# Online Human Activity Recognition on Smart Phones

Mustafa Kose, Ozlem Durmaz Incel, Cem Ersoy
Computer Engineering Department
Bogazici University, Istanbul, Turkey
{mustafa.kose, ozlem.durmaz, ersoy}@boun.edu.tr

## ABSTRACT
This paper analyzes the performance of different classification methods for *online* activity recognition on smart phones using the built-in accelerometers. First, we evaluate the performance of activity recognition using the Naïve Bayes classifier and next we utilize an improvement of Minimum Distance and K-Nearest Neighbor (KNN) classification algorithms, called Clustered KNN. For the purpose of online recognition, clustered KNN eliminates the computational complexity of KNN by creating clusters, i.e., smaller training sets for each activity and classification is performed based on these compact, reduced sets. We evaluate the performance of these classifiers on five test subjects for activities of walking, running, sitting and standing, and find that Naïve Bayes provides not satisfactory results whereas Clustered KNN gives promising results compared to the previous studies and even with the ones which consider offline classification.

## Categories and Subject Descriptors
I.5.2 **[Design Methodology]**: *Classifier design and evaluation*

## General Terms
Algorithms, Experimentation, Performance

## Keywords
Activity Recognition, Naïve Bayes, Clustered K-Nearest Neighbor, Smart Phones.

## 1. INTRODUCTION
Human activity recognition using sensory data has become an active field of research in the domain of pervasive and mobile computing. It involves the use of different sensing technologies to automatically collect and classify user activities for different application domains, ranging from medical applications, home monitoring & assisted living, sports & leisure applications. Initially, vision-based sensing, using cameras has been the focus of research studies and more recently inertial sensing, using movement based sensors that can be attached to the user's body has been investigated [10].

Motivated by the recent studies, activity recognition using the smart phones equipped with a rich set of embedded sensors, such as the accelerometer, GPS, microphone [1-9], has been introduced. Algorithms used in the classification of activities originate from statistical machine learning techniques. However, a trendy algorithm in machine learning research may not exhibit a superior performance in the field of activity recognition [12], especially on the mobile phone platform with limited resources, such as the limited processing power and battery. Moreover, when we look at the literature on activity recognition using inertial sensors, we see that most of the studies first collect sensory data and apply classification algorithms offline on the collected data, using a large part of the collected data for training[1]. It is clear that larger the amount of overlap between the training data and the testing data, better recognition results will be achieved. Offline processing exploits this advantage.

Offline processing can be used for applications where online recognition is not necessary. For instance, if we are interested in following the daily routine of a person, such as in [4], the sensors can collect the data during a day; the data can be uploaded to a server at the end of the day and can be processed offline for classification purposes. However, for applications such as a fitness coach where the user is given a program with a set of activities, their duration and sequence, we might be interested in what the user is currently doing [13]. Another application can be the recruitment for participatory sensing applications [11]. For instance, the application might be interested in collecting information from users that are currently "walking" in a particular part of a city. Therefore, online recognition of activities becomes important.

In this paper, we focus on activity recognition using the embedded accelerometers on smart phones. Our objective is the classification of basic movements of a user, such as walking, running, sitting and standing. As we mentioned, in contrast to the offline processing of data, we focus on online classification of these activities and evaluate the performance of classifiers such as Naïve Bayes. For this purpose, we developed an activity recognition system for Android phones. Besides online recognition of activities, the training phase is also performed on the phone instead of processing the data offline to learn the model parameters. It is known that training phase of a classification method used in activity recognition is costly and there is a general interest in providing an activity recognition system that does not need training [14] or requires only limited training by the end user. In this paper, we are interested in the performance of classifiers with limited training data considering the limited memory available on the phones. In the proposed system, training data can be collected only in a few minutes and can be used directly for classification steps which reduce the burden on the users. Being one of the first Android applications used for activity recognition is another important motivation for this study. Another contribution of our paper is that the performance evaluation is carried out by using several smart phone models with different capabilities instead of focusing on one model.

Besides the performance of classifiers such as Naïve Bayes, we also take advantage of a classification scheme, called Clustered


*2nd International Workshop on Mobile Sensing, April 16, 2012, Beijing, China.*


[1] We consider supervised learning methods thus the models require labeled training data to learn the model parameters

KNN, which is an improvement of minimum Distance and k-nearest neighbor (KNN) classification algorithms that works in real-time. Performance of these classifiers is tested on five different subjects. Test results show that the clustered KNN approach outperforms other classifiers in terms accuracy and execution time. The impact of sensor sampling rate and window size, which is used for segmenting the data, on the performance of activity recognition is also analyzed.

The rest of the paper is organized as follows. In Section 2 we present the related work. In Section 3, we will detail clustered KNN method and the implementation details of the activity recognition application will be provided. Section 4 presents the experiments and the results. Finally in Section 5, future work directions will be given together with conclusions.

# 2. RELATED WORK ON ACTIVITY RECOGNITION WITH SMART PHONES

In [1], a novel design framework for energy efficient mobile sensing system (EEMSS) is proposed. A hierarchical sensor management strategy is used to recognize user states as well as to detect state transitions. User states may contain a combination of features such as motion (running, walking), location (staying at home or on a freeway) and background condition (loud or quiet) which all together describe user's current context. State transition system implemented on Nokia N95 can define the following states: walking, vehicle, resting, home talking, home entertaining, working, meeting, office loud, place quiet, place speech, place loud. The sensors being used for activity recognition are accelerometer, Wifi detector, GPS and microphone. EEMSS is able to detect states with approximately 92.5% accuracy and improves battery lifetime by over 75% compared to an existing application Cenceme [5].

In a similar study, Reddy et. al proposed a different model for activity recognition [2]. In this study, they designed, implemented and evaluated transportation mode classification system which runs on a mobile phone by using 3-axis accelerometer and GPS sensors. They focused directly on outdoor activities and classified them into walking, stationary, biking, running, and motorized transport. The performance of different classification algorithms is evaluated. According to the evaluation steps, decision tree classification followed by Hidden Markov Model provided the best results. Weka Machine Learning Toolkit (WMLT) and Generalized HMM library is employed to implement the classification steps on Nokia N95. Overall, the system provides higher than 93% accuracy.

Different from the previous studies, in [3], a classification algorithm is aimed to run on iPhone. For this purpose, three iPhone applications are developed called as iLog, iModel, iClassify respectively. iLog can be used for data collection for different activities in real time. iModel is a desktop tool for learning and testing models. Data saved by iLog can be imported into iModel which is a Java application built on Weka Machine Learning Toolkit. By using iModel, labeled data can be used to test an existing model or to learn a new model. Lastly, iClassify can be used to classify activities in real time on iPhone platform. In terms of the methodology and execution time of classification steps, this study differs from the previous two studies. There is an offline data processing step to learn data models and addition to iPhone 3-axis accelerometer sensor, Nike+iPod Sports Kit is used to collect data which imports collected data periodically to iPhone via Bluetooth connection. iClassify can report activity classifications once per second. Activities are classified into running, walking, biking, and sitting states by using Naïve Bayes method with nearly 97% accuracy.

The studies in [4], [5], and [6] have similarities with the studies stated above in terms of methodology and ideas. The most common sensor used in these studies is the accelerometer sensor. Wifi, GPS or any other sensors are added to strengthen the sensing power and accuracy of the results. Sitting, Standing, Walking, Running, Driving, and Biking are the main activities being targeted to be recognized in the applications. Additionally commercially-targeted applications include more activities to be recognized as in [5]. Only a few of the algorithms have online classification capabilities and most of them (except [3]) are implemented on Nokia N95 which is one of the first mobile phones with data sensing capabilities. Most of the classification algorithms benefit from WMLT and none of the studies consider the training phase to be performed on the phone.

# 3. ACTIVITY RECOGNITION WITH CLUSTERED KNN

In this section first we introduce the clustered KNN method and next we explain the implementation details of the activity recognition application on the Android platform.

## 3.1 Clustered KNN

In the literature, it has been reported that minimum distance classifier does not perform well when used alone [2]. Additionally, KNN results are always better than minimum distance in terms of accuracy. However, KNN is not an online classifier since it requires high computational burden and especially considering the limited resources on smartphones, it does not appear as a preferable method.

Considering these mentioned facts, we propose to combine the advantages of the two classification methods and offer clustered KNN for online classification. According to the new method, called clustered KNN, training data is first preprocessed and four features, which are *average*, *minimum*, *maximum*, and *standard deviation*, are extracted and in the second step classification takes place.

### *3.1.1 Preprocessing in Clustered KNN*

In the preprocessing step, our objective is to define activity sets from the training data based on the mentioned features. For all feature sets, we use the magnitude of acceleration values. For the training data sets we do not have any limitation in terms of the sample count. Additionally, we aim to decrease the burden of comparisons with training set generated within the preprocessing step for an online classifier. By this way, instead of comparing all the data in the training set, we compare the test data only with the compact training data set that we selected from the original training set.

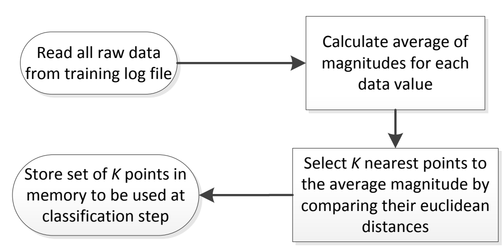


**Figure 1 Preprocessing for the "average" feature**

During the preprocessing step, compact training sets are created for each feature and for each activity. For each feature, except the standard deviation, $K$ data points are selected from the training data. This process is summarized in Figure 1 for the "average" feature and just for one activity. For instance, for the minimum

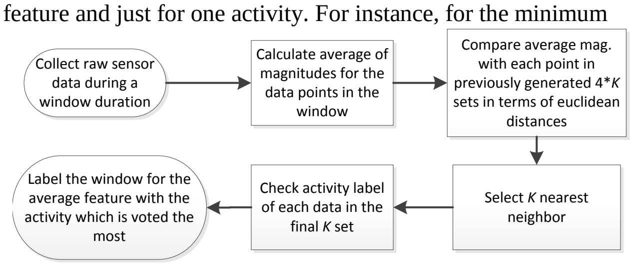


**Figure 2 Classification process for "average" feature**

feature set, *K* "minimum" data points are selected from the training data. Similarly, we create a "maximum" set by selecting the *K* maximum data points. The average value of the training data is calculated and nearest *K* data points are included in the "average" set. For the "standart deviation set", standard deviation of training values for each activity is calculated. As an example, if *K* is selected as 10, we have in total 40 samples after the preprocessing step for four activities per feature except the standard deviation. *K* value is an important system parameter and when *K* is smaller computational complexity and classification execution times decrease. However, at the same time accuracy of the results are expected to decrease with smaller *K*, so that there is an important tradeoff between accuracy and execution time considering the value for *K*. Impact of the *K* value both on classification accuracy and computational complexity is analyzed in Section 4.

### *3.1.2 Classification in Clustered KNN*

In the classification step, during a window with a predefined size we collect test data, in other words we segment the data[2]. After the window is filled, classification starts, and average, minimum, maximum, standard deviation values of the data in the window is calculated. These values are compared one by one with the values in the compact training sets which were created during the preprocessing step. *K* nearest sample to test data is selected from training sets and voting is made by looking at the final list of activities. We label the data in the related window as the activity for which we have maximum amount of data in the final *K* set. For instance, if *K* is 10 and the final list is as "1 1 5 3" (1 vote for running, 1 vote for walking, 5 votes for sitting and 3 votes for standing) for the average feature, then the activity is labeled as sitting according to the average feature. This process is summarized below in Figure 2 for the average feature. Same process is applied for minimum and maximum data sets as well. We make one last comparison for the standard deviation coming from the related window with the standard deviation of each training sets for different activities. The one which is closer to the standard deviation of that window is selected as the recognized activity by the standard deviation feature. At the end, we have four labels coming from voting results of each feature. We label the window as the activity for which we have the highest vote and finalize the classification.

[2] Same segmentation procedure is also applied to other classifiers whose performance is evaluated in Section 4.

## 3.2 Activity Recognition Application on Android Platform

The clustered KNN classifier and the other classifiers are implemented on Android phones to detect four main activities; which are walking, running, standing and sitting. For this purpose, the process is divided into two phases.

At first stage, training data is collected for each activity separately. For this purpose we developed an application called Activity Logger. In this application, user selects the activity to be performed, puts the phone into the packet and starts to perform the related activity. For each activity, the application creates different training data files in which raw data from the 3-axes of the accelerometer is being logged. Low pass filter is applied to raw data for noise removal. Before starting the activity recognition tests, a few minutes of training data for each activity is collected by each subject.

In the second stage, activity recognition is performed using the selected classifier. First, the application extracts necessary features of training sets for each activity according to the classification method being used. Depending on the size of the training set and the processor performance of the phone, this step may take a few minutes. The main screen of the application allows the user to select the system parameters, such as the sensor sampling rate and window size. The user interface of the application is presented in Figure 3.

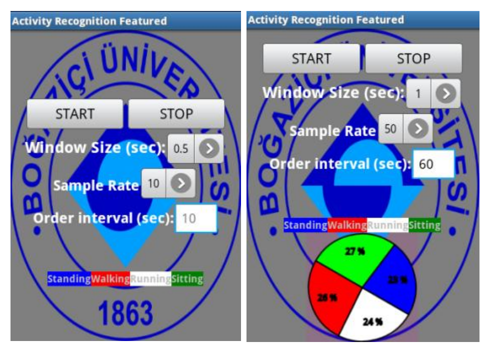


**Figure 3 Interface of Activity Classifier**

In order to monitor the recognition performance of the classifier, the ground truth data, i.e. which activity is actually performed by the user, is logged. For this purpose, the application gives voice commands repeatedly to perform an activity. Activity order is predefined in the system whereas activity duration "order interval" is given directly as the user input to the system in unit of seconds. During our experiments each activity is performed for 60 seconds for one cycle. Finally, using these ground truth values, i.e., activity tags, activity recognition performance and other performance metrics of the classifiers are calculated.

# 4. ACTIVITY RECOGNITION TESTS AND PERFORMANCE EVALUATION

## 4.1 Experiment Design

Tests are performed with five volunteer subjects whose average age is 27 (1 female, 4 male). All subjects carried the mobile phone in the front pocket of their pants during both test and training phase. Each subject performed the same predefined activity pattern "running, walking, sitting, standing" during the classification step. Each experiment lasts 4 minutes where each activity is performed for 60 seconds. The same test scenario is repeated 9 times with different system parameters based on window size and sampling rate. Additionally, *K* value is varied as 10, 50, and 100 for clustered KNN tests which resulted in 27 tests in total. Window sizes are selected as 0.5, 1, 2 seconds whereas 10, 50, and 100 msec is used for the sampling interval. With these

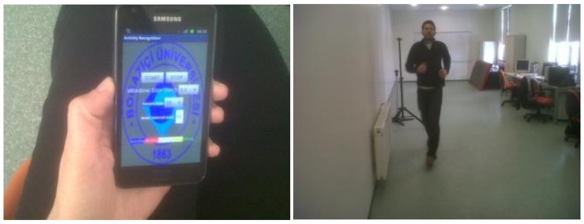

**Figure 4 Snapshots from the Experiments**

changes, our objective was to examine the effect of window size and sampling interval on the performance of activity recognition. All tests are performed on Android mobile phones. Phone models are Samsung SII Galaxy, HTC Desire Z, Samsung Galaxy Gio and T-Mobile G2 Touch (HTC Hero). Each subject performed all the tests on the same platform. At the end of the classification process, precision, recall, accuracy and f-measure metrics are calculated and results are written in a final result file. Classification results are also shown with a graphical pie chart on the phone screen (Figure 3).

## 4.2 Classification Results

### *4.2.1 Performance of Naïve Bayes*

We observed that accuracy rates ranged from 37% to 54% when we apply Naïve Bayes classification method which is highly dependent on the system parameters. Naïve Bayes achieved a 48% average accuracy rate for all subjects with different sampling rates and window sizes, as shown in Table 1. Similarly, performance in terms of F-measure was close to 47%. Moreover, best performance results are obtained with window size of 1 second regardless of the effect of the sampling rates.

**Table 1 Comparison of Clustered KNN and Naive Bayes**

| | Clustered KNN | Naïve Bayes |
|---|---|---|
| Accuracy | 92.13% | 47.61% |
| Precision | 92.45% | 51.15% |
| Recall | 92.09% | 42.30% |
| F-measure | 92.27% | 46.19% |

### *4.2.2 Performance of Clustered KNN*

As mentioned previously, *K*, window size and the sampling interval is determined as system parameters for the clustered KNN method. Classification performance of clustered KNN is also presented in Table 1. Compared to Naïve Bayes, on average, clustered KNN achieved a much better classification performance, around 92% accuracy, precision, recall and F-measure. Test results show that accuracy rates for online classification with Clustered KNN method are highly comparable with the previous studies referenced in this paper and even with the ones which are considering offline classification.

The confusion matrix for clustered KNN is presented in Table 2 to evaluate the classification performance of clustered KNN for each activity. Compared to the performance of activities of running, standing and sitting, the classifier presents slightly worse performance for walking where it is sometimes classified as running or standing. However, the overall performance for clustered KNN is around 92% accuracy considering all activities.

We also evaluated the impact of *K* value on the classification performance of clustered KNN. As expected, increasing the *K* value affected accuracy rates positively. We observed on average 87% accuracy with *K*=10 whereas it increased to 91% when *K* is

**Table 2 Overall Confusion Matrix**

| | | Classification | | | |
|---|---|---|---|---|---|
| | | Running | Walking | Standing | Sitting |
| Truth | Running | 1020 | 46 | 14 | 7 |
| | Walking | 178 | 729 | 146 | 7 |
| | Standing | 121 | 12 | 1063 | 18 |
| | Sitting | 18 | 2 | 30 | 1084 |

selected as 50. Although accuracy rates are not affected from further increase of *K* values, classification times significantly increase because of this change. Table 3 summarizes the performance of clustered KNN in terms of average accuracy rates with changing *K* values.

**Table 3 Average Accuracy Rates (%)**

| Window size(sec) | | 0.5 | | | 1 | | | 2 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Sampling Interval(msec) | | 10 | 50 | 100 | 10 | 50 | 100 | 10 | 50 | 100 |
| K | 10 | 87.9 | 87.8 | 87.7 | 88.4 | 90.3 | 89.5 | 88.6 | 87.8 | 89.3 |
| | 50 | 91.1 | 90.0 | 91.4 | 91.9 | 92.1 | 90.8 | 88.9 | 89.4 | 91.0 |

Looking at all the performance results, we observe the best accuracy and execution time results with *K* value selected as 50. We further analyzed the effect of sampling rate and window size on accuracies. In general, worst results are observed in cases when window size is selected as 2 seconds whereas best results are obtained with window size of 1 second. For the window sizes studied, as shown in Table 3, sampling interval does not have a significant impact on the accuracy results. When we consider the overall effect of all system parameters we obtained best results in the case where *K* is selected as 50, window size is selected as 1 second and sampling interval is selected as 50 msec. According to the tests performed with five different subjects, we obtained average 92% accuracy rate for this case. Table 3 summarizes the accuracy results with changing parameter values.

## 4.3 Execution Time for Clustered KNN

Besides the classification performance, we also evaluated the performance of clustered KNN in terms of execution times. As expected, classification execution times are considerably reduced as *K* parameter is decreased. Moreover, classification times are highly dependent on the device model and capabilities as well.

**Table 4 CPU & Memory Usage for Activity Classifier and other applications on Samsung Galaxy**

| Activity Classifier | | |
|---|---|---|
| | CPU Usage | Memory Usage |
| KNN Clustered | 29% | 21.9 MB |
| Naïve Bayes | 42% | 12.6 MB |
| Min Distance | 27% | 19.6 MB |
| TTS Service | 13% | 12.4 MB |
| Benchmark Applications | | |
| | CPU Usage | Memory Usage |
| System | 10% | 28.8 MB |

| | | |
|---|---|---|
| Norton Mobile | 4% | 19.4 MB |
| Internet | 2% | 37.0 MB |
| Google Maps | 1% | 31.2 MB |

Tests performed on Samsung Galaxy SII showed that classification times are increased up to 300 msec whereas it decreased to 50 msec when *K* is selected as 10. File management and logging times are also included in these results as well. On the other hand, execution times on the less capable T-Mobile G2 Touch varied between 200 and 900 msec for different parameters. The CPU capability is important: Samsung Galaxy SII has a 1.2 GHz dual-core system on a chip (SoC) processor whereas T-Mobile G2 Touch has a 528 MHz ARM11 processor.

## 4.4 Resource Consumption of Clustered KNN

The first part of Table 4 summarizes the CPU and memory usage of the activity classifier with different classification methods. All measurements are taken from Samsung Galaxy Gio. Resource usage of activity classifier highly depends on the classification methods being used at runtime.

According to the results, CPU and memory usage never exceeded 42% and 22MB respectively. Applications using minimum distance classifier and clustered KNN consume nearly the same amount of resources. On the other hand, Naïve Bayes has considerably higher CPU usage. Text to Speech (TTS) Service, which is being used for voice commands to guide the user about which activity to be performed during runtime, is also included in the table for comparisons as well.

For benchmarking, resource usage values for common applications are presented in the second part of Table 4. According to these results, the performance of activity classifier application is comparable with frequently used applications in terms of resource efficiency.

# 5. CONCLUSION

In the literature, there are only a few works on online activity recognition using the sensors on smart phones. In this paper, we proposed an activity recognition system working on Android platforms that supports online training and classification while using only the accelerometer data for classification. Online classification performance of Naïve Bayes classifier is evaluated and a clustered KNN method is used. The clustered KNN method exhibited a much better performance than the Naïve Bayes classifier in terms of accuracy on mobile platforms with limited resources. These results reveal that compared to the previous studies and even with the ones which are considering offline classification, clustered KNN provides promising results. As a future work, we will implement decision tree classification method on Android platform. More activities, such as biking and transportation with a vehicle, will be added to the classification states. Moreover, the impact of using different feature sets with the clustered KNN method will be analyzed.

# 6. ACKNOWLEDGMENTS

This work is supported by the Turkish State Planning Organization (DPT) under the TAM Project, number 2007K120610 and by the Bogazici University Research Fund under the grant agreement number ”6370” and ”6056”.

# 7. REFERENCES

[1] Yi Wang, Jialiu Lin, Murali Annavaram, Quinn A. Jacobson, Jason Hong, Bhaskar Krishnamachari, Norman Sadeh. A Framework of Energy Efficient Mobile Sensing for Automatic User State Recognition. ACM/Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services, 2009.

[2] Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin, Mark Hansen, Mani Srivastava. Using Mobile Phones to Determine Tranportation Modes. ACM Transactions on Sensor Networks, Vol. 6, No. 2, Article 13, February 2010.

[3] T. Scott Saponas, Jonathan Lester, Jon Froehlich, James Fogarty, James Landay. iLearn on the iPhone: Real-Time Human Activtity Classification on Commodity Mobile Phones. UW-CSE-08-04--02 Tech Report, 2008.

[4] Jun Yang. 2009. Toward physical activity diary: motion recognition using simple acceleration features with mobile phones. In *Proceedings of the 1st international workshop on Interactive multimedia for consumer electronics* (IMCE '09). ACM, New York, NY, USA, 1-10.

[5] Emilliano Miluzzo, Nicholas D. Lane, Kristof Fodor, Ronald Peterson, Hong Lu, Mirco Musolesi, Shane B. Eisenman, Xiao Zheng, Andrew T. Campbell. Sensing Meets Mobile Social Networks: The Design, Implementation and Evaluation of the CenceMe Application. SenSys '08 Proceedings of the 6th ACM conference on Embedded network sensor systems, 2008.

[6] Martin Mladenov, Michael Mock. A Step Counter for Java – Enabled Devices Using a Built-in Accelerometer. CAMS '09 Proceedings of the 1st International Workshop on Context-Aware Middleware and Services: affiliated with the 4th International Conference on Communication System Software and Middleware, 2009.

[7] Akos Fabian, Norbert Gyorbiro, Gergely Homanyi. Activity Recognition System For Mobile Phones Using The MotionBand Device. MOBILWARE '08 Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications, 2008.

[8] G.R.Yavuz, M.E.Kocak, G.Ergun, H.Alemdar, H.Yalcin, O.Durmaz Incel, C.Ersoy. A Smartphone Based Fall detector with Online Location Support, in Proceedings of PhoneSense 2010, Zurich, 2010.

[9] Nicholas D. Lane, Mashfiqui Mohammod, Mu Lin, Xiaochao Yang, Hong Lu, Shadid Ali, Afsaneh Doryab, Ethan Berke, Tanzeem Choudhury, Andrew Campbell. BeWell: A Smartphone Application to Monitor, Model and Promote Wellbeing, in Pervasive Health 2011, Dublin, 23-26 May 2011.

[10] Avci, A. and Bosch, S. and Marin-Perianu, M. and Marin-Perianu, R.S. and Havinga, P.J.M. (2010) Activity Recognition Using Inertial Sensing for Healthcare, Wellbeing and Sports Applications: A Survey. In: 23th International Conference on Architecture of Computing Systems, ARCS 2010, 22-23 Feb 2010, pp. 167-176.

[11] S. Reddy, D. Estrin, M. Srivastava. Recruitment Framework for Participatory Sensing Data Collections, in proceedings of the International Conference on Pervasive Computing (Pervasive), May 2010.

[12] Ulf Blanke, Diane Larlus, Kristof Van Laerhoven, Bernt Schiele, Standing on the Shoulders of Other Researchers - A Position Statement, in Pervasive Workshop "How to do good activity recognition research? Experimental methodologies, evaluation metrics, and reproducibility issues" 2010, Helsinki, Finland.

[13] E. Munguia Tapia. "Using Machine Learning for Real-time Activity Recognition and Estimation of Energy Expenditure". Ph.D. Thesis, Massachusetts Institute of Technology, 2008.

[14] Zachary Fitz-Walter and Dian Tjondronegoro. 2009. Simple classification of walking activities using commodity smart phones. In *Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group: Design: Open 24/7* (OZCHI '09). ACM, New York, NY, USA, 409-412.