consecutively recorded locations are within a threshold distance of a central location.

values, (4) improve the user interface by allowing the use of icons as annotation items, (5) assess how multi-scale sampling (spatial and temporal) affects the data quality, (6) offer the annotating system as a stand alone option, (7) extend Mobility Collector so that it includes the optimal collection and selective fusion of measurements from multiple, complementary positioning technologies (WiFi, network-based, etc), and (8) integrate Mobility Collector with existing open-source services and / or projects (e.g., OpenShift) to reduce deployment issues and make it accessible to a larger audience.

Acknowledgments

This work was partly supported by the Trafikverket (Swedish Transport Administration) under Grant "TRV 2014/10422"; KTH Transport Research Environment with Novel Perspectives (TRENoP) Strategic Research Area; and KTH innovative Transport Research Undergraduate Experience (iTRUE).

References

- Aharony, Nadav, Wei Pan, Cory Ip, Inas Khayal, and Alex Pentland. 2011. "Social fMRI: Investigating and shaping social mechanisms in the real world." *Pervasive and Mobile Computing* 7, no. 6: 643–659.
- Asakura, Yasuo, and Eiji Hato. 2009. "Tracking individual travel behaviour using mobile phones: recent technological development." In *The Expanding Sphere of Travel Behaviour Research. Selected Papers from the 11th International Conference on Travel Behaviour Research. International Association for Travel Behaviour Research*: 207–233.
- Ashbrook, Daniel, and Thad Starner. 2002. "Learning significant locations and predicting user movement with GPS." In *Wearable Computers (ISWC 2002). Proceedings. Sixth International Symposium on*: 101–108. IEEE.
- Bachmann, Anja, Christian Borgelt, and Gy Gidfalvi. 2013. "Incremental Frequent Route Based Trajectory Prediction." In *Proceedings ACM SIGSPATIAL IWCTS*: 49–54. ACM.
- Bar-Gera, Hillel. 2007. "Evaluation of a cellular phone-based system for measurements of traffic speeds and travel times: A case study from Israel." *Transportation Research Part C: Emerging Technologies* 15, no. 6: 380–391.
- Ben-Abdessellem, Fehmi, Andrew Phillips, and Tristan Henderson. 2009. "Less is more: energy-efficient mobile sensing with senseless." In *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*: 61–62. ACM.
- Chaitin, Gregory J. 1982. "Algorithmic information theory". *John Wiley & Sons, Inc.*
- Gidfalvi, Gy, Christian Borgelt, Manohar Kaul, and Torben Bach Pedersen. 2011. "Frequent route based continuous moving object location-and density prediction on road networks." In *Proceedings of ACM SIGSPATIAL GIS'11*: 381–384. ACM.
- Gidfalvi, Gy, and Fang Dong. 2012. "When and where next: individual mobility prediction." In *Proceedings of ACM SIGSPATIAL MobiGIS 2012*: 1–8. ACM.
- Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. "The WEKA data mining software: an update." *ACM SIGKDD explorations newsletter* 11, no. 1: 10–18.
- Herrera, Juan C., Daniel B. Work, Ryan Herring, Xuegang Jeff Ban, Quinn Jacobson, and Alexandre M. Bayen. 2010. "Evaluation of traffic data obtained via GPS-enabled mobile phones: The *Mobile Century* field experiment." *Transportation Research Part C: Emerging Technologies* 18, no. 4: 568–583.
- Jenkins, Michael P., Geoff Gross, Ann M. Bisantz, and Rakesh Nagi. 2011. "Towards context-aware hard/soft information fusion: Incorporating situationally qualified human observations into a fusion process for intelligence analysis." In *Cognitive Methods in*

- Situation Awareness and Decision Support (CogSIMA), 2011 IEEE First International Multi-Disciplinary Conference*: 74–81. IEEE.
- Jensen, Christian S., and Stardas Pakalnis. 2007. "TRAX: real-world tracking of moving objects." In *Proceedings of the 33rd international conference on Very large data bases*: 1362–1365. VLDB Endowment.
- Jing, Jin, Abdelsalam Sumi Helal, and Ahmed Elmagarmid. 1999. "Client-server computing in mobile environments." *ACM computing surveys (CSUR)* 31, no. 2: 117–157. ACM.
- Kawasaki, Tomoya, and K. Axhausen. 2009. "Choice set generation from GPS data set for grocery shopping location choice modelling in canton Zurich: Comparison with the Swiss Microcensus 2005." *Arbeitsberichte Verkehrs-und Raumplanung* 595.
- Klepeis, Neil E., Andy M. Tsang, and Joseph V. Behar. 1996. "Analysis of the National Human Activity Pattern Survey (NHAPS) respondents from a standpoint of exposure assessment." *US EPA, Las Vegas, NV*.
- Kim, Donnie H., Younghun Kim, Deborah Estrin, and Mani B. Srivastava. 2010. "Sensloc: sensing everyday places and paths using less energy." In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*: 43–56. ACM.
- Kohavi, Ron. 1995. "A study of cross-validation and bootstrap for accuracy estimation and model selection." In *IJCAI*, vol. 14, no. 2: 1137–1145.
- Lee, Choonoh, Giwan Yoon, and Dongsoo Han. 2012. "A context-based energy optimization algorithm for periodic localization in smartphones." In *Proceedings of the First ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems*: 86–92. ACM.
- Li, Hainan, Randall Guensler, Jennifer Ogle, and Jun Wang. 2004. "Using global positioning system data to understand day-to-day dynamics of morning commute behavior." *Transportation Research Record: Journal of the Transportation Research Board* 1895, no. 1: 78–84.
- Li, Xia, and Anthony Gar-On Yeh. 2004. "Analyzing spatial restructuring of land use patterns in a fast growing region using remote sensing and GIS." *Landscape and Urban planning* 69, no. 4: 335–354.
- Llinas, James, and David L. Hall. 1998. "An introduction to multi-sensor data fusion." In *Circuits and Systems, 1998. ISCAS'98. Proceedings of the 1998 IEEE International Symposium on*, vol. 6: 537–540. IEEE.
- Martin, Eladio, Oriol Vinyals, Gerald Friedland, and Ruzena Bajcsy. 2010. "Precise indoor localization using smart phones." In *Proceedings of the international conference on Multimedia*: 787–790. ACM.
- Mladenov, Martin, and Michael Mock. 2009. "A step counter service for Java-enabled devices using a built-in accelerometer." In *Proceedings of the 1st international workshop on context-aware middleware and services: affiliated with the 4th international conference on communication system software and middleware (COMSWARE 2009)*: 1–5. ACM.
- Ratti, Carlo. 2004. "Urban texture and space syntax: some inconsistencies." *Environment and Planning B: Planning and Design* 31, no. 4: 487–499.
- Ravindranath, Lenin, Arvind Thiagarajan, Hari Balakrishnan, and Samuel Madden. 2012. "Code in the air: simplifying sensing and coordination tasks on smartphones." In *Proceedings of the Twelfth Workshop on Mobile Computing Systems and Applications*: 4. ACM.
- Reddy, Sasank, Min Mun, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava. 2010. "Using mobile phones to determine transportation modes." *ACM Transactions on Sensor Networks (TOSN)* 6, no. 2: 13.
- Schüssler, Nadine, Montini, Lara and Dobler, Christoph. 211. "Improving post-processing routines for GPS observations using prompted-recall data". *Eidgenössische Technische Hochschule Zurich, IVT, Institute for Transport Planning and Systems*.
- "Stat Counter". 2014, accessed on 29.03.2014 http://gs.statcounter.com/#mobile_vs_desktop-ww-monthly-201010-201310
- Stenneth, Leon, Ouri Wolfson, Philip S. Yu, and Bo Xu. 2011. "Transportation mode detection using mobile phones and GIS information." In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Sys-*

- tems*: 54–63. ACM.
- Stopher, P., P. J. Bullock, and F. Horst. 2003. "Conducting a GPS survey with a time-use diary." In *Transportation Research Board 82nd Annual Meeting Pre-print CD-ROM*.
- Stopher, Peter, Eoin Clifford, Jun Zhang, and Camden FitzGerald. 2008. "Deducing mode and purpose from GPS data." *Institute of Transport and Logistics Studies*.
- Verburg, Peter H., Koen P. Overmars, and Nol Witte. 2004. "Accessibility and landuse patterns at the forest fringe in the northeastern part of the Philippines." *The Geographical Journal* 170, no. 3: 238–255.
- Ting, Kai Ming. 2010. "Precision and Recall." In *Encyclopedia of Machine Learning*: 781. Springer US.
- Wolf, Jean, Randall Guensler, and William Bachman. 2001. "Elimination of the travel diary: Experiment to derive trip purpose from global positioning system travel data." *Transportation Research Record: Journal of the Transportation Research Board* 1768, no. 1: 125–134.
- Wolf, Jean, Marcelo Oliveira, and Miriam Thompson. 2003. "Impact of underreporting on mileage and travel time estimates: Results from global positioning system-enhanced household travel survey." *Transportation Research Record: Journal of the Transportation Research Board* 1854, no. 1: 189–198.
- Xiao, Jieying, Yanjun Shen, Jingfeng Ge, Ryutaro Tateishi, Changyuan Tang, Yanqing Liang, and Zhiying Huang. 2006. "Evaluating urban expansion and land use change in Shijiazhuang, China, by using GIS and remote sensing." *Landscape and urban planning* 75, no. 1: 69–80.
- Zheng, Yu, Like Liu, Longhao Wang, and Xing Xie. 2008. "Learning transportation mode from raw gps data for geographic applications on the web." In *Proceedings of the 17th international conference on World Wide Web*: 247–256. ACM.
- Zhuang, Zhenyun, Kyu-Han Kim, and Jatinder Pal Singh. 2010. "Improving energy efficiency of location sensing on smartphones." In *Proceedings of the 8th international conference on Mobile systems, applications, and services*: 315–330. ACM.