Multithreaded Signal Preprocessing Approach for Inertial Sensors of Smartphone

Sahak Kaghyan
Institute for Informatics and Automation Problems of NAS RA, Armenia
sahak.kaghyan@gmail.com

Hakob Sarukhanyan
Institute for Informatics and Automation Problems of NAS RA, Armenia
hakop@ipia.sci.am

ABSTRACT
Cell phones and other mobile devices become the part of human culture and change activity and lifestyle patterns. Mobile phone technology continuously evolves and incorporates more and more sensors for enabling advanced applications. Latest generations of smart phones incorporate GPS and WLAN location finding modules, vision cameras, microphones, accelerometers, temperature sensors etc. The availability of these sensors in mass-market communication devices creates exciting new opportunities for data mining applications. Particularly healthcare applications exploiting build-in sensors are very promising. This chapter reviews different aspects of human activity recognition, including review of state-of-the-art, implementation and algorithmic aspects.

Keywords
Activity recognition, signal preprocessing, smartphone sensors, mobile computing, feature extraction

1. INTRODUCTION
With the advent of miniaturized sensing technology, which can be body-worn or integrated in mobile devices, it is now possible to collect, store and process data on different aspects of human physical activity. This data can enable automated activity profiling systems to generate activity patterns over extended periods of time for health monitoring. Collection of activity patterns is dependent on recognition algorithms that may efficiently interpret body-worn sensor data.

Existing activity recognition systems are constrained by practical limitations such as the number, location and nature of used sensors. Other issues include ease of deployment, maintenance, costs, and the ability to perform daily activities unimpeded. Sensors' outputs might vary for the same activity across different subjects and even for the same individual. Errors can also arise due to variability in sensor signals caused by differences in sensor orientation, placement, and from environmental factors such as temperature sensitivity. Three main classes of activity recognition are reviewed including coarse location tracking, video stream analysis and inertial navigation systems (INS) such as accelerometers. Sensor data are typically communicated from sensors to servers for data processing. Alternatively, signal processing can be performed in mobile devices such as smart-phones.

Collected sensor data are analyzed using data mining and machine learning techniques to build activity models and perform pattern recognition [10], [11]. Recognizing a predefined set of activities is a recognition (classification) task: features are extracted from the space-time information collected by sensors and then used for classification. Feature representations are used to map the data to another representation space with the intention of making the classification problem easier to solve. In most cases, a model of classification is used that relates the activity to sensor patterns. The learning of such models is usually done in a supervised manner (human labeling) and requires a large annotated datasets recorded in different settings.

In general, activity recognition algorithms can be divided into two major categories [9]. The first one is based on supervised and unsupervised machine learning methods. Supervised learning requires the use of labeled data upon which an algorithm is trained. Unsupervised learning is based on unlabeled data and applies the following steps: (1) acquire unlabeled sensor data, (2) aggregate and transform them into features; (3) model data by e.g. clustering techniques. The second broad category exploits logical modeling and reasoning. The steps are the following: (1) use a logical formalism to explicitly define and describe a library of activity models, (2) aggregate and transform sensor data into logical terms, and (3) perform logical reasoning based on observed actions, which could explain the observations.

2. SPLITTING ACTIVITY RECOGNITION TASKS FOR MOBILE COMPUTING

The review paper [1] elaborated sensors placed both externally or internally with respect to mobile devices. The research [2] focuses on activity recognition chain when exploiting internal sensors. An Android OS based mobile application was developed to study the efficiency of splitting activity recognition tasks between the mobile device and the server accounting for the trade-offs between accuracy of classification, signal reading rates and estimated battery power. The next subsection will review the specific aspects of mobile computing environment in application to the studied recognition task, while the next section describes in detail the classification flow and the experimental results.

2.1. Multithreading layered mobile computing

Here a multilayered architecture is described for implementing activity classification. Main idea of data collecting and further processing flow is illustrated in Fig. 1. The diagram describes the sequence of steps, starting from raw sensor signal acquisition to either training or recognition of a given signal block.

Preliminary setup. The signal processing chain is divided into several layers. Before starting the signal processing thread, the user can manually setup some information that is needed in further data processing. The setup stage is mostly necessary to distinguish "training" modes of the application. For example, to improve the training process, the user can specify the part of his body where the mobile device is attached. In addition, the user specifies the signal retrieving duration and reading rate. There are two options to specify "read-rate": a) Android OS provides four predefined rates, from lower to higher sampling frequency: Normal, UI, Game, and Fastest, and b) one can specify own rate in microseconds (for example, 35000mcrs correspond to 106/35000=28.6Hz frequency).

Main thread (Thread 1). First layer is represented by the main thread, which communicates with the user, so called "user interface" or UI thread (Fig. 2). When the user starts data processing, the application loads three more threads per sensor for data acquisition and analyzing. Therefore, for two sensors the application will initialize six more threads for simultaneous processing. All signal processing threads work in pairs, in a producer-consumer mode. Each pair is connected with one (except the 3-rd thread) data queue. Each data queue holds specific objects, corresponding to particular sensor [2].

There are two queues for processing: raw signals' data holder queue (RSDQ) and preprocessed signals' data holder queue (PSDQ).

Raw signals collector thread (Thread 2). Thread is responsible for raw signals collection from a sensor of the mobile device. Each sensor returns a signal of a certain format. For example, signal from accelerometer returns accelerations along three axes (x, y, and z). The thread collects raw signals and constructs an object, which contains accelerations along each axis, and sample collection time stamp (microsecond tick). The thread sequentially stores the constructed objects in RSDQ.

Signal preprocessor thread (Thread 3). The whole processing chain is divided into several layers. Third thread reads data from RSDQ. After initial specified delay the thread consecutively reads chunks of data, which is calculated by $(\Delta p \cdot 1000)/\Delta t$ formula, where Δp is the window length in seconds, and Δt is the sampling period in microseconds. Data preprocessing stage includes two operations: noise reduction and feature extraction. The thread constructs feature vectors and stores in the second queue – PSDQ. Preprocessing stage is very important, as it effects on further recognition accuracy.

Preprocessed data executor thread (Thread 4). When all preprocessing operations are completed, the application stores constructed feature vector into PSDQ data structure. In addition, the last thread from the chain starts final data execution. The final thread that completes all preprocessed data is the classification thread. This thread works with PSDQ data structure. The thread works continuously and stops only when period duration is over and there are no more feature vectors available in the queues for processing. The feature vector processing depends on the performing activity behavior. Efficiencies of the following two strategies are compared: (a) mobile-side data processing and (b) server-side data processing.

Mobile-side data processing. Mobile-side data processing is performed autonomously in the mobile device without involving external servers. Therefore, the mobile device saves energy due to mobile-server communication, such as Wi-Fi link, which is selected for the experiments. We conditionally divide the mechanism of acquired signals processing into two parts: "training" data processing mechanism and "classification" of unknown data. The mobile-based processing constraints training data sizes and complexity levels of classification algorithms. Computationally complex algorithms (SVM, HMM, etc.) are more feasible to implement on servers. In addition, for local recognition and data storage one can efficiently use SQLite portable mobile database and its features. SQLite supports multiple connections for parallel data writing and reading threads enabling more optimal computations of advanced algorithms.

Server-side data processing. Unlike "mobile-side data processing" approach, the server-side processing approach has significant advantages in following aspects:

- Servers usually have multiple (>4) supporting cores (unlike mobile device manufacturers who usually equip their phones with 2 or 4 cores);
- Servers usually have at least 32GB of random access memory (RAM).

Accessible physical memory size can be at least 100GB.

Thus, the "heavy" computations related to sophisticated classification algorithms can be performed on the server side. One should also mention a drawback of server-based approaches related to the necessity of supporting a mobile-server communication link, which consumes battery power. If this communication link is available, the extracted feature data can be sent to the server for further processing and feature vector object will be removed from smartphone memory. Here the processing stage depends on user intentions - training dataset accumulation or unknown activity classification.

For both unknown feature classification approaches (classification on smartphone and classification on server), it comes out that application recognizes movement activity within frame periods. In both cases, the application tracks recognized activities, and then counts the corresponding labels ("standing", "sitting", etc.) for each type of moves and then displays average activity, performed during the whole time. Other option to display recognized activities within the given time period is the so called "activity chain". In this case, sequential classification results that describe the same action are merging and result activity becomes a chain of series of primitive activities (for example, "standing"-"walking"-"running"-"standing"-"sitting").

3. ADVANCED ALGORITHMS FOR ACTIVITY CLASSIFICATION BY MOBILE DEVICES

Depending on the algorithm, motion-tracking signals can go through different preprocessing stages. For example, when dealing with accelerometer, the three axes can be individually median filtered to remove spikes as in [11]. A fifth order FIR filter with cut-off at .5 radians/sec was used instead in [16]. After filtering, the effect due to gravity can be subtracted [10, 12]. A third-order elliptic low pass IIR filter with cutoff at 0.25Hz was used to separate gravity in [13]. An additional preprocessing step where the mean is subtracted and the signal is normalized by the standard deviation can be seen in [14].

4. PERFOMANCE TESTING AND EFFICIENCY COMPARING

The two most popular ways of testing these types of human activity recognition algorithms are leave-one-subject-out validation and personalized validation. In the leave-one-subject-out validation the algorithm is tested on one subject but trained on the rest of the subjects in the dataset. In personalized validation, the algorithm is cross-validated by training and testing on a single subject [6], [5], and [15]. The results in [6], [5] seem to indicate that the performance of personalized systems is higher but leave-one-user-out validation is still important because systems that are previously trained on a set of subjects other than the user may be more convenient for deployment [8]. In other cases all frames to be classified are mixed and the algorithm is tested with folded cross-validation [5], [16], [17].

Performance evaluation for smartphone-based methods may also be different from attachable accelerometer based methods. In cases where the purpose of the algorithm is to recognize activities independently of the orientation of the device, it may prove useful to test the algorithm with a leave-one-orientation-in validation, which means that the algorithm is trained for one orientation but tested on several orientations as in [16]. However, these algorithms can also be tested by acquiring data with random orientations or orientations chosen by the subject and then performing personalized and leave-one-user-out validation with and without the orientation-invariant signal preprocessing to find out if there is any difference in performance between the two.

5. CONCLUSION

This work addresses an important direction of human activity recognition and processing in mobile environments. First, a review of the state-of-the-art is provided which is then followed by implementation aspects in mobile devices and introduction to advanced algorithms. The research area is rapidly developing and projects to commercialization judging by many products offered nowadays. Still there are many problems that persist on improving activity recognition rates, optimizing implementation techniques, and using universal algorithms, which are invariant to different placement of mobile devices.

REFERENCES

[1] S. Kaghyan, H. Sarukhanyan, D. Akopian, “Human Movement Activity Classification Approaches that use Wearable Sensors and Mobile Devices”, IS&T/SPIE Electronic imaging symposium, Conference on Multimedia and Mobile Devices, vol. 8667, Burlingame, CA, USA, 2013.

[2] S. Kaghyan, D. Akopian, H. Sarukhanyan, “Platform-dependent optimization considerations for mHealth applications”, IS&T/SPIE Electronic imaging symposium, Proceedings of SPIE Conference on “Mobile Devices and Multimedia: Enabling Technologies, Algorithms, and Applications”, 2015.

[3] J.A. Morales, D. Akopian, S. Agaian, “Human activity recognition by smartphones regardless of device orientation”, Proceedings of SPIE Conference on Mobile Devices and Multimedia: Enabling Technologies, Altorithms, and Applications, 2014.

[4] M. Oner, J.A. Pulcifer-Stump, P Seeling, and T. Kaya, “Towards the Run and Walk Activity Classification through Step Detection – An Android Application”, 34th Annual International Conference of the Engineering in Medicine and Biology Society, San Diego, 2012.

[5] N. Ravi, N. Dandekar, P. Mysore, M.L. Littman, “Activity recognition from accelerometer data”, Proceedings of the 17th conference on Innovative applications of artificial intelligence - Volume 3 (IAAI'05), Bruce Porter (Ed.), Vol. 3. AAAI Press, pp. 1541-1546, 2005.

[6] L. Bao, S. Intille, “Activity Recognition from User-Annotated Acceleration Data. Lecture Notes Computer Science 3001”, pp. 1-17, 2004.

[7] S. Kaghyan, H. Sarukhanyan, “Activity Recognition Using K-Nearest Neighbor Algorithm on Smartphone with Tri-axial Accelerometer”, International Journal of Informatics Models and Analysis (IJIMA), ITHEA International Scientific Society, Bulgaria, pp. 146-156, 2012.

[8] S. Das, L. Green, B. Perez, B. Murphy, A. Perring, “Detecting user activities using the accelerometer on Android smartphones”, 2010.

[9] E.M. Tapia, S.S. Intille, K. Larson, “Activity recognition in the home using simple and ubiquitous sensors”, Pervasive Computing, Springer Berlin/ Heidelberg, pp. 158-175, 2004.

[10] C.M. Bishop, “Pattern Recognition and Machine Learning”, Springer, New York, 2007.

[11] V.N. Vapnik, “Statistical Learning Theory”, New York: John Wiley & Sons, 1998.

[12] L. Chen, I. Khalil, “Activity recognition: Approaches, Practices and Trends”, Atlantis Press Review, pp. 1-23, 2010.

[13] J.R. Kwapisz, G.M. Weiss, S.A. Moore, “Activity Recognition using Cell Phone Accelerometers”, SIGKDD Explor. Newsl. 12, pp. 74-82, 2010.

[14] D.M. Karantonis, M.R. Narayanan, M. Mathie, N.H. Lovell, B.G. Celler, "Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring," IEEE transactions on Information Technology in Biomedicine, vol.10, no.1, pp. 156-167, 2006.

[15] K. Hwang, S.-Y. Lee, "Environmental audio scene and activity recognition through mobile-based crowdsourcing", IEEE transactions on Consumer Electronics, vol.58, no.2, pp. 700-705, 2012.

[16] A. Henpraserttae, S. Thiemjarus, S. Marukatat, "Accurate Activity Recognition Using a Mobile Phone Regardless of Device Orientation and Location", International conference on Body Sensor Networks (BSN) 2011, pp.41-46, 2011.

[17] V. Q. Viet, H.M. Thang, D.-J. Choi, "Balancing Precision and Battery Drain in Activity Recognition on Mobile Phone", IEEE 18th International conference on Parallel and Distributed Systems (ICPADS), pp.712-713, 2012.

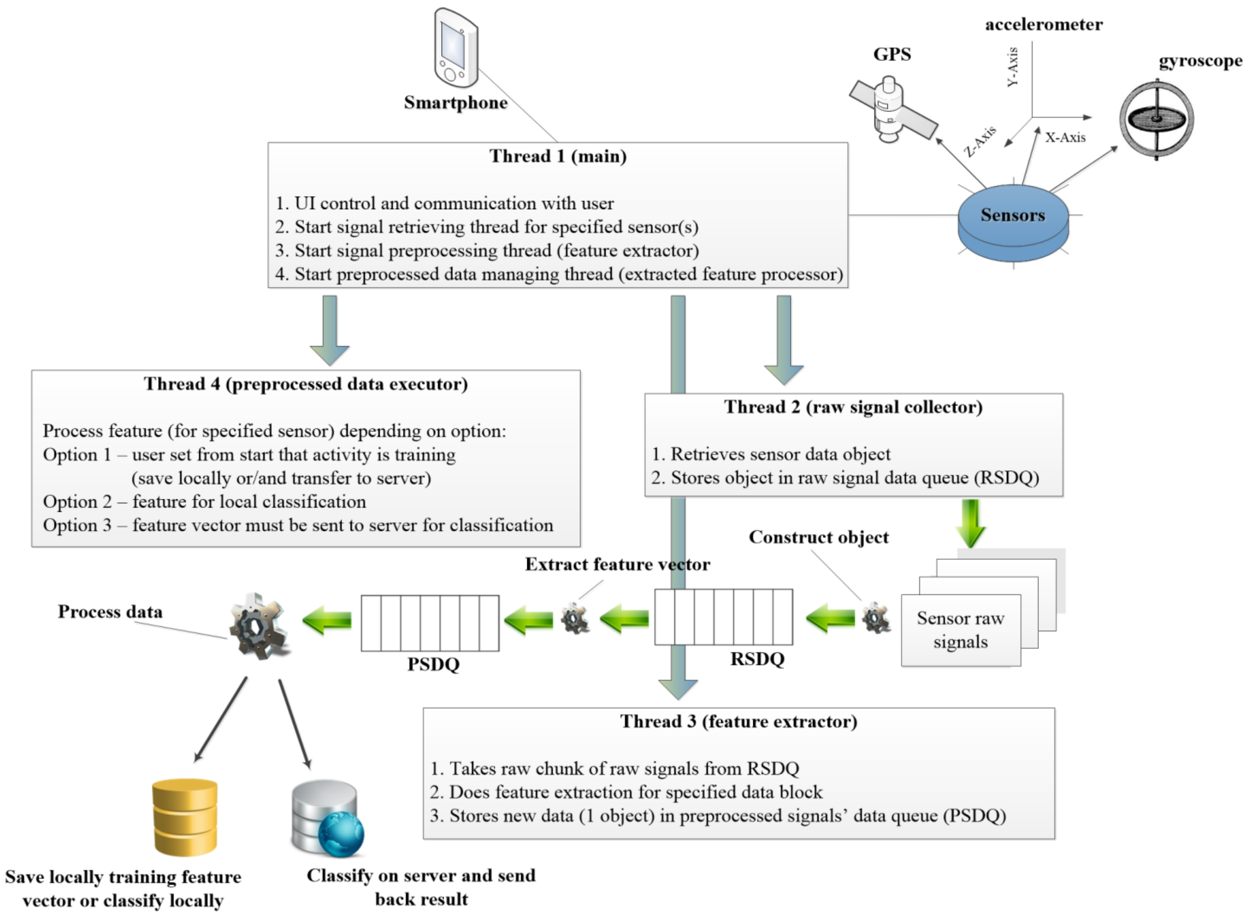
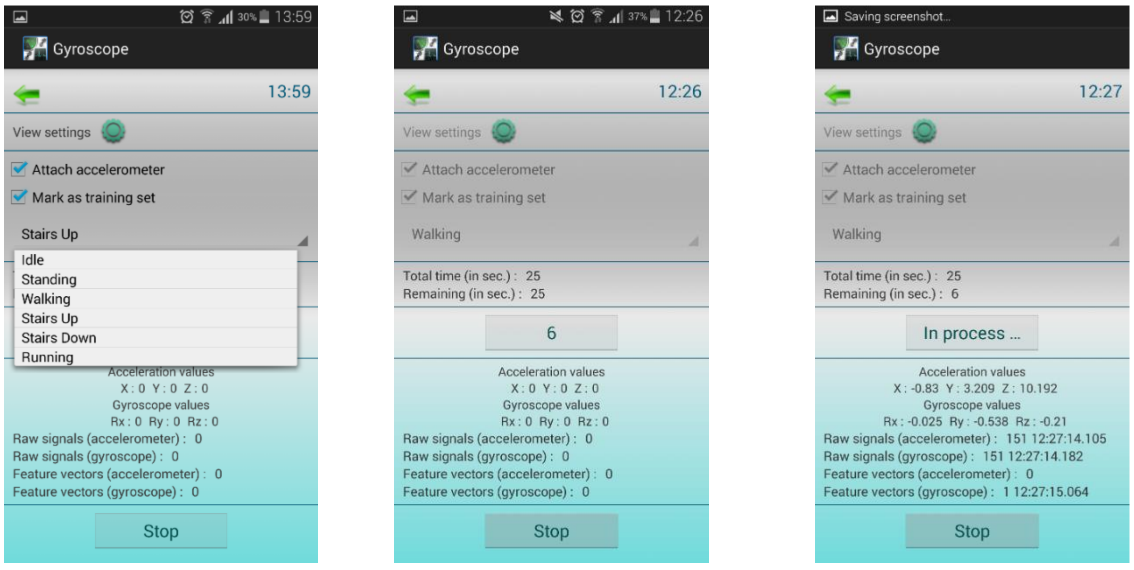

Smartphone
GPS
accelerometer
Y-Axis
Z-Axis
X-Axis
gyroscope
Sensors
Thread 1 (main)
1. UI control and communication with user
2. Start signal retrieving thread for specified sensor(s)
3. Start signal preprocessing thread (feature extractor)
4. Start preprocessed data managing thread (extracted feature processor)
Thread 4 (preprocessed data executor)
Process feature (for specified sensor) depending on option:
Option 1 – user set from start that activity is training
(save locally or/and transfer to server)
Option 2 – feature for local classification
Option 3 – feature vector must be sent to server for classification
Thread 2 (raw signal collector)
1. Retrieves sensor data object
2. Stores object in raw signal data queue (RSDQ)
Construct object
Extract feature vector
Process data
PSDQ
RSDQ
Sensor raw signals
Thread 3 (feature extractor)
1. Takes raw chunk of raw signals from RSDQ
2. Does feature extraction for specified data block
3. Stores new data (1 object) in preprocessed signals' data queue (PSDQ)
Save locally training feature vector or classify locally
Classify on server and send back result

Figure 1. Multithread mechanism diagram for sensor signal continuous sequences processing.



a)

30% 13:59
Gyroscope
13:59
View settings
☑ Attach accelerometer
☑ Mark as training set
Stairs Up
Idle
Standing
Walking
Stairs Up
Stairs Down
Running
Acceleration values
X : 0 Y : 0 Z : 0
Gyroscope values
Rx : 0 Ry : 0 Rz : 0
Raw signals (accelerometer) : 0
Raw signals (gyroscope) : 0
Feature vectors (accelerometer) : 0
Feature vectors (gyroscope) : 0
Stop

b)

37% 12:26
Gyroscope
12:26
View settings
☑ Attach accelerometer
☑ Mark as training set
Walking
Total time (in sec.) : 25
Remaining (in sec.) : 25
6
Acceleration values
X : 0 Y : 0 Z : 0
Gyroscope values
Rx : 0 Ry : 0 Rz : 0
Raw signals (accelerometer) : 0
Raw signals (gyroscope) : 0
Feature vectors (accelerometer) : 0
Feature vectors (gyroscope) : 0
Stop

c)

Saving screenshot...
Gyroscope
12:27
View settings
☑ Attach accelerometer
☑ Mark as training set
Walking
Total time (in sec.) : 25
Remaining (in sec.) : 6
In process ...
Acceleration values
X : -0.83 Y : 3.209 Z : 10.192
Gyroscope values
Rx : -0.025 Ry : -0.538 Rz : -0.21
Raw signals (accelerometer) : 151 12:27:14.105
Raw signals (gyroscope) : 151 12:27:14.182
Feature vectors (accelerometer) : 0
Feature vectors (gyroscope) : 1 12:27:15.064
Stop

Figure 2. Inertial signal processing stage. a) user determines whether motion is for training or for classification, in case of training he specifies activity type, b) count down timer works before actual data processing starts, constructs necessary data structures, c) signal processing stage that uses multithreading