A Mobile Platform for Real-time Human Activity Recognition

Óscar D. Lara and Miguel A. Labrador
Department of Computer Science and Engineering
University of South Florida, Tampa, FL 33620
olarayej@mail.usf.edu, labrador@cse.usf.edu

***Abstract*—Context-aware applications have been the focus of extensive research yet their implementation in mobile devices usually becomes challenging due to restrictions in regards to processing power and energy. In this paper, we propose a mobile platform to provide real-time human activity recognition. Our system features (1) an efficient library, MECLA, for the mobile evaluation of classification algorithms; and (2) a mobile application for real-time human activity recognition running within a Body Area Network. The evaluation indicates that the system can be implemented in real scenarios meeting accuracy, response time, and energy consumption requirements.**

***Index Terms*—Body Area Networks; Human-centric sensing; Machine learning; Mobile applications.**

I. Introduction

Providing accurate and opportune information on people's activities and behaviors is one of the most important and well studied tasks in pervasive computing. Innumerable applications can be visualized for medical, security, entertainment, and tactical purposes. In recent years, the prominent development of sensing devices (e.g., accelerometers, cameras, GPS, etc.) has facilitated the process of measuring attributes related to the individuals and their surroundings. In addition, most sensors are nowadays equipped with communication capabilities which allow their integration with other mobile devices within Personal Area Networks (PANs) or Body Area Networks (BANs). However, most applications require much more than simply collecting measurements from variables of interest. In fact, additional challenges for enabling context awareness involve the design of techniques to perform data analysis and inference, as the raw data (e.g., acceleration signals or electrocardiogram) provided by the sensors are, in many cases, useless.

Even though the first works in *human activity recognition* (HAR) date back to the late 90's [1], there are still significant research challenges within the field. In this work, we concentrate on one of the most relevant ones: the implementation of a HAR system in a mobile phone. This task becomes difficult because of the energy and computational constraints present in the devices. Such limitations are particularly critical for activity recognition applications which entail high demands due to decoding, feature extraction, classification, and transmission of large amounts of raw data. Furthermore, to the best of our knowledge, available machine learning API's such as WEKA [2] and JDM [3] are not supported by current mobile platforms. This fact accentuates the necessity of an efficient mobile library for evaluating machine learning algorithms and implementing HAR systems in mobile devices.

At the same time, we foresee that a mobile HAR system will bring important advantages and benefits. For instance, a HAR system running in a BAN should substantially reduce energy expenditures, as raw data would not have to be continuously sent to a server for processing. The system would also become more robust and responsive because it would not depend on unreliable wireless communication links, which may be unavailable or error prone; this is particularly important for medical or military applications that require real-time decision making. Finally, a mobile HAR system would be more scalable since the server load would be alleviated by the locally performed feature extraction and classification computations.

In this paper, we introduce a mobile framework in support of real-time human activity recognition under the Android platform —reported as best-selling smartphone platform in 2010 by Canalys [4]— to address previously mentioned issues. A mobile application was implemented on top of Centinela [5], a HAR system based on acceleration and physiological signals to automatically recognize physical activities. The application utilizes the C4.5 classification algorithm as a proof of concept. The implementation and evaluation of our framework present the following main results and contributions:

- A library for the *mobile evaluation of classification algorithms* (MECLA) was successfully implemented.
- An application for real-time HAR was implemented in an Android cellular phone. It uses MECLA and supports multiple sensing devices integrated in a BAN.
- The evaluation shows that the system can be effectively deployed in current cellular phones. Indeed, the application can run for up to 12.5 continuous hours with a response time of no more than 8% of the window length and an overall accuracy of 92.6%.
- Different users with diverse characteristics participated in training and testing phases, assuring flexibility to support new users without the need of re-training the system.

The rest of the paper is organized as follows: Section II analyzes the state of the art in human activity recognition and cell phone implementations. Later, Section III describes the design of the system. Then, Section IV presents the evaluation methodology and main results. Finally, Section V summarizes the major conclusions and findings.

II. Related Work

The recognition of daily activities using accelerometer data has been the focus of extensive research [6]. Nonetheless, many systems rely on sensors placed in different parts of the body or need the user to carry computers and other devices [7]. This might be invasive, uncomfortable, expensive, and therefore, not suitable for practical activity recognition systems. Our proposed system only requires a single sensing device, which is comfortable and unobtrusive, as well as an Android cellular phone with Bluetooth connectivity. Moreover, we also consider physiological signals, in view of their usefulness to improve activity recognition accuracy [5].

Another important aspect is the data collection protocol followed by the individuals. In 1999, Foerster et al. [1] demonstrated 95.6% of accuracy for ambulation activities in a controlled data collection experiment, but in a natural environment, the accuracy dropped to 66%! In this work, we have carried out a naturalistic data collection procedure.

Human activity recognition is a well studied field yet very few works have successfully been deployed in mobile phones. In 2010, Berchtold et al. introduced *ActiServ* as an activity recognition service for mobile phones [8]. They make use of a fuzzy inference system to classify daily activities, achieving up to 97% of accuracy. Nevertheless, it requires a runtime duration in the order of days! When their algorithms are executed to meet a feasible response time, the accuracy drops to 71%. ActiServ can also reach up to 90% after personalization, in other words, a subject-dependent analysis (i.e., the system needs to be re-trained for new users). Brezmes et al. [9] proposed a mobile application for HAR under the Nokia platform; but they used the k-nearest neighbors classifier, which is impractical for mobile phones as it needs the entire training set —which can be fairly large— to be stored in the device. Besides, their system requires each new user to collect additional training data in order to obtain accurate results. Finally, Riboni et al. [10] presented COSAR, a framework for context-aware activity recognition using statistical and ontological reasoning under the Android platform. The system recognizes a number of activities very accurately yet it only supports statistical features and the multiattribute logistic regression classifier. In this work, we have included a variety of classification and feature extraction methods, providing a wider range of mobile pattern recognition applications. Further, we have studied the energy comsumption and response time of the system, something not formally evaluated in any of the previously described works.

III. Description of the System

The mobile application has three different user profiles: *trainer*, *tester*, and *administrator*. A tester is a regular user whose activities are to be monitored and recognized. A trainer is, as indicated by its name, intended to collect training data that can be used to build additional classification models. In this profile, the user is also required to enter the real performed activity as a ground truth. Finally, the administrator is able to do training, testing, and modifying the application settings. Figure 1 shows the main components of the system and their interrelationships. First, the *communication* and *sensing* modules allow for collecting raw data from the sensing devices.

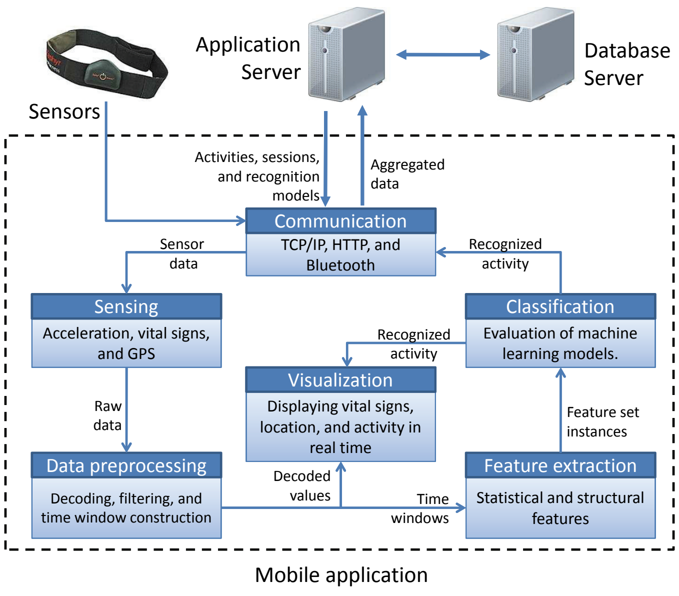

Application Server
Database Server
Sensors
Activities, sessions, and recognition models
Aggregated data
Communication
TCP/IP, HTTP, and Bluetooth
Sensor data
Recognized activity
Sensing
Acceleration, vital signs, and GPS
Classification
Evaluation of machine learning models.
Recognized activity
Visualization
Displaying vital signs, location, and activity in real time
Raw data
Feature set instances
Data preprocessing
Decoding, filtering, and time window construction
Decoded values
Time windows
Feature extraction
Statistical and structural features
Mobile application

Fig. 1. System architecture.

These data are decoded and organized in time windows by the *data preprocessing* component. Then, the *feature extraction* module extracts statistical and structural features from each time window, producing a feature set instance which is later evaluated by the *classification* module. The output of the classifier is indeed the recognized activity, displayed by the *visualization* module and sent to the server for further analysis and historical querying. When the system is working on training mode, all raw data are also sent to the server. Next, we will elaborate on the mobile application components.

A. Sensing devices

The system currently supports three sensing devices, namely the phone GPS, phone accelerometer, and the BioHarness^{TM} BT chest sensor strap, manufactured by Zephyr. The strap is unobtrusive, lightweight, and can be easily worn by any person. It allows for measuring the person's heart rate, respiration rate, breath amplitude, skin temperature, posture (i.e., inclination of the sensor), electrocardiogram amplitude, galvanic skin response, SpO_2, and 3D acceleration. The strap accelerometer records measurements at 50 Hz, within a $-3g$ to $3g$ range, where g stands for the acceleration of gravity. On the other hand, physiological signals are sampled at 1 Hz since they are not expected to change much in short periods of time.

B. Communication

The communication module encompasses three levels: (1) receiving raw data from the sensors (via Bluetooth), (2) sending raw data or activity results to the server (via TCP/IP), and (3) querying the database (via HTTP servlets). In the first level, three different types of packets are received from the sensing device: acceleration, vital signs, and electrocardiogram. In the second level, these packets are aggregated and sent to the application server, which decodes and stores them into the

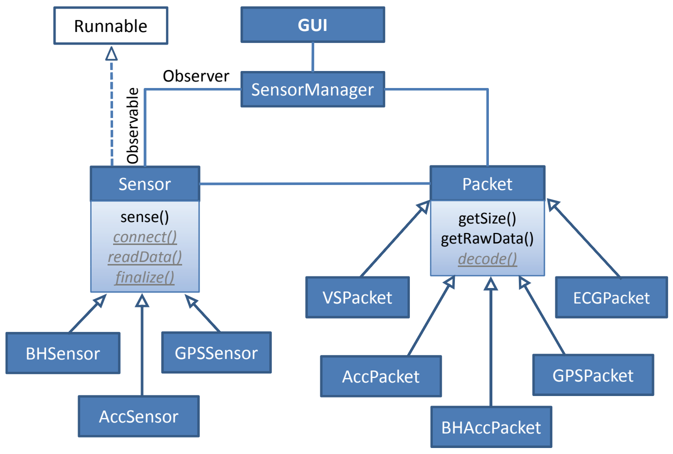


Fig. 2. Simplified UML class diagram for the sensing component.

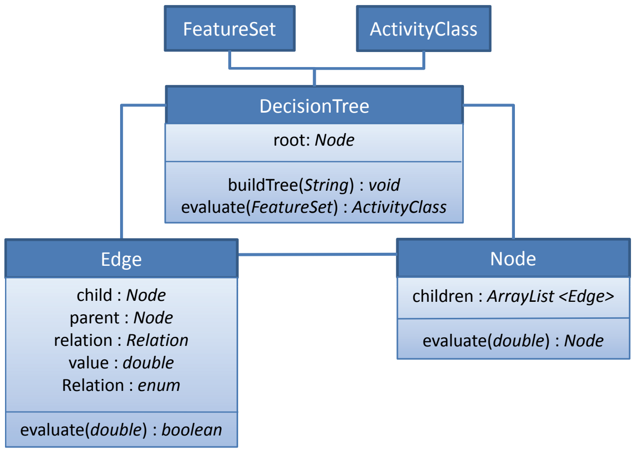


Fig. 3. Simplified UML class diagram for the classification component.

database. Finally, in the third level of communication, HTTP servlets allow for validating user credentials and querying the list of activities to be recognized, the list of features to be extracted, as well as the classification models to be used.

C. Sensing and data preprocessing

The sensing component manages and synchronizes the flow of data from all sensing devices, i.e., accelerometer, GPS, and the Bioharness strap. Figure 2 illustrates the interrelationship among the classes in this module. Sensors are represented as entities that extend from the abstract class *Sensor* and implement methods to *connect*, *read data*, and *finalize*. Additionally, they periodically report measurements to the *SensorManager* class through the Observer-Observable pattern [11]. With this model, new sensors can be easily incorporated to the system and they are able to work concurrently, as the class *Sensor* also implements the Java *Runnable* interface.

Every piece of raw data is represented as a packet, which might contain one or several samples, according to the sensor specifications. Each specific type of packet extends from the abstract *Packet* class, and implements a particular *decode* method. The *SensorManager* class receives all packets and controls the data flow to achieve the recognition of activities.

D. Feature extraction

This component provides an efficient implementation of statistical and structural feature extraction methods for nine attributes: heart rate, respiration rate, breath amplitude, skin temperature, posture, ECG amplitude, and 3D acceleration. The methods are briefly introduced below yet the interested reader might refer to [5], [6] for more details.

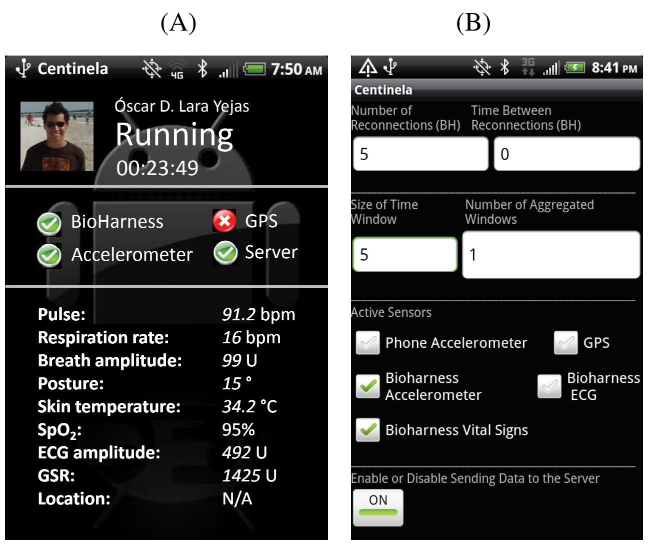
1) Statistical features: These are: *mean*, *variance*, *standard deviation*, *correlation between axes*, *interquartile range*, *mean absolute deviation*, *root mean square*, and *energy*. They were applied to acceleration signals (i.e., 24 features in total).

2) Structural features: Structure detectors use an arbitrary function f with a set of free parameters $\{a_0, ..., a_n\}$ to fit the points of a given time series S; these parameters are, in fact, the extracted features. This fitting process is done by means of the Least Squares algorithm. For each of the vital sign time windows, the goal is to find the function f^* with the smallest sum of squared error (SSE). In our previous study [5], we found that polynomial functions had the lowest SSE using degrees one, two, and three. Finally, we considered two more features, the *trend*, and the *magnitude of change*, intended to describe the behavior of the vital signs during intervals of vital sign stabilization. Structural features are applied to physiological signals (i.e., a total of 66 features).

E. Classification

Many classification algorithms have been employed in activity recognition: C4.5 [7], Fuzzy Logic [6], and Support Vector Machines [12], among others. Nevertheless, implementing them in a cellular phone turns out to be a hard task. Although there exist Java API's to train and evaluate classification models, such as WEKA [2] and JDM [3], these are not suitable for mobile phones. This is mainly because some classes and interfaces required by them, such as *java.lang.Cloneable*, are not available in the Android platform. Therefore, we wrote our own library, MECLA, to evaluate classification models on the phone. Our current implementation supports decision trees, thereby enabling the use of *C4.5*, *ID3*, *Decision Stump*, *Random Tree*, and *M5*, among other classification algorithms. We are currently working on integrating additional algorithms to the library. More on MECLA can be found in [13].

The design of the classification module is shown in Figure 3. Nodes correspond to features (e.g., mean acc. in X) whereas edges define *relations* of an attribute with a numeric or nominal *value* (e.g., mean acceleration in X is greater than 2.382g). The method *evaluate* of the class *Edge* returns whether or not a given value fulfills the edge's relational condition. This method is utilized by class *Node*'s *evaluate* method, which returns the next child node to be visited given a particular attribute value. In this way, the evaluation of a feature vector (i.e., an instance of the class *FeatureSet*) starts at the root node and follows the corresponding branches according to the output of the node evaluation methods. The process stops when the current node


(A)

Centinela 7:50 AM

Óscar D. Lara Yejas

Running

00:23:49

✓ BioHarness ✗ GPS

✓ Accelerometer ✓ Server

Pulse:	*91.2* bpm
Respiration rate:	*16* bpm
Breath amplitude:	*99* U
Posture:	*15* °
Skin temperature:	*34.2* °C
SpO_2:	95%
ECG amplitude:	*492* U
GSR:	*1425* U
Location:	N/A

(B)

8:41 PM

Centinela

Number of Reconnections (BH)	5
Time Between Reconnections (BH)	0
Size of Time Window	5
Number of Aggregated Windows	1

Active Sensors

- [ ] Phone Accelerometer
- [ ] GPS
- [✓] Bioharness Accelerometer
- [ ] Bioharness ECG
- [✓] Bioharness Vital Signs

Enable or Disable Sending Data to the Server

ON

Fig. 4. Mobile application user interface.

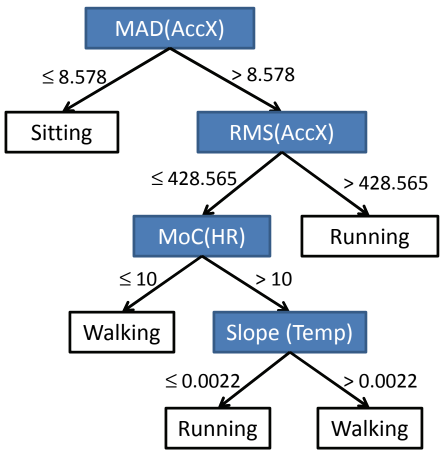


Fig. 5. The classification model generated by the C4.5 algorithm. Four features are part of the tree: (1) *MAD(AccX)*, i.e., the maximum absolute deviation of the acceleration in the X axis; (2) *RMS(Accx)*, i.e., the root mean square of the acceleration in the X axis, (3) *MoC(HR)*, i.e., the magnitude of change of the heart rate signal, and (4) *Slope(Temp)*, i.e., the slope of the line that best fits the skin temperature signal.

does not have any children, and the associated activity class (e.g., running, walking, etc.) is returned.

All classification models were priorly trained in WEKA using the C4.5 algorithm under different parameter configurations (i.e., window sizes and feature extraction methods). The alphanumeric representations of all decision trees, as given by the WEKA output, were stored in our database. An HTTP servlet allows the mobile application to query a decision tree model in accordance to the parameter values set by the user. We used a recursive algorithm to build a *DecisionTree* object (i.e., nodes and edges) based upon the alphanumeric representation retrieved by the servlet.

F. Feature selection

Even though the classification models were trained with a total of 90 features, only a subset of them are used. That depends on the window size, the enabled sensors, and the data themselves. Hence, the mobile application only extracts the features that are actually part of the decision tree. For this purpose, another HTTP servlet retrieves the list of features to be extracted given the window size and the active sensors.

G. Visualization

Figure 4 displays the two main screens, namely the *monitor* and the *configuration*. The former displays the sensed data (e.g., heart rate, respiration rate, skin temperature, etc), the current user's activity, and the data collection time in real-time. The latter, allows the administrator to set parameters such as the window size, number of aggregated packets, maximum number of Bluetooth reconnections, time between Bluetooth reconnections, and feature extraction methods. It also permits turning on and off each individual sensor, as well as enabling or disabling data transmission to the server.

IV. Evaluation

The mobile application was tested on an HTC Evo 4G mobile phone, being compatible with Android 2.1 or higher. The classification model (shown in Figure 5) was generated by the C4.5 algorithm using 5-second-long time windows with 50% overlap. In [5], the reader may find the data collection protocol and the physical characteristics of the individuals who collected training data. As a proof of concept, in the present study, three activities were considered, namely *running*, *walking*, and *sitting*. Two new users (a male and a female), who did not participate in the training phase, performed each activity in a sequential fashion during approximately 5 minutes. A total of 360 instances (i.e., time windows) were classified. The evaluation encompasses three main aspects: accuracy, response time, and energy consumption.

A. Accuracy

To avoid potential mistakes when labeling the data, the assessment of the classification accuracy was done programatically. In training mode, each user set the *real activity* as the ground truth, and performed said activity for certain amount of time. Meanwhile, the application computes the *classified activity* for each time window. With these data, the confusion matrix is calculated and displayed on the phone. The results are shown in Figure 6, where columns correspond to the real activity and rows represent the classified activity. Precision and recall were also calculated for each activity. The overall accuracy was 92.64%.

	Running	**Walking**	**Sitting**	**Precision**
Running	123	0	0	**1**
Walking	5	112	21	**0.8115942**
Sitting	0	1	105	**0.990566**
Recall	**0.96094**	**0.9915**	**0.8333**	

Fig. 6. Confusion matrix.

B. Response time

The response time was measured for each time window as the total time spent by (1) preprocessing, (2) feature extraction, and (3) classification. The reader may notice that the decision tree shown in Figure 5 is fairly small and only involves four features. In order to demonstrate the feasibility of the application under more challenging scenarios, we have also measured the response time for two larger decision trees that use different window sizes and feature sets. The average response times for all trees, after issuing 100 time windows, are shown in Table I.

TABLE I
AVERAGE RESPONSE TIME IN MILLISECONDS.

Window length	Features	Nodes	Total
5s	4	9	66.54 ms
5s	19	47	395.33 ms
12s	9	22	499.33 ms

The most demanding stage was the feature extraction which, on average, consumed 60% of the total response time. This was expected as the feature extraction algorithms run in $O(n)$, where the number of samples n can be either 250 or 600 for window lenghts of 5s and 12s, respectively. On the other hand, the classification stage only consumed about 12% of the time, as it runs in $O(\log m)$, where the number of nodes m is between 9 and 47. Hence, the window size and the number of features have more significant impact in the overall response time than the number of nodes. Note that, in the worst case, the response time is less than 8% of the window length, confirming that our system can be deployed in current cellular phones.

C. Energy consumption

Our hypothesis is that executing activity recognition locally in the phone —rather than sending all raw data to the server— is effective to save energy. In order to prove it, three cases were considered: (1) recognizing activities locally in the phone without sending raw data to the server; (2) sending all raw data to the server to recognize the activities remotely; and (3) not running the application to measure the energy consumed by the operating system and other phone services alone. The classes *Intent* and *BatteryManager* of the Android API were used to measure the battery charge difference after three continuous hours of use. Then, energy consumption was estimated given that the phone's battery works at 3.7 V and has a total charge of 1500 mAh. The results, shown in Table II, indicate that performing local activity recognition allows for increasing the application lifetime in 25%. More precisely, the system can run for up to 12.5 hours in case 1, versus 9.38 hours in case 2. Now, excluding the energy consumed by the operating system, the total energy savings in case 1 with respect to case 2 were roughly 26.7%. This result becomes remarkable as case 2 was already using data aggregation as an energy saving mechanism.

TABLE II
ESTIMATED ENERGY CONSUMPTION AFTER EXECUTING THE APPLICATION FOR THREE HOURS.

Case	Charge diff.	Current	Power	Energy
1	0.36 Ah (24%)	0.12 A	0.444 W	4795.2 J
2	0.48 Ah (32%)	0.16 A	0.592 W	6396.6 J
3	0.03 Ah (2%)	0.01 A	0.037 W	399.6 J

V. CONCLUSIONS

In this work, we present a mobile framework for real-time human activity recognition under the Android platform. The system features a library for mobile evaluation of classifiers (MECLA), which can be utilized in further machine learning applications. The evaluation shows that current cellular phones are more than capable to run our system. Besides, we found a substantial reduction in energy consumption (up to 26.7%) compared to a server-based HAR system. The accuracy of the C4.5 classifier was also satisfactory, validating the utilization of this algorithm in this and other applications.

VI. ACKNOWLEDGEMENTS

This work was partially funded by the National Science Foundation under grant No. 1062160. Special thanks to Elizabeth Rodriguez, Aaron Young, and Jose Cadena, who worked on the development of the mobile application, and Anamaria Bejarano for reviewing the manuscript.

REFERENCES

[1] F. Foerster, M. Smeja, and J. Fahrenberg, "Detection of posture and motion by accelerometry: a validation study in ambulatory monitoring," *Computers in Human Behavior*, vol. 15, pp. 571–583, September 1999.
[2] "Waikato Environment for Knowledge Analysis (WEKA)," http://www.cs.waikato.ac.nz/ml/weka/.
[3] "The Java Data Mining Platform," http://www.jdmp.org/.
[4] "Google's Android becomes the world's leading smart phone platform," http://www.canalys.com/newsroom/google%E2%80%99s-android-becomes-world%E2%80%99s-leading-smart-phone-platform.
[5] O. D. Lara, A. J. Perez, M. A. Labrador, and J. D. Posada, "Centinela: A human activity recognition system based on acceleration and vital sign data," *Pervasive and Mobile Computing*, vol. DOI information: 10.1016/j.pmcj.2011.06.004, 2011.
[6] Y.-P. Chen, J.-Y. Yang, S.-N. Liou, G.-Y. Lee, and J.-S. Wang, "Online classifier construction algorithm for human activity detection using a tri-axial accelerometer," *Applied Mathematics and Computation*, vol. 205, no. 2, pp. 849–860, 2008.
[7] J. Parkka, M. Ermes, P. Korpipaa, J. Mantyjarvi, J. Peltola, and I. Korhonen, "Activity classification using realistic data from wearable sensors," *IEEE Transactions on Information Technology in Biomedicine*, vol. 10, pp. 119 –128, jan. 2006.
[8] M. Berchtold, M. Budde, D. Gordon, H. Schmidtke, and M. Beigl, "Actiserv: Activity recognition service for mobile phones," in *Proceedings of the International Symposium on Wearable Computers*, pp. 1–8, 2010.
[9] T. Brezmes, J.-L. Gorricho, and J. Cotrina, "Activity recognition from accelerometer data on a mobile phone," in *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, vol. 5518, pp. 796–799, Springer, 2009.
[10] D. Riboni and C. Bettini, "Cosar: hybrid reasoning for context-aware activity recognition," *Personal and Ubiquitous Computing*, vol. 15, pp. 271–289, 2011.
[11] "The Observer-Observable pattern in Java," http://download.oracle.com/javase/6/docs/api/java/util/Observable.html.
[12] Z.-Y. He and L.-W. Jin, "Activity recognition from acceleration data using ar model representation and svm," in *Proceedings of International Conference on Machine Learning and Cybernetics*, vol. 4, pp. 2245–2250, 12-15 2008.
[13] "The MECLA website," http://www.csee.usf.edu/˜olarayej/MECLA.