Delay-tolerant positioning for location-based logging with mobile devices

André Coelho, Filipe Meneses, and Rui José

Centro Algoritmi, University of Minho, Guimarães, Portugal
{acoelho, meneses, rui}@dsi.uminho.pt

Abstract. In this paper, we explore the viability and possible limitations of a delay-tolerant positioning process for location-based logging. Instead of estimating the device position on the device, a delay-tolerant technique stores the radio and Wi-Fi data needed to calculate that position, sends it to a server when appropriate, and it is the server that will calculate the position. This is very suitable for location logging because it supports the generation of frequent position registers without incurring in high communication and energy costs. We have conducted a controlled experiment to assess the accuracy of this technique and the results suggest that the accuracy of the positions calculated on the server using this technique is the same as those that can be obtained by calculating the position directly on the device.

Keywords. Location; Urban Computing; Location logging; Human mobility

1 Introduction

Sensing is critically important for Ambient Intelligence, which relies to a large extent on the ability to perceive the physical environment and the activities taking place in it. Mobile phones, with their already substantial data capture and connectivity capabilities, have a unique potential to become powerful sensing devices for uncovering new knowledge about the realities of our world and the patterns of Human behaviour [1]. In particular, considering their widespread use and their continuous presence in people's lives, they represent a major resource for location-based data collection. For example, to study mobility patterns within cities, there is a need to collect traces of users moving across the city in their daily life. In Experience Sampling studies [2, 3], there is a need to register, either implicitly or as part of an explicit user action, events as they occur in people's daily lives and annotate them with location information that will normally be crucial for their interpretation.

Location-based logging is fundamentally shaped by the need to combine frequent device positioning with the consequences that the process can have for users. These processes normally involve recruiting people to run the data collection applications on their own mobile phones and as part of their normal daily activities. This is crucial for generating realistic data and enabling larger scale studies. However, if the data collection implies significant energy, communication or privacy costs for users, it will be-

come a severe obstacle to large scale use and volunteer recruitment. Therefore, in our location logging processes we have included two important design principles: the first is avoiding the use of GPS. We would always need an alternative solution because we need to consider indoor locations, but the key issue is that the continuous use of the GPS would necessarily have a very high cost in terms of power consumption [6]; the second implication is to avoid depending on connectivity. In part this is also important to save energy, but since many people will not have a flat-rate data plan, there is also the issue that they will not accept the potential costs associated with data communications. The independence from connectivity would also allow us to perform the positioning without having to wait for the availability of a network connection.

To comply with these principles, we introduce the concept of delay-tolerant positioning. In most location-based services, location is normally part of an interactive feature and thus needs to be immediately available. On the contrary, in location-based logging, location information is needed to annotate an event. Therefore it is possible to just store the information needed to determine location, and leave the actual location calculation to some later point in time. Our data collection application, stores on the device the radio and Wi-Fi data generated that is used by the location API to determine location and when a connection becomes available, a batch of GSM and Wi-Fi information is sent to a server that will then use that information to calculate the positions. This approach does not make any use of the GPS and works very well with only occasional connectivity. For location-based logging applications, this means that frequent positioning records can be generated without forcing the device owner to incur in significant power or network costs.

In this paper, we assess the viability of this delay-tolerant approach for location-based logging. In particular, we aim to compare the level of accuracy with what can be obtained directly on the device and also to study the effect of other variables in that accuracy, such as the effect of time spent at the location before the location estimation and the nature of the Wi-Fi landscape at the point where location was estimated.

After revising related work in the next section, we describe in section 3 the essential steps of our research methodology. In section 4 we present the main results regarding each of the proposed objectives of the study and, finally, in section 5 we provide our concluding remarks.

2 Related Work

Yoshida et al. detail the creation and deployment of a localization system based on Wi-Fi fingerprinting [7]. The paper focuses on the effect and efficiency of the method used for acquisition of fingerprint data and the influence that it could have on the accuracy of the localization results produced. Even though this work has different objectives, it has informed us about the need to devise a data collection protocol that could accommodate for the variation in the network landscape.

Zandbergen [8] describes a study to estimate accuracy of positioning techniques, using a similar methodology. A mobile phone (iPhone 3G) was used to collect location data (A-GPS, Wi-Fi and Cellular positioning) at several distinct metropolitan

locations, and test the accuracy of each of the iPhone's positioning methods against a benchmark location (ground truth). We aimed to test the accuracy of a delay-tolerant location calculation (with Wi-Fi and GSM data), against the location data provided by the device on-site, and also in relation to ground truth.

PlaceLab [9] is a positioning system that allows users to locally (on their device) calculate a location, based on the BSSIDs and signal levels of nearby Wi-Fi access points. This location is calculated by crossing the data gathered in real-time by the device with information stored locally on a database, to which the user previously subscribed. While this may address some of the connectivity issues we have identified, it would have important disadvantages in terms of deployment, given the need to install additional client-side software and database.

Herecast [10] allows users to determine their symbolic location, e.g. building floor. Information about locations is kept in a database that is maintained and accessed by the community. While an alternative for cases in which information only needs to be generated occasionally and in familiar locations, this is not suitable for frequent location logging.

BikeNet [11] supports the collection of data related to performance, environmental and health aspects of cycling, and provides an example of a sensing system that demonstrates the effectiveness of the overall approach of relaying sensing data back to a server to address specific requirements of the sensing process.

3 Delay-tolerant location data collection

To investigate the potential and limitations of delay-tolerant positioning in an off-the-shelf mobile phone, we have devised a controlled experiment in which we conducted multiple positioning calculations at known locations using our own location data collection application to obtain real-time and delay-tolerant positions.

3.1 The data collection Application

To support the data collection process, we have developed an Android application to generate position records. The Android platform supports the Google location API [4] that allows devices to calculate their position using multiple types of data providers. In devices with a GPS chip, the GPS provider determines location with a high degree of accuracy and no data costs, but with a high energy cost. The network provider makes a combined use of GSM and Wi-Fi data by gathering data from two Android APIs: TelephonyManager API and WifiManager API. The first generates information about the cellular network, e.g. local Cell IDs and respective signal level, for both the cell that the device is connected to, as well as neighbouring cells. The other enables the device to perform scans of the surrounding Wi-Fi radio landscape and retrieve the BSSIDs and signal levels of nearby Wi-Fi access points. When an application calls *requestLocationUpdates()* in the LocationManager API [5], information from all the active providers is obtained, sent to an undisclosed Google service and the respective position is returned. In our application, we disabled the GPS pro-

vider and used only data from the network provider. When a position was to be estimated, the procedure involved the following steps: 1) obtain data generated by the network provider 2) determine location using the LocationManager API; 3) generate a position record to be stored on the mobile device and uploaded to the server when appropriate. Each record comprises the following information:

- Reading ID: A unique hash generated from the timestamp;
- Timestamp: A timestamp of when the reading was done;
- LocationID: A key identifying the test location where the reading was made;
- Network Provider info: This is the information about the Wi-Fi and 2G GSM landscape at the moment of the reading, as generated by the network provider;
- Real-time network position: The position as estimated on the device using information from the network provider and respective accuracy estimation.

3.2 Data Collection

We have selected a total of 11 collection locations, as listed in Table 1. Considering that the nature of locations may affect the quality of the positioning process, we chose a diversified set of locations. In particular, we have selected 6 indoor locations and 5 outdoor locations. The outdoor locations were all located in residential areas with high density of Wi-Fi access points, and thus enough radio information for positioning techniques. To address the fact that different locations may have much stronger and much more frequent variation in their Wi-Fi radio landscape than others, we included 4 locations on campus, where we would expect the radio landscape to be more stable, and 7 on other locations across town, where a more dynamic landscape would be expected.

Table 1. Data collection locations

Indoor	***Outdoor***
Campus room, Campus cafeteria, Campus library, Café in the city, Cinema, Restaurant	Outside rural residence 1 and 2 Outside urban residence 1 and 2 Outside location on Campus

Additionally, we have defined a data collection protocol that explicitly addressed the possible effect of movement patterns in the data collected. In our target scenario, we should be able to track the position of a person that may be moving in a city, and therefore may only spend a few seconds at any given location.

To explicitly address the possible effects of movement we defined a data collection protocol, in which the researcher would activate the application at some distance from the reference location. He would then approach the reference location and immediately, upon arrival, trigger the data collection process. This would immediately generate the first record. The researcher would then wait at the reference location for the application to automatically generate the second record, two minutes later, and the third record, four minutes later. This was repeated twice at different times of the day.

3.3 Data processing

Once in the server, a Perl script was used to calculate locations. The script parsed each location record and generated a JSON request with the respective GSM and Wi-Fi information to be posted through HTTP to Google's geolocation service[1] that returns a JSON reply containing the estimated position. The script then decodes the JSON response and stores the result as the delay-tolerant position for that record. We have made this estimation multiple times over the experience period to accommodate possible changes in accuracy through time.

To compare the accuracy of the position records, we have defined a ground truth by setting a reference position for each of the locations in our study using the coordinates obtained from a high-sensitivity GPS navigator (Garmin eTrex Legend HCx). For indoor locations, we used satellite imagery from Google Maps, always oriented towards north, and overlaid perfect squares over the images where one of the vertexes would overlay the point whose coordinates we wished to determine. Having done so, we then proceeded to take GPS coordinates of 2 of the vertexes (from the same edge) and then we used the Haversine formula [12] to calculate the distance between those 2 points relative to the Earth's surface.

$$haversin\left(\frac{d}{r}\right) = haversin\,(\phi_2 - \phi_1) \times \cos(\phi_1) \times \cos(\phi_2) \times haversin(\Psi_2 - \Psi_1) \tag{1}$$

In formula (1) d is distance, r is radius (the radius of the Earth in our particular case), finally ϕ1 and ϕ2 and Ψ1 and Ψ2 are respectively, the X and Y coordinates for the relevant points. We solve the haversine for d and obtain the distance. Afterwards, having performed the distance calculations, we chose a vertex from each square, along which we would be travelling the previously determined distance, along a fixed bearing according to the following formulas for latitude and longitude:

$$lat_{dest} = \text{asin}\left(\sin(lat_{orig}) \times \cos\left({}^{d}/_{r}\right) + \cos(lat_{orig}) \times \sin\left({}^{d}/_{r}\right) \times \cos(\Theta)\right) \tag{2}$$

$$lon_{dest} = lon_{orig} + \text{atan2}\left(\sin(\Theta) \times \sin\left({}^{d}/_{r}\right) \times \cos(lat_{orig}), \cos\left({}^{d}/_{r}\right) - \sin(lat_{orig}) \times \sin(lat_{dest})\right) \tag{3}$$

For formulas (2) and (3), d and r are again distance and radius (Earth's radius) respectively, Θ is the bearing (in radians, clockwise from north, i.e., North = 0, East = 90, South = 180 and West = 270, given a conversion from degrees to radians), the rest of the variables are self-explanatory in name. In Formula (3) we use atan2(Y, X) which is a variant of the arctangent function which returns the arctangent of Y/X in the range –Π to Π (mathematical PI).

4 Results

Using our data collection application, we generated position records at the 11 chosen locations during a three week period, with three weekly observations at each of

[1] http://www.google.com/loc/json

the reference locations, two times a day. This resulted in a total 54 readings per location, or 594 readings in total. These were processed at the server at different moments, generating a total of 2816 delay-tolerant positions.

To assess the accuracy of the delay-tolerant positions, we have used two different types of reference data: the estimated error (reported) given by the Google location API at the server and at the device; and our own estimation of that error (real) based on the ground truth positioning data calculated for each of the locations. Having determined all the ground truth points, we then calculated for each location the distance from this reference position to each of the position records, including the delay-tolerant positions.

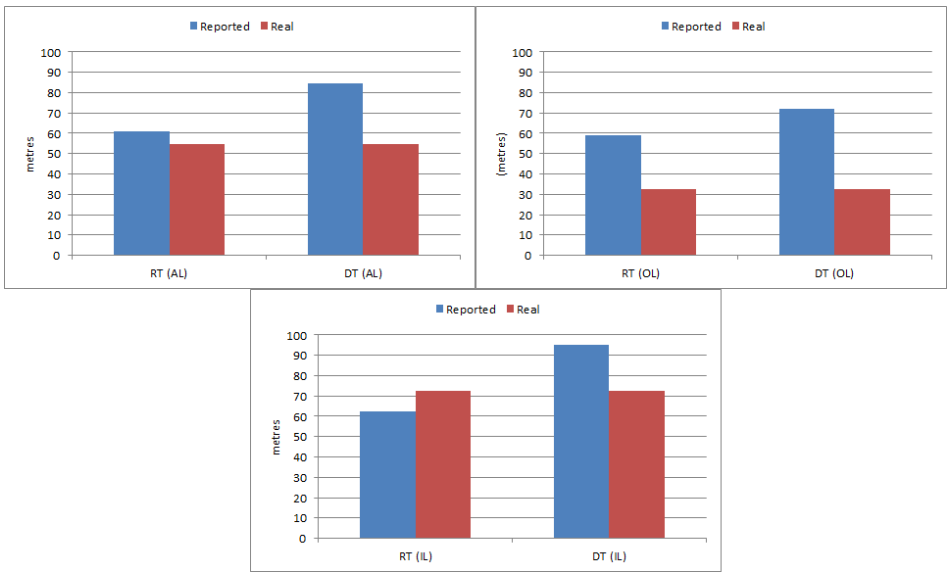

Fig. 1. Real-time (RT) vs delay-tolerant (DT) error results in reading 1 (left – all locations (AL), right – outdoor locations (OL) and bottom – indoor locations (IL)).

The main conclusion from Fig. 1 is that, when considering the error in relation to the ground truth, there are no observed differences between the location determined in real-time at the device and the location determined later at the server. This is the most fundamental observation of this study in the sense that it backs up our initial hypotheses regarding the viability of delay-tolerant positioning. It is also relevant to note that despite the differences in accuracy, the similarity between real-time and delay-tolerant locations existed for both indoor and outdoor environments. We can observe, however, that the error estimated by the Google Location API is shown to be lower in real-time when compared to the error reported by the delay-tolerant estimation process, for all cases but one. We have no explanation for this behaviour of the Google API, but since the effective accuracy is not affected, it should not constitute any sort of problem for location data logging.

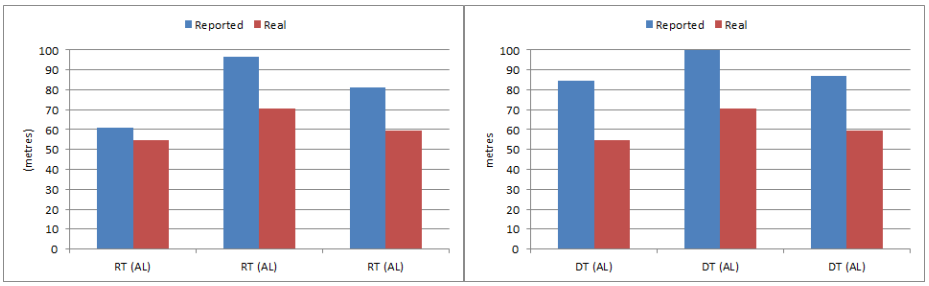

Fig. 2. Left – Real-time (RT) error results for all locations (AL) in readings 1, 2 and 3. Right – Delay-tolerant (DT) error results for all locations (AL) in readings 1, 2, and 3.

Fig. 2 reveals the errors obtained for all locations in all three readings, for the real-time and the delay-tolerant processes. For all scenarios the first reading is always the best, with the third reading coming a close second and the second reading always being the worst. In regard to the time when the location was estimated on the server, we have not observed any meaningful effect.

5 Conclusions and Future Work

The main conclusion of this work is that a delay-tolerant positioning process is a perfectly viable alternative approach for our usage scenario, especially considering that in terms of real error (in relation to ground truth) both methods achieve the same performance. This is a very important contribution to inform the design of any sort of location-based logging tools in which, as in our case, the position information is not needed at the moment of logging. The second conclusion is that time spent at the target location does not improve the accuracy of the positioning process. Moreover, there seems to be no gain whatsoever in staying more than 2 minutes at a given location. Together these two observations suggest that this process will perform well for the generation of location traces in high mobility scenarios. The only observed difference is in the estimated error that is associated with the location estimations, but this should not have any impact for most application domains.

A limitation of this study is our lack of knowledge about the internals of the Google location API. There are no public details about how it uses the radio and Wi-Fi information to calculate position, and whatever the current approach might be, it may suddenly change without any prior announcement. Such changes could possibly affect the results obtained in this study and lead to potentially different conclusions. Also, the internals of specific devices and particularly their support for capturing radio and Wi-Fi information may also vary and lead to potentially different results in specific types of mobile phones. As such, one possible future direction of research would be to conduct the experiment with several different devices and analyse the results obtained to get a grasp on what sort of variability can be expected from different devices in terms of positioning error. Another line of research would be to understand the maximum validity of the network information stored by the mobile device

and sent later to the server. As network information associated with a position evolves, observations made in the past will eventually become unsuitable for an adequate location determination. Understanding the timescale in which this effect may become relevant would help in defining upload policies for location logging tools.

References

1. Reichenbacher, T.: Geographic relevance in mobile services. Proceedings of the 2nd International Workshop on Location and the Web - LOCWEB '09. pp. 1-4. ACM Press, New York, New York, USA (2009).
2. Hektner, J.M., Schmidt, J.A., Csikszentmihalyi, M.: Experience sampling method: Measuring the quality of everyday life. Sage Publications, Inc (2006).
3. Consolvo, S., Walker, M.: Using the experience sampling method to evaluate ubicomp applications. IEEE Pervasive Computing. 2, 24-31 (2003).
4. Google: obtaining-user-location @ developer.android.com, http://developer.android.com/guide/topics/location/obtaining-user-location.html.
5. Google: LocationManager @ developer.android.com, http://developer.android.com/reference/android/location/LocationManager.html.
6. Kjærgaard, M.B., Langdal, J., Godsk, T., Toftkjær, T.: EnTracked: energy-efficient robust position tracking for mobile devices. Proceedings of the 7th international conference on Mobile systems, applications, and services - Mobisys '09. p. 221. ACM Press, New York, New York, USA (2009).
7. Yoshida, H., Ito, S., Kawaguchi, N.: Evaluation of pre-acquisition methods for position estimation system using wireless LAN. Third International Conference on Mobile Computing and Ubiquitous Networking (ICMU 2006). pp. 148-155 (2006).
8. Zandbergen, P.A.: Accuracy of iPhone locations: A comparison of assisted GPS, WiFi and cellular positioning. Transactions in GIS. 13, 5-25 (2009).
9. Schilit, B.N., LaMarca, A., Borriello, G., Griswold, W.G., McDonald, D., Lazowska, E., Balachandran, A., Hong, J., Iverson, V.: Challenge: Ubiquitous Location-Aware Computing and the "Place Lab" Initiative. Proceedings of the 1st ACM international workshop on Wireless mobile applications and services on WLAN hotspots - WMASH '03. p. 29. ACM Press, New York, New York, USA (2003).
10. Paciga, M., Lutfiyya, H.: Herecast: An open infrastructure for location-based services using WiFi. WiMob'2005, IEEE International Conference on Wireless And Mobile Computing, Networking And Communications, 2005. pp. 21-28. IEEE (2005).
11. Eisenman, S.B., Miluzzo, E., Lane, N.D., Peterson, R.A., Ahn, G.-S., Campbell, A.T.: The BikeNet mobile sensing system for cyclist experience mapping. Proceedings of the 5th international conference on Embedded networked sensor systems - SenSys '07. p. 87. ACM Press, New York, New York, USA (2007).
12. Robusto, C.C.: The cosine-haversine formula. The American Mathematical Monthly. 64, 38-40 (1957).