Evaluating the iPhone as a Mobile Platform for People-Centric Sensing Applications

Emiliano Miluzzo, James M. H. Oakley, Hong Lu, Nicholas D. Lane,
Ronald A. Peterson, Andrew T. Campbell
Computer Science Department, Dartmouth College, Hanover, NH 03755, USA

Abstract

A number of mobile phones such as the Nokia N95 and Apple iPhone are being used by researchers to explore new people-centric sensing applications. These top-end phones include various sensors (e.g., accelerometer, proximity sensor, GPS, camera, microphone), radios (e.g., Bluetooth, WiFi, cellular), operating systems (e.g., Symbian, customized Mac OS X), and processors (e.g., 330 Mhz ARM, 412 Mhz ARM). While there is some data on the Nokia N95, little, however, is known about the ability of Apple's iPhone to support the necessary sensing, processing and communications needs of these emerging sensing applications. We present the first quantitative performance evaluation of the iPhone's sensors, localization engine, and networking stack while running CenceMe, a representative people-centric sensing application. We profile the performance of CenceMe running on the iPhone in terms of energy consumption and computational speed of its algorithms. One drawback of using the iPhone over the N95 is that it does not allow third party applications such as CenceMe to run as a background process, making continuous sensing problematic. The upside is that the iPhone offers a rich UI architecture, high computational capability, and an efficient application distribution system through Apple's App Store.

I. Introduction

New top-end mobile phones such as the Nokia N95 [7] and Apple iPhone [8] are enabling a new class of mobile, people-centric applications [1] [2] [3] [4] to emerge. While researchers have demonstrated that the Nokia N95, a Symbian-based mobile phone, can efficiently host sensing applications [2] [3] [4], little is known about the ability of the Apple iPhone to support people-centric sensing applications. Recently, Apple opened up to the deployment of third-party applications with the release of the iPhone SDK in March 2008. This, combined with its ease of use, rich UI, and efficient application distribution system through the Apple App Store makes the iPhone an appealing platform for development of new mobile applications. A natural question for our community is what are the trade-offs when implementing and deploying a sensing application using the iPhone; more specifically:

- How easy is it to program a sensing application on the iPhone?
- What are the pros and cons of the iPhone in comparison to other sensor capable mobile platforms?
- What is the energy profile when the iPhone's sensors, WiFi and cellular radio are involved in realizing the application?
- What is the processing performance of the iPhone when running signal processing algorithms such as fast fourier transform, a common tool used to interpret audio and accelerometer sensor data?

We address these questions in this paper. While the presentation of our results is limited due to space, we provide a short qualitative comparison of a number of devices used for mobile sensing including the Apple iPhone, Nokia N95, and Intel Mobile Sensing Platform (MSP) [11]. The main contribution of this paper is the observations and insights when running CenceMe, a representative people-centric sensing application [4], on the iPhone. Specifically, we quantitatively evaluate the iPhone's computational capability, energy profile, and localization accuracy. We believe this study will be useful to the growing community of iPhone developers, particularly, those interested in building people-centric sensing applications.

II. Comparison of Mobile Sensing Platforms: iPhone, N95, and MSP

In what follows, we present a short qualitative comparison of the Apple iPhone, Nokia N95 mobile phone, and the MSP. All these devices are actively being used in support of mobile sensing applications and systems development. The N95 is currently one of the top-end Nokia mobile phones equipped with an accelerometer and GPS, while the MSP is representative of the class of embedded devices used for human activity recognition research. A simple

TABLE I: Platform comparison for the Apple iPhone, Nokia N95, and Intel MSP

	iPhone	**Nokia N95**	**Intel MSP 430**
Processor	412 MHz ARM	330 MHz ARM	416 MHz Xscale
RAM	up to 70 MB	up to 128 MB	256 KB
ROM	20 MB	up to 160 MB	32 MB
Storage	up to 8GB/16GB	min-SD card (up to 8 GB)	mini-SD card (up to 8 GB)
Sensors	3-axis accel, mic, GPS	3-axis accel, mic, GPS	3-axis accel, mic, light, barometer, temp, IR, humidity, compass
Radio	WiFi	WiFi, Bluetooth	Bluetooth, Zigbee

comparison of some of the technical details of the three devices is reported in Table I. As shown in Table I, all three platforms present similar computational capabilities given similar processors, and large storage and ROM size. The RAM on the MSP is much smaller than on the iPhone and N95, which first and foremost are designed as mobile phones, hence the need to handle multiple processes at the same time including graphics computation. The MSP's short-range radio technology is flexible allowing the implementation of advanced pairing algorithms between nodes while the use of the iPhone and N95's short-range radio is limited to simple neighbor interactions. The main difference between the three devices is represented by the sensing capability; specifically, the MSP outshines both the iPhone and the N95 in terms of the number of available sensors. This is not surprising given that the MSP is an embedded purpose-built platform for activity recognition. However, even with a reduced set of on board sensors, the iPhone and N95 are powerful devices and capable of inferring human activites - for example, we have implemented the full featured CenceMe application on the N95 [4] as well as a version on the iPhone [10]. Providing mobile phones with more sensing capabilities (e.g., gyroscope) would greatly enhance the humans presence classification accuracy given the broader input to the classifiers feature vectors.

III. Programmability Characteristics of the iPhone

In what follows, we analyze the programmability characteristics of the iPhone. Any third-party application is handled by the iPhone OS using a sandboxing model which does not allow the application to access some of the iPhone functionalities (such as WiFi APIs or iTunes) for security reasons. A simplified version of a SQL database, namely *sqlite* [14], designed to run on resource constrained environments, is also supported as a means to ease application on-the-phone storage.

By following a systematic approach, we intend to answer the following question: what are the positive and negative aspects of the iPhone as a programmable platform? Although the iPhone presents a rich set of features making it potentially a good platform for the development of sensing applications, the iPhone SDK also provides some barriers in its current stage of development (i.e., iPhone SDK for iPhone OS 2.2). In what follow, we briefly discuss the pros and cons of the current iPhone development environment.

Advantages:

- *Programming Language*. The iPhone is programmed in Objective-C [12]. Objective-C is a superset of the C language, with some object oriented programming features. The advantage of Objective-C over other languages such as Symbian C++ adopted by Nokia, is that it is a quite simple language to learn and use. The iPhone APIs and emulator (which runs on desktop/laptop machines) make programmability, UI design, and code debugging an efficient process for developers.

- *APIs*. The APIs are well designed and documented, abstracting the developer from low level components. For example, the location engine API returns data transparently to the user regardless of whether the location coordinates come from WiFi, cellular triangulation, GPS, or a combination of sources. In addition, the accelerometer and microphone APIs are cleanly designed and make accessing these devices simple and strightforward to use.

- *Indoor Localization*. By using WiFi [6] and cellular triangulation to determine the location, the iPhone localization for indoor spaces is quite accurate, as discussed in Section IV. This is an important feature, for example, for mobile social networking applications considering that people spend a large amount of time indoors.

- *User Interface*. The iPhone experience is greatly enhanced by the Cocoa-Touch layer architecture [9] that provides for a good user experience. Combined with a powerful graphics framework, this makes the iPhone UI one of the best presentation layers of any mobile devices.

- *Application Distribution*. Apple provides an efficient way to distribute third-party applications to the public through the App Store [13]. Once an application is tested and approved by Apple, the application is posted on the App Store. After that the application can be downloaded and automatically installed on any iPhone.

Disadvantages:

- *Lack of Background Mode*. The main drawback of the iPhone is the lack of background mode to run third-party applications. This is enforced by Apple for security reasons. This means that anytime the phone goes into sleep mode or the user launches another application, the currently running third-party application is terminated. As a result of this design decision, sensing applications cannot provide continuous sensor data feeds. Therefore, applications can only generate intermittent data streams. This limits the effectiveness of continuous sensing application such as CenceMe. Apple's response to the lack of background capability is the *Push Notification Service* coming in the next releases of the SDK. With the push notification service, probes can be sent by the Apple backend servers, which, in turn, serve as relays for push messages sent by a sender host to a receiver iPhone. As the receiver iPhone is woken up by the probe the user is asked by the iPhone OS whether to let the application run in response to the probe message or not. It is worth noting that the Nokia N95 and Intel MSP support background mode and therefore support the implementation of continuous sensing applications.

- *Short-Range Radio API Limited Capability*. Currently, it is not possible to access directly the Bluetooth or WiFi radio stack APIs on the iPhone. The only way to exchange information between neighboring iPhones is by using the iPhone networking stack via the Bonjour service through WiFi. The short-range interactions of devices via this networking capability is therefore very limited and does not allow developers to build sophisticated pairing protocols.

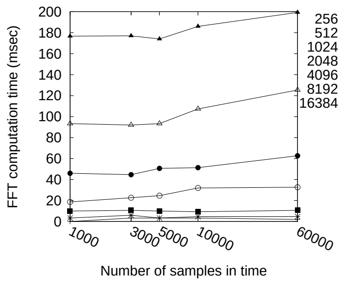
IV. Performance Evaluation

In this section, we report some initial results from a number of experiments aimed at evaluating the iPhone computational capability, battery duration, and localization accuracy by using the original iPhone model (without GPS) and the new iPhone 3G (with GPS) running, respectively, iPhone OS 2.0 and 2.1.

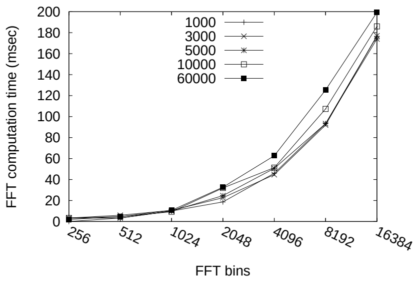
Computational Capability. In order to evaluate the processing power of the iPhone we run a fast fourier transform (FFT) algorithm, which is part of the CenceMe software suite, and measure the iPhone computation time. The FFT computation evaluation is performed during normal CenceMe usage process [10]. The FFT implemented as part of the CenceMe application is the Kiss FFT [15], a well known open source high performance FFT library. We choose the FFT as a means to evaluate the iPhone under high computational load because the FFT is a common tool used in inference techniques applied to sensor data such as accelerometer and audio data streams. As shown in Figure 1, the iPhone computation time up to 4096 FFT points is below 60 msec even for a large number (i.e., 60000) of sampled events in time. Many sensor data analysis algorithms make use of 512 - 2048 FFT points calculation, which means that they could efficiently leverage the iPhone's computational power. Large data bins in time, up to 60000 samples in our experiment, could also be quite efficiently handled by the FFT on the iPhone in at most 200 msec.

Battery Lifetime. We perform some experiments to quantify the battery drain of the iPhone when running CenceMe compared to the baseline power usage without CenceMe. We set the screen saver to off so that the phone never goes into standby mode. The battery duration for different data upload rates when CenceMe is running is compared to the duration when CenceMe is not running, as shown in Figure 2(a). With the phone's standby mode off and running CenceMe continuously, the battery lifetime spans between 4 hours and 37 min to 7 hours according to the upload rate. We then turn the screen saver back on and set it to 5 minutes and run CenceMe with the following cycle: run for 5 minutes, let the screen saver go off, leave the screen saver up for 5 minutes, wake the phone up for 5 minutes, and so on, until the battery discharges completely. In this way, for the same upload rates, we obtain a phone usage time (meaning time available to operate CenceMe) between 4 hours 50 min and 5 hours 20 min. (Note, the battery maximum usage is between 10-11 hours). This battery duration is similar to the duration obtained with iPhone usage patterns comparable to the one of our experiment when running different applications than CenceMe. This is because the prevalent battery drain is due to the iPhone LCD screen rather than the networking activity for data transmission/reception operated by CenceMe.

Localization Accuracy. To evaluate the localization accuracy of both the old model iPhone (without GPS) and the



(a)



(b)

Fig. 1: FFT computation time as a function of (a) the number of samples in time while varying the FFT bin size (as shown in the legend) and (b) the FFT bin size while varying the number of samples in time (as shown in the legend).

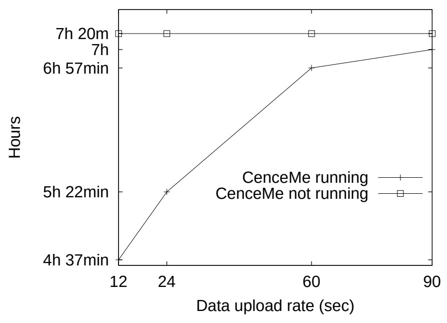
TABLE II: Localization accuracy for different places in the Dartmouth Campus - Legend: **C.S.** = Cellular Signal; **A** = Old iPhone localization accuracy (m); **B** = iPhone 3G localization accuracy (m); **C** = Garmin GPS accuracy (m); **D** = Old iPhone-Garmin GPS localization difference (m); **E** = iPhone 3G-Garmin GPS localization difference (m)

	Location	**WiFi**	**C.S.**	**A**	**B**	**C**	**D**	**E**
1	Computer Science Bld indoor	good	good	83	22	N/A	N/A	N/A
2	Computer Science Bld outdoor	good	good	44	17	14	29	36
3	Library outdoor	good	good	17	9	8	0	1
4	Library indoor	good	mediocre	13	5	N/A	N/A	N/A
5	Golf course	none	good	759	17	5	45	1
6	Engineering Bld	weak	weak	95	17	5	14	0
7	Main St.	none	weak	179	47	11	5	4
8	The Green	none	good	323	47	5	24	2

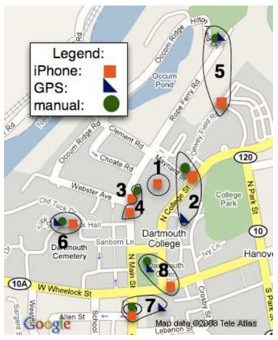
iPhone 3G (with GPS) we carry out the following experiment: we walk in the Dartmouth College campus with both the iPhone models and a Garmin eTrex GPS device. We record the geographical coordinates from the Garmin and the iPhone devices at eight different locations. Eight clusters are shown on the maps in Figure 2(b) and Figure 2(c), each cluster indicating the location manually tagged by the person carrying the devices, the location reported by the Garmin, and the location indicated by the old model and 3G iPhones. The old model iPhone's localization engine uses WiFi [6] and cellular triangulation. Therefore, when WiFi and/or the cellular coverage is poor the resulting localization accuracy is low. This can be seen for locations associated with clusters 5 and 8 where there is poor WiFi and/or cellular coverage. In case of clusters 1 and 4, which are indoor locations where GPS performs poorly, the iPhone localization is more accurate given the high quality of the WiFi and cellular signal. The old iPhone model estimates an accuracy between 13 and 759 meters, as shown in column A of Table II. Dartmouth College is located in the small college town of Hanover, NH, and subquently not served by many cell towers. Clearly, the availability of more cell towers would allow the iPhone's localization triangulation algorithm to be more accurate. The actual distance difference between the old iPhone and Garmin GPS reported locations, as shown in column D of Table II, is 45 m at most, indicating that the iPhone localization algorithm uses a conservative approach to estimate its accuracy. The GPS boosts the localization accuracy of the iPhone 3G, being particularly effective where there is a lack of WiFi coverage or when the cellular signal is poor. This can be seen from columns B and E of Table II where, respectively, the error estimated by the iPhone 3G and the localization difference between the iPhone 3G and Garmin GPS are reported. It is evident how the iPhone 3G-Garmin GPS localization difference is smaller than when using the old iPhone model.

V. Related Work

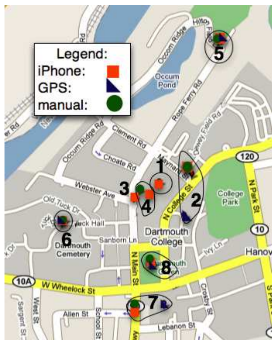
There is a growing body of work on the evaluation of sensing platforms for embedded sensing systems. For example [16] discusses the technical details and performance evaluation of the Telos motes, a widely used sensing platform in wireless sensor networks research. With the increasing interest in people-centric sensing applications, where sensing devices are carried by individuals, new sensing platforms are being developed, evaluated, and reported



(a)



(b)



(c)

Fig. 2: (a) Battery duration with and without CenceMe running while the iPhone screen saver is set to off. - Localization accuracy for eight different locations in the Dartmouth campus of (b) the old iPhone (no GPS), and (c) the iPhone 3G (with GPS).

in the literature. While the Intel MSP is used for activity recognition [11] there is little available in the literature on the evaluation of mobile phones for sensing. In [3] and [4] the authors present some early performance studies when using mobile phones for sensing applications. In [3], the authors show some energy profiling of the Nokia N95. In [4], the authors present a detailed evaluation of the N95 in terms of programmability and computational performance including detailed energy considerations.

VI. Conclusion

In this paper, we presented an evaluation of the iPhone running the CenceMe application. We quantitatively evaluated the iPhone's computational capability, energy profile, and localization accuracy. We showed that the computational capability of the iPhone is sufficient to handle high load FFT calculations. We also showed that iPhone's cellular and WiFi assisted localization outperforms pure GPS-based localization, as long as cellular and WiFi coverage are present. We believe this study is useful to iPhone developers, particularly those interested in building people-centric sensing applications.

Acknowledgments

This work is supported in part by Intel Corp., NSF NCS-0631289, and ISTS at Dartmouth College. ISTS support is provided by the U.S. Department of Homeland Security under award 2006-CS-001-000001, and by award 60NANB6D6130 from the U.S. Department of Commerce. The authors would like to thank Daniel Peebles for the useful insights provided to this work.

References

[1] A. Kansal, M. Goraczko, and Feng Zhao, Building a sensor network of mobile phones. In *Proc. of IPSN 2007*, April 25-27, 2007.
[2] E. Agapie, et al., Seeing Our Signals: Combining location traces and web-based models for personal discovery. In *Proc. of HotMobile 2008*, Napa Valley, CA, USA, February 25-26, 2008.
[3] Shravan Gaonkar, Jack Li, Romit Roy Choudhury, Landon Cox, Al Schmidt, Micro-Blog: Sharing and Querying Content through Mobile Phones and Social Participation. In *Proc. of MobiSys 2008*, Brekenridge, CO, USA, June 17-20, 2008.
[4] Emiliano Miluzzo, et al., Sensing Meets Mobile Social Networks: The Design, Implementation and Evaluation of the CenceMe Application. In *Proc. of SenSys 2008*, Raleigh, NC, USA, Nov. 5 - 7, 2008.
[5] Andrew T. Campbell, et al., The Rise of People-Centric Sensing. In *IEEE Internet Computing Special Issue on Mesh Networks*, July/August 2008.
[6] Skyhook Wireless. http://www.skyhookwireless.com/
[7] Nokia N95 series. http://www.nseries.com/index.html
[8] Apple iPhone. http://www.apple.com/iphone/
[9] Apple iPhone Dev Center. http://developer.apple.com/iphone/
[10] CenceMe portal. http://www.cenceme.org.
[11] Tanzeem Choudhury, et al., The Mobile Sensing Platform: An Embedded Activity Recognition System. In *IEEE Pervasive Computing*, April-June 2008, Vol. 7, No. 2, pp. 32-41.
[12] Objective-C. http://developer.apple.com/documentation/Cocoa/Conceptual/ObjectiveC/.
[13] Apple App Store. http://www.apple.com/iphone/appstore/.
[14] SQLite. http://www.sqlite.org/.
[15] Kiss FFT library. http://sourceforge.net/projects/kissfft/.
[16] Joe Polastre, et al., Telos: Enabling Ultra-Low Power Wireless Reasearch. In *IPSN/SPOT 2005*, April 25-27, 2005, Los Angeles, USA.