Energy Efficient Localization Framework for Mobile Applications

Jamal A. Madni
University of California, Los Angeles
Bioengineering Department
Los Angeles, California, USA
jmadni@ucla.edu

Rahul Rao Basava
University of California, Los Angeles
Computer Science Department
Los Angeles, California, USA
brahulrao@ucla.edu

Ethan Chen
University of California, Los Angeles
Computer Science Department
Los Angeles, California, USA
ethanc@cs.ucla.edu

***Abstract*—Smart phones are increasingly becoming powerful sensor devices, which can be used to continuously sense the user's environment such as location coordinates. Numerous smartphone applications such as augmented reality and navigation requires accurate stream of location data. One of the major challenges in location sensing is energy efficiency. GPS is the primary source of location information and the continuous use of GPS has a major impact on the device's battery life. In this paper, we describe an energy efficient framework, which can provide a continuous stream of location data to the applications. We describe an adaptive duty-cycling strategy combined with a location estimation mechanism. We show that our location estimator provides reasonable accuracy while providing high-energy efficiency.**

I. INTRODUCTION

Mobile phones are powerful platforms with a wide variety of sensors, which enable the applications to be continuously aware of the user's environment. An ever growing number of applications requires continuous stream of the user's location to provide sufficient context to the information they process. A few (among the numerous) applications of continuous location sensing on mobile phones include participatory sensing [5], augmented reality [1], social networking [2], maps and navigation.

One of the main challenges in enabling continuous location sensing on mobile phones is energy efficiency. The power usage is different for different and localization primarily uses the GPS hardware which has a major impact on battery life [4]. Modern mobile operating systems such as Android provide WiFi based mechanisms to detect the location. Even though this mechanism is relatively energy efficient (compared to GPS), it sacrifices the location accuracy to a great extent and it limits its applicability. As a result, it is essential to use GPS for applications, which require higher accuracy.

In this paper, we describe a framework for energy efficient location detection for continuous sensing applications. Various mechanisms have already been mentioned in the literature for energy efficient localization [6, 7, 8]. Inspired from these mechanisms, our framework is implemented based on the following principles: 1) Reduce the use of GPS by using a combination less power hungry sensors (whenever possible) to detect and estimate location data, 2) Detect user movement and turn off GPS when the user is idle 3) Duty-cycle GPS and other sensors used in the framework.

One of the problems with duty-cycling is the impact on data accuracy. If the application requires location data between two duty-cycles, it will only receive the GPS coordinate from the previous duty-cycle. If the duty-cycle interval is large, the user can move a significant distance between two duty-cycles resulting in reduced accuracy. This accuracy can be improved by reducing the interval but only at the cost of energy efficiency. In this paper, we also describe a scheme to improve the accuracy without reducing the duty-cycle interval by using location estimation. We describe the design, implementation and evaluation of the of the energy efficient localization framework containing the location estimator.

Various mechanisms for location prediction have already been proposed. For example, [6] describes a mechanism called EnLoc, which used human mobility patterns as a heuristic to estimate the user location. In our scheme, we estimate the speed using the information from the previous duty-cycle and current movement pattern (which is calculated using the accelerometer). Our estimation scheme is based on user speed prediction using accelerometer samples. We propose the use of a caching mechanism, which dynamically learns the correlation between the value calculated from the accelerometer and the actual speed of the user.

Our implementation is based on an assumption that the user will continue to move in a straight line between two consecutive duty-cycles. Even with this assumption we show that our framework can provide reasonable data accuracy and high energy efficiency.

This paper is organized as follows. Section 2 describes the design details of the energy efficient localization framework and section 3 talks about the location estimator component of the framework, Section 4 mentions a few important details from the implementation and section 5 details the experimental results followed by the conclusion in section 6.

II. DESIGN

Our group chose to design a framework to provide location data to possible clients. The philosophy followed largely stems from the design philosophy used in sockets, in which the multiple potential streams and sources of sensor data used to provide a single location are abstracted away, presenting a single stream of location data to client programs. We divide up our sensors into modules roughly correlating with our three localization schemes. We break up our modules into a GPS based scheme, a WiFi based scheme, and a cache and WiFi based scheme. Our three schemes have differing power consumption figures as well as accuracy, due to the sensors and algorithms used within. This allows the framework to switch between the various schemes for better power saving, increased accuracy, and redundancy. The schemes used are named for the primary sensor used to obtain location. From these three schemes, the framework itself caches the last several location samples, for use in the last significant module in the system, the location estimator module.

The framework provides location data to clients, but the rate it polls its own modules is independent of the rate at which the clients require location data. Additionally, since the framework must allow module polling to be adjusted based on power concerns as well as redundancy concerns, the framework must incorporate an estimation module to fill in data at time points in between module samples. The estimation module is responsible for providing location data between module location fixes, as well as smoothing out location data based on location values cached by the framework itself. It uses location data coupled with bearing and velocity to return a predicted location.

The framework connects all of the aforementioned modules, decides which location module to use based on the set of sensors available for use, the remaining amount of system power, and the power savings settings specified by the user. If the client polls the framework at such a time far enough from a location module poll, the estimator module is used to provide a best guess location instead. As estimation algorithms go, accuracy degrades very severely very quickly, so the framework must rely on the next location module poll interval to lock location to an "accurate" value.

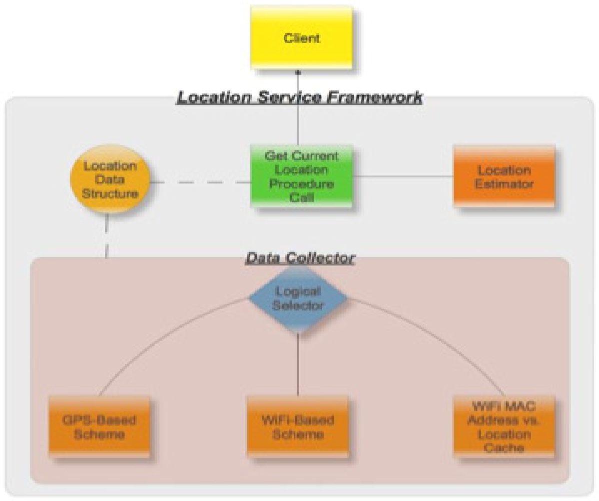

Figure 1. Overview of Design

The design lends itself to the stated goal of power efficiency by allowing the system to modify the set of sensors currently in use for a more power efficient set, but to adjust the rate at which it samples those sensors through the use of an estimation module.

A. *GPS*

The GPS module, so named because the primary source of location is the GPS, relies on sensing idle time to reduce its power consumption to the minimal it can. This is significant as the GPS sensor's power usage is one of the highest, as illustrated in this chart. We bring GPS power usage in this scheme down by using the 802.11 WiFi and accelerometer sensors to detect idleness, instead reporting back the last known GPS location, only using GPS for location when under periods of motion and location change.

While Figure 1 shows that one of the few sensors that does in fact consume just as much power as the GPS is the 802.11 WiFi chip, Figure 1's data shows power consumption for WiFi in a data transfer state. Our scheme uses WiFi only to sample the MAC addresses of the base stations broadcasting within range. These values are cached in the GPS module, to be used later as a fingerprint identifying the particular area it was sampled in. Since base stations may or may not be visible in repeated scans of the same area, the cache includes a similarity comparator, which will match area fingerprints provided the base station MAC addresses seen are within some percentage of similarity of each other. By comparing a scan's fingerprint against the cache, we can establish whether or not the current location is new. This allows the GPS sensor to establish idleness with relatively good power consumption.

While 802.11 WiFi can be used to establish whether the current location has changed, in order to ascertain whether the system is in motion, the accelerometer is used. When WiFi scans reveal an unknown location fingerprint, the accelerometer is polled for a movement factor. This movement factor will be discussed in more detail later, but for now it is sufficient to say that the movement factor is a sum value of acceleration in 3D space. The GPS module uses this movement factor to decide whether the system is in motion, and from there whether to poll the GPS sensor for a more accurate fix.

By taking advantage of the usage profile of mobile phones, which we believe tends towards long stationary periods punctuated by short periods of movement, the GPS module designed has the potential to reduce power consumption by a significant amount. However, of the three sensor modules designed, the GPS module is still the most power hungry. On the scale of power consumption versus accuracy, the GPS module still tends towards accuracy.

B. *WiFi*

The WiFi module is named for the fact it acquires location by polling for WiFi signatures and comparing against a 3rd party online WiFi location database. There are several of these services available, including Skyhook [3] and Google's own WiFi location implementation available on its Android platform. This module uses the WiFi location sources in an analogous manner to the GPS sensor. As noted before in Figure 1, the WiFi chip's power consumption is high, but the assumption again holds that this is for data transfer. The opportunity for power savings here comes from the fact that the WiFi location is based around signal strength and base station MAC addresses. By avoiding constant trips to the location database when idle, power can be saved. However, we do not expect this approach to yield as significant savings as the scheme that uses GPS due to the need to periodically use the WiFi to scan the current location.

As noted before, the WiFi module is very similar to the GPS module in terms of design. They share the same general algorithm of polling the location source under movement while scanning using the WiFi and accelerometer for fingerprints and movement factors, respectively. The WiFi scans provide a test for idleness, while the movement factor obtained from the accelerometer is used in the decision to poll the main location sensor. Power usage is primarily minimized by the scan for idleness.

Since the WiFi module is so similar to the GPS module, its intended use case is that of a backup to the GPS module. Should the GPS be unavailable due to a physical lack of GPS hardware, user GPS use restriction, lack of GPS signal, or even GPS sensor damage, the WiFi module can be used as a replacement. Obviously the WiFi module's accuracy falls compared to GPS, as systems such as Skyhook report accuracy of approximately 10-30 meters [3]. Additionally, we anticipate that a comparison of energy efficiency against the GPS module will result in modest gains at best. Again, this is why we have designed the WiFi module from the outset as a backup or replacement for the GPS module.

C. *WiFi & Cache*

This module is named for the fact that it gets location based on WiFi fingerprints already present in the cache. As mentioned before, both the GPS and WiFi modules incorporate a cache of WiFi location fingerprints. The idea behind this module is to use the cache of location fingerprints directly to look up locations. The accelerometer and compass is then used to provide a bearing for use by the estimator module. In essence, this module is in many ways a micro-implementation of a WiFi location scheme, such as Skyhook. Because this scheme relies on local sensors only, it is the most limited and inaccurate, and it is only used in the case that GPS and WiFi location have failed, have become too power- intensive, or have become user-restricted.

The reason we believe this scheme can be functional again lies with user usage profiles. Typical mobile phone usage tends towards repeat visits to a similar set of locations. From this observation we feel it is likely that cached locations will be visited again, and an estimated location can be generated from previously observed patterns. Additionally, as it relies most heavily on a lookup of data, power usage is minimal. However, it again is worth noting that this scheme is has many more drawbacks compared to GPS and WiFi location, including potentially poor accuracy, fragility, and failure to return data, and as such is intended for use in situations where any other localization modules are not available, not as a primary location source.

III. LOCATION ESTIMATION

The location estimation method is a threshold- based scheme between GPS duty cycles. Conceptually, our group partitioned GPS duty cycles into equal length time instances. A T' time threshold was then determined in an adaptive manner. Here T' was defined as the elapsed time from when our framework polled GPS to gather the exact location and the current time. If a client polled our framework with a get current location event inside of our T' value, our estimator would simply return the prior GPS duty cycle GPS sample, with the justification that the prior sample is still "fresh" enough to maintain relative accuracy. However, if a client polled our framework outside the T' value, then our location estimator method would be invoked.

Assuming the time threshold was set to T1, the location returned to satisfy the request of event E1 would simply be the location value when actually polling GPS at C1. However, the location estimator would be invoked for event E2 and return an approximate location based on the GPS sampled location at C1 as the method's baseline location. It is important to note that the location estimator method is an iterative process and thus has a cumulative error characteristic. For instance, if a get current location event E2* occurs between time instances T2 and T3, the location estimator will base its approximation computation for satisfying the event by using the location at the last satisfied event, namely the location returned at E1, as the baseline. Similarly, if a get current location event E2** occurs between T2 and C3, the estimator will satisfy this request by using the location generated for E2* as its baseline.

It is also important to note that these characteristics are then reset at the next GPS duty cycle. Thus, event E3 is satisfied in a similar manner as event E1, now using the GPS location at the next duty cycle, C2. Furthermore, based on this new GPS polled value, events E4, E4* and E4** will be dealt with in the same manner as E2, E2* and E2** respectively in the prior GPS duty cycle.

The algorithm for the location estimation scheme begins by calculating the system time and comparing this Delta T value with the previously defined T' threshold. If the Delta T is less than or equal to the T' threshold value, this means that the get current location event request lies within our threshold parameter and the previously sampled GPS value can simply be returned as it is "fresh" enough. If, however, the Delta T value is greater than the T' threshold value, then the event request is outside this freshness boundary and an approximation must be done on location to satisfy the request. Thus, the scheme enters an accelerometer module whereby the sum of the net acceleration of the mobile device is calculated and averaged over N very closely timed samples. Here N is again adaptive and the extremely minute time instances between samples are for the purpose of simulating a continuous time interval of the mobile phone's movement behavior. This behavior was qualified as the Current Movement Factor of the phone and is defined as the "movement state" of the phone.

If this Current Movement Factor is zero, we assume that velocity is also zero and thus we return the current location stored (either the location from the previous GPS sample or an approximate location generated for a prior event). If the Current Movement Factor is equal to the Previously Recorded Movement Factor, then we use the corresponding speed of the previously recorded movement factor as an input to our formula estimator. If non-equality holds between the above two movement factors, then the scheme enters a Movement Factor vs. Speed Hash Map, which is described in greater detail below. Upon entry into the Hash Map, if a corresponding speed exists for the Current Movement Factor, we use this speed as an input to the formula estimator, however, if no such corresponding speed exists, we simply use the previously recorded speed (which again corresponds to the speed of the previously recorded movement factor) as our input. Once the scheme enters the formula estimator, a new location is generated based on the movement factor, speed, bearing, time and location stored. The variable values are then updated and the scheme is then prepared for subsequent event requests.

The formula estimator calculates a new location based on the Haversine formula, which is a well-known trigonometric and iterative equation for navigation. Our version of the Haversine formula calculates a new location based on: 1.) the current location in our system, 2.) the distance determined as a product of the elapsed time from the derived location of the prior event request and the probable speed (derived from the Movement Factor vs. Speed Hash Map), and 3.) the bearing, which is assumed to be constant since the last GPS sampled location.

The Movement Factor vs. Speed Hash Map is a necessary data structure since directly calculating velocity as the integration of acceleration is exceptionally error- prone due to shaky movements. Thus the purpose of the Hash Map is to maintain a history of Movement Factor vs. Speed correlations for GPS samples collected during past duty cycles. The map tries to match the Current Movement Factor with a cache of speeds. One of the main problems with this mechanism is that there is a many-to-one correspondence between speed and movement factor, and thus the movement factor gets frequently overwritten. For instance, if an individual is walking 2 m/s while keeping the mobile device extremely stable in his/her hand, and then enters a car that will travel 40 m/s (but continues to keep the mobile device in exactly the same stable manner in his/her hand as in the walking case), then the movement factor has two corresponding velocities. Similarly, this property can be extrapolated to many velocities. Our system deals with this by autocorrecting the movement factor at every new GPS sample, and thus the Hash Map serves as an intelligent mechanism that attempts to learn, over time, the relationship between Movement Factor and Speed.

IV. IMPLEMENTATION DETAILS

In this section, we describe a few important aspects of the framework implementation.

As mentioned before, the GPS hardware is duty-cycled in our framework to reduce energy consumption. We use an adaptive duty-cycling scheme where the interval between two duty-cycles is dynamically calculated using the user's speed. This is done by making the distance traveled between two duty-cycles constant. In our current implementation, this value is defined as 100m and the duty-cycle interval is calculated by dividing this value with the current speed. The problem with this approach is that the user's speed could increase in large amount within the same duty-cycle interval, which in turn reduces the accuracy. But the location estimator mitigates this problem by estimating the locations in between regardless of the speed variations.

In order to detect the movement factor, the accelerometer is used, which is also duty-cycled at a constant rate. In our current implementation, this interval has been chosen to be 10s. At every duty-cycle, a few accelerometer samples are collected and the net acceleration on each sample is calculated. These net accelerations are averaged to get the movement factor.

In order to detect movement when the user is idle, WiFi signatures are continuously scanned to check if they have been changed. This is also done at a constant rate every 10 seconds. We also discard weak signals for better stability.

V. EXPERIMENTAL RESULTS

The evaluation was done on a Motorola Droid X using ANDROID Platform- 2.2. The evaluations were done to measure two factors, battery consumption and accuracy.

Accuracy is measured as the deviation from GPS coordinates as obtained from the sensor directly. It should be noted that in some cases the GPS coordinates directly obtained from the sensor may have significant deviation from the ground truth. Thus we are only measuring the deviation of the values as estimated by the framework with values as given by GPS sensor. We are not comparing the values with real path taken by the subject. For accuracy measurements we poll both the framework and the GPS sensor at the same time at a fixed rate. This information is recorded as pairs of locations obtained at the same time. Since the GPS sensor is continuously used, there is a significant drain on the battery. In the evaluations below, the experiments for battery consumption and accuracy measurement have been kept separate for this reason.

Accuracy was evaluated for two cases. The first is a set of measurements gathered when the subject is walking or running. The user will walk interspersed with random amounts of time spent in running or waiting at stop signs or road crossings. The second is a set of measurements gathered when the subject is driving.

A. *Test Scenario 1*

This scenario involved walking short distance with near constant speed. The subject does not run or stop at any time when the sample was collected. The walk was in a near straight line. The locations were sampled every 5 seconds. Figure 7 shows a plot of GPS coordinates (in yellow) and the locations returned by the framework (in blue).

The locations are marked with numbered symbols. The numbers indicate from which pair the point was plotted on the map. Some of points returned by the framework will not be visible since they coincide with the GPS locations and thus are rendered under the symbol for the GPS locations. This will be observed in all the test scenarios for accuracy measurement.

B. *Test Scenario 2 & 3*

These scenarios involve walking long distances with variable speed. The subject occasionally runs and stops at road crossings. The walk was in a near straight line. The locations were sampled every 5 seconds. Figure 8 and 9 shows a plot of GPS coordinates (in yellow) and the locations returned by the framework (in blue) for each of the scenarios respectively.

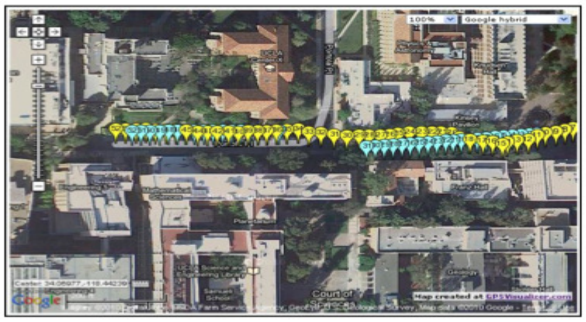

Figure 2. Test Scenario 1

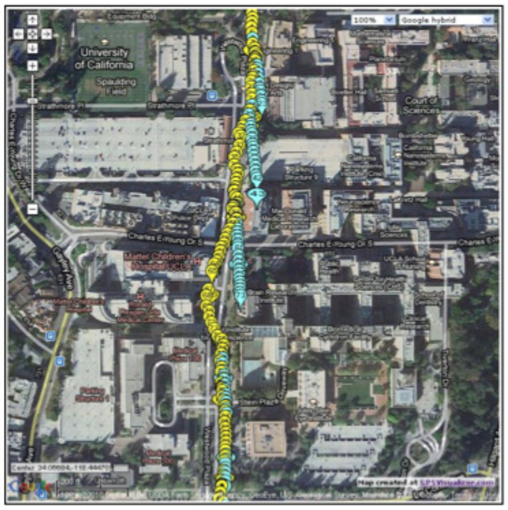

Figure 3. Test Scenario 2

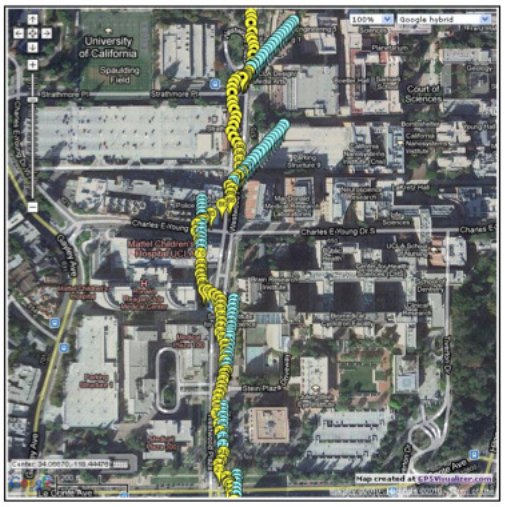

Figure 4. Test Scenario 3

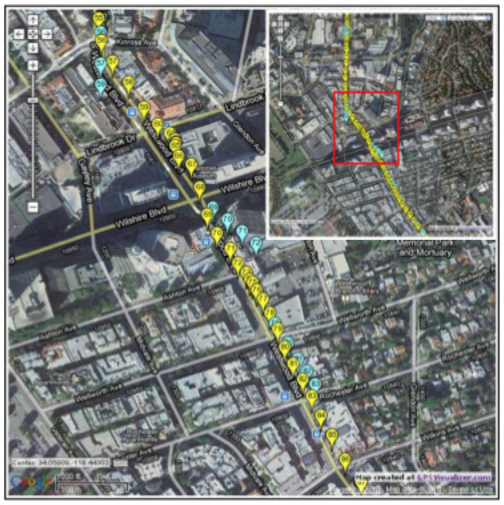

Figure 5. Test Scenario 4

C. *Test Scenario 4*

This scenario involved driving with variable speed. The subject accelerates at random times and stops at signals. The drive route was a near straight line except for one major change in direction. The locations were sampled every 3 seconds. The figure 10 shows a plot of GPS coordinates (in yellow) and the locations returned by the framework (in blue). The figure 10 (inset) shows the how the majority of the points are plotted. The figure 10 (main) shows a specific part of the plot where a change in direction occurred. The average speed was 6.146 m/s and the variance in speed was 29.22 m/s. These are measured using the speed information returned by the GPS location API.

D. *Discussion*

In our implementation the bearing of the subject was assumed constant and the results of the test scenarios reflects the same. As can be observed from the figures, the blue symbols, when not aligned with the yellow symbols, are at tangents to the path formed by the yellow symbols. This is directly due to the fact that the bearing is assumed at the point of divergence and that point onwards estimated points occur in a straight line whereas the GPS location changed direction. It can be observed that the errors are cumulative in nature and errors in previous estimations add to the current estimation until the framework synchronizes back with the GPS locations from the sensor. This can be markedly observed in Test Scenario 3 where the tangents are significantly off the mark. However in the case of driving (Test Scenario 4), the tangents are not so marked. This could be attributed to the adaptive cycling used for GPS sampling (which reduces the sampling interval for higher speeds) thus leading to a faster synchronization of the framework to the actual GPS coordinates.

E. *Power Measurement*

The power consumption measurements are obtained by running separate runs for a scripted route. The drop in battery power for the new framework at various instances of time with respect to the reference start point of measurement is recorded. Similarly drop in battery power for a direct GPS polling at various instances of time with respect to the reference start point of measurement is recorded. The interval for which the battery API in Android returns a change event is phone dependent. The Droid X showed only 10% battery changes.

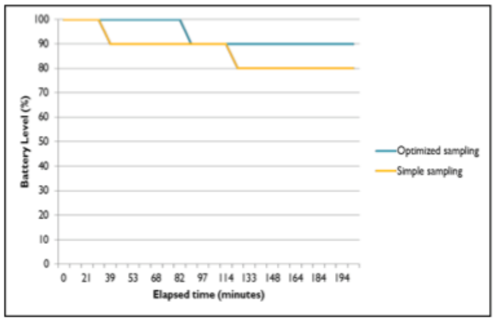

Figure 6. Battery Usage

Figure 11 shows the battery consumption for sampling using our framework in comparison to sampling using simple GPS sensor polling at a 30 second interval. The framework was sampled every 5 seconds. These measurements were done for over a 3 hour duration. The scripted scenario included walking and periods of no movement. Clearly it can be seen that the framework gives much better battery usage. However, increasing test time can provide much better results. Also we have to evaluate the extent to which this gain is due to entering a sleep state when the device is not in motion.

VI. CONCLUSION

The experimental results show that reasonable accuracy can be obtained by using location estimation mechanism even with the assumption that the user must only move in a straight line. One drawback of our approach is that successive estimations within the same duty-cycle will result in compounded error. Thus, if the application does not poll the location a large number of times, reasonable accuracy can be guaranteed.

We also show that adaptive duty-cycling combined with user idle detection can save a great deal of energy. This energy saving is directly proportional to the idle time of the user. With mechanisms such as WiFi signature detection in place, quick user movements could be detected so that GPS sampling can be restarted before the user covers a large distance.

We would like to acknowledge that a practical application of this mechanism would require detection of bearing changes between two different duty-cycles. This can be done using the compass and the orientation sensor and the investigation of this mechanism is left to our future work.

REFERENCES

[1] http://www.layar.com/
[2] http://cenceme.org/
[3] http://www.skyhookwireless.com/
[4] F.B Abdesslem, A. Phillips, T. Henderson, “Less is more: energy-efficient mobile sensing with senseless”, MobiHeld, 2009.
[5] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, M.B. Srivastava, “Participatory sensing”, SenSys, 2006.
[6] I. Constandache, S. Gaonkar, M. Sayler, R.R. Choudhury, L. Cox, “EnLoc: energy-efficient localization for mobile phones”, Infocom, 2009.
[7] M.B. Kjaergaard, J. Langdal, T. Godsk, T. Toftkjaer, “EnTracked: energy-ecient robust position tracking for mobile devices”, MobiSys, 2009.
[8] Y. Wang, J. Lin, M.Annavaram, Q. A. Jacobsen, J. Hong, B. Krishnamachari, N. Sadeh, “A framework of energy efficient mobile sensing for automatic user state recognition”, MobiSys, 2009.