Gesture Recognition for Interest Detection in Mobile Applications

Jérôme Van Zaen, Jody Hausmann, Kevin Salvi and Michel Deriaz*
Institute of Services Science
University of Geneva, Switzerland
{jerome.vanzaen, jody.hausmann, kevin.salvi, michel.deriaz}@unige.ch

***Abstract*—Gestures are a fast and efficient mean to transmit information. They are used in a large number of situations where speaking is not as effective or even not possible, such as to indicate precisely a point of interest or to warn about a danger in a noisy environment. Furthermore, gestures can also be used for intuitive human-computer interfaces where specific tasks would otherwise require navigating through graphical interface menus. Consequently, solutions to provide reliable and accurate gesture recognition have been investigated extensively in the past years. In this paper, we propose a gesture recognition system to detect user interest with a sensor-embedded mobile phone. Specifically, this system uses hidden Markov models to recognize pointing gestures. Once such a gesture has been recognized, it is straightforward to identify the point of interest based on the user location and the phone orientation. In a subject-independent scenario, we obtained a recognition accuracy above 91% with the accelerometer when discriminating between pointing gestures and similar gestures that are common with a mobile phone (e.g. looking at the screen). When using the gyroscope in addition to the accelerometer, the accuracy raised above 98%.**

***Keywords*—Gesture recognition, mobile phone, accelerometer, gyroscope, vector quantization, hidden Markov model.**

I. Introduction

The importance of efficient human-computer interactions is increasing rapidly in the framework of interactive and intelligent computing, and several approaches have been considered for this purpose. Gesture recognition, the process by which the gestures made by the user are recognized by the receiver, is one of these approaches [1]. A large number of recognition systems have been proposed recently, most of them either based on computer vision techniques or motion sensors. In particular, motion sensors have gained increased popularity lately due to their widespread availability, low price and ease of use. For instance, almost all recent mobile devices, such as smartphones and tablet computers, integrate a range of sensors [2], the most common ones being accelerometers, gyroscopes and magnetometers. Consequently, gestures have the potential to become an intuitive, fast and efficient input method for specific tasks [3], [4].

In particular, gestures can be used to detect user interest. For instance, in a city guide application, the user can perform a pointing gesture with his mobile device in order to obtain information concerning a building or a monument. Indeed, once the pointing gesture is detected, the point of interest can be identified based on the user position (obtained from GPS or another system) and the device orientation (obtained from an electronic embedded compass). Detecting user interest is one of the objectives of the MoveYourStory project. Specifically, it is aimed at integrating gesture recognition for interest detection in the Walking the Edit (WtE) application (http://walking-the-edit.net/en/) for mobile phones. This application generates personalized movies based on the user's location, displacement and behavior. An editing algorithm dynamically selects geolocalized audiovisual excerpts from a database of annotated media files in order to construct a narrative. Location and displacement information is obtained from GPS data, and the behavior is estimated by recognizing the activity of the user [5]. For instance, if the user is traveling fast by bike, the generated story will contain many short videos, whereas they will be longer if he is taking a stroll. Potential applications of this movie generation framework include (but are not limited to) cultural, patrimonial and touristic uses.

The rest of this paper is organized as follows. In Section II, an overview of related work on gesture recognition is presented. Then, the proposed recognition system is described in full detail in Section III. Experimental results are reported and discussed in Section IV. Finally, a brief conclusion is given in Section V.

II. Related Work

Gesture recognition has been studied extensively over the past two decades as a promising approach for human-computer interactions [1], and it has been used in a wide range of applications. Typical examples are detection of activities of daily living [6], [7], user identification and authentication [8], sign language recognition [9], [10], user interfaces [3], [4], [11], [12], and human-robot interfaces [13]. As mentioned in Section I, the two most widely-used approaches for gesture recognition are based either on computer vision techniques or on motion sensors. However, vision-based techniques are not ideal in a truly mobile context due to limited wearability, high computation requirements, and sensitivity to lighting conditions and camera facing angle [6], [14]. By contrast, motion sensors have low cost and are widely available. Several devices with such sensors (mostly accelerometers and gyroscopes) have been used for gesture recognition. Typical examples include Wii controllers [8], [12], [14], [15], mobile phones [3], [4], wrist watches [16], [17], and custom sensor-embedded devices [11]. As our system is based on motion sensors, we focused on this type of solutions.

In most of the proposed solutions, the gesture recognition process can be separated in several stages: data acquisition, feature extraction and recognition. The first step is straightforward: gestures are recorded from a set of motion sensors. Then, features characterizing the gestures are extracted. The choice of features varies widely across different systems. However, some approaches are common to several solutions, such as windowing sensor data over sliding overlapping windows to reduce the effect of noise [3], [8], [14] or scaling the data to take into account intra- and inter-subject variability [11]. Another approach is to discard data samples with no motion or that are too similar to the previous ones in order to reduce the computational load [12]. Furthermore, in the cases where the next stage requires discrete data, vector quantization is applied to generate codewords [11], [12]. Once the features are extracted, recognition is performed. This stage is often based on dynamic time warping (DTW) [3], [8] or hidden Markov models (HMMs) [11], [12], [16]. DTW requires only limited computational power compared to HMMs. However, it does not perform well in subject-independent scenarios since, unlike statistical approaches, it lacks resilience to inter-subject variability. Indeed, in cases where the system should be reliable with different subjects, statistical models such as HMMs are more appropriate. Other techniques were also applied successfully to this problem: support vector machines [14] and conditional random fields [18] to name a few. Collectively, these studies suggest that many different approaches can be used for gesture recognition. However, they all have specific advantages and drawbacks, and thus the ultimate choice depends on the intended application. Therefore, as our system is designed to achieve high recognition accuracy in subject-independent scenarios, we chose an approach based on HMMs because of their ability to model inter-subject variability.

III. System Description

The gesture recognition system we propose is composed of several parts: data acquisition, data windowing, feature extraction, vector quantization, and model matching. The complete structure of the system is shown in Fig. 1 where dashed lines indicate operations performed during training only. All parts are described in full detail in the following sections. This system was implemented using both MATLAB and the Android platform.

A. Data Acquisition

Signals are acquired from the 3-axis accelerometer and 3-axis gyroscope sensors of a Samsung Galaxy S4 mobile phone running the Android operating system. The gravity component is included in accelerometer data. The sampling period is set to 20 ms. It is important to note that the Android platform is not real-time and this period is only a suggested value. Consequently, some sampling jitter is present in sensor data. Fig. 2 shows typical histograms of jitter for both accelerometer and gyroscope.

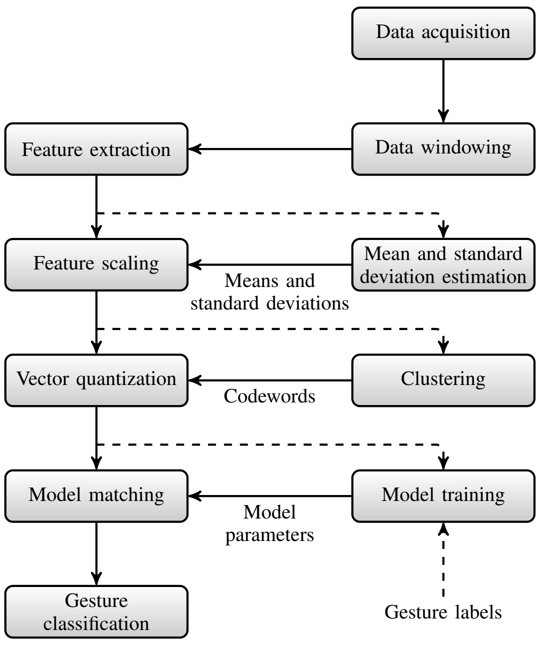


Fig. 1. Structure of the system for gesture recognition. Dashed lines denote training-only operations.

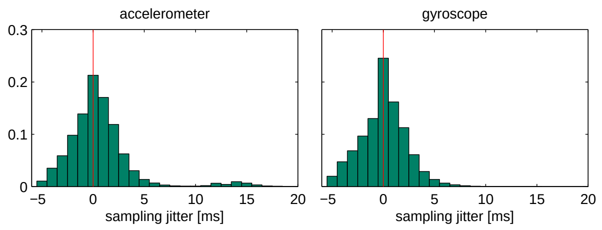


Fig. 2. Histograms of sampling jitter for the accelerometer and gyroscope sensors. In some cases, delays longer than 60 ms were observed (not shown).

B. Data Windowing

In order to compensate for the irregular sampling, the signals recorded from the sensors are windowed into sliding windows. This results in sliding windows with a constant delay between them. It is important to mention that the number of data samples in each window is not constant due to sampling jitter. In the uncommon case where a window does not contain any data samples, it is simply discarded.

C. Feature Extraction and Scaling

A feature vector is then extracted from each sliding window. In the training phase, the means and standard deviations of the components of all feature vectors in the training set are estimated. These values are then used to center and scale the feature vectors. We investigated two simple sets of features. In the first one, the mean of accelerometer values are computed along each axis, resulting in 3-dimensional feature vectors. In the second one, the gyroscope is used in addition to the accelerometer, yielding 6-dimensional feature vectors.

D. Vector Quantization

In the next stage, the feature vectors are quantized by searching in a set of prototype vectors, known as the codewords, for the closest one in terms of Euclidean distance (L^2 norm). In other words, each feature vector is translated to the index of the nearest codeword. The prototype vectors are computed in the training phase by clustering all feature vectors in the training set with the k-means++ algorithm [19], which is basically Lloyd's method [20] with an optimized random initialization scheme. Since this algorithm only finds a local minimum, it is run 10 times and the best clustering in terms of distance from each feature vector to the closest prototype vector is selected.

E. Model Matching and Classification

The quantized feature vectors, also known as the sequence of observations, are then fed to HMMs with discrete emission probabilities to perform gesture recognition. One HMM is used to represent each gesture that needs to be recognized. Left-right models with a jump limit of one (at most one state can be skipped) were selected since most gestures evolve smoothly from a start point to an end point. It is of course possible to use an ergodic structure (or a less common one) if needed. Recognition is performed with the scaled forward procedure [21]. This yields the probability of the current sequence of observations given each model. The current gesture is then classified as the one corresponding to the model with the largest probability.

The model parameters for each gesture, namely the matrix of transition probabilities, the matrix of emission probabilities and the initial probabilities, are trained with the Baum-Welch algorithm [21], modified in order to take into account multiple training sequences. It is worth mentioning that the initial probabilities are never updated due to the left-right structure. Indeed, they are always equal to one for the first state and to zero for the others. The training algorithm is stopped after 2000 iterations or when the relative change in log-likelihood falls under 10^{-3}. Furthermore, any entry of the matrix of emission probabilities less than 10^{-9} is set to 10^{-9} in order to avoid zero probability during recognition. Naturally, if any modification is made, the matrix is re-normalized to ensure that each row sums to one. As the Baum-Welch algorithm only leads to a local maximum of the likelihood of training sequences, it is applied five times with random initial parameters, and the set of parameters yielding the largest likelihood for training data is selected.

IV. Experimental Results

The performance of the gesture recognition system was evaluated on a database of gestures for different sets of parameters for both subject-dependent and subject-independent scenarios in terms of accuracy (percentage of gestures classified correctly). All details regarding this evaluation are given in the following sections.

TABLE I
List of gestures

Name	Description
`lcall`	answering a phone call with the left hand
`lpointf`	pointing forward with the left hand
`lpointl`	pointing left with the left hand
`lpointr`	pointing right with the left hand
`lscreen`	looking at the screen with the left hand
`ltime`	checking the time with the left hand
`rcall`	answering a phone call with the right hand
`rpointf`	pointing forward with the right hand
`rpointl`	pointing left with the right hand
`rpointr`	pointing right with the right hand
`rscreen`	looking at the screen with the right hand
`rtime`	checking the time with the right hand

A. Gesture Data

Sensor data were collected from six healthy subjects (two females, one left-handed) with a custom Android application for gesture acquisition. This application was running on a Samsung Galaxy S4 smartphone and logged data from the embedded 3-axis accelerometer and 3-axis gyroscope to text files. The sampling period was set to 20 ms, but it was only a suggested value as discussed in Section III-A. The subjects were instructed to press a button on the touchscreen at the beginning of the gesture and release it at the end in order to obtain segmented gesture data.

As the main reason for integrating gesture recognition in the WtE application is to detect user interest, we focused on pointing gestures and common gestures that can be similar, such as checking the time or looking at the screen. The complete list of the 12 considered gestures is given in Table I. Several pointing gestures were considered to take into account some of the large variability of such gestures. Furthermore, *looking at the screen* and *checking the time* gestures are very similar, but the latter ones have smaller amplitudes and slightly shorter durations. As the subjects were asked to perform the gestures as naturally as possible, there were only few differences between some of them. The starting position for all gestures was holding the phone in the hand with the arm along the body. Gesture data were recorded over five days. Each day, each gesture was recorded 10 times for each subject. This resulted in a database of 600 gesture recordings per subject, or 3600 recordings in total. This protocol was used in order to investigate how the recognition system can cope with day-to-day variations. Two examples of accelerometer and gyroscope signals for `lpointf` and `rpointf` gestures are shown in Figure 3. Once all gestures were recorded, they were imported in MATLAB for further processing.

B. Parameters

The effects of several parameters on recognition performance were investigated. First, we tested whether gyroscope data can improve the correct recognition rate. For this purpose, the accuracy of the system was measured when using only accelerometer data and when using both accelerometer and gyroscope data. Second, the influence of the window length

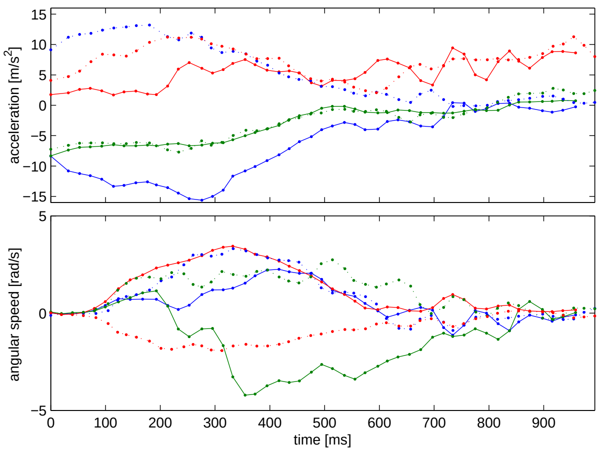


Fig. 3. Examples of accelerometer (top) and gyroscope (bottom) signals for `lpointf` (solid lines) and `rpointf` (dotted lines) gestures. The signals for the x-axis, the y-axis and the z-axis are plotted in blue, green and red.

and the shift between consecutive windows was investigated by varying the former from 40 ms to 100 ms in 20 ms steps and the latter from 20 ms to 50 ms in 10 ms steps (50% overlap in all cases). For vector quantization, four different numbers of codewords were tested: 40, 60, 80 and 100. Finally, the number of states in the HMMs was varied between 3, 4 and 5. The number of states was the same for all models, as all gestures have approximately equivalent duration and complexity. All the other parameters were set to the values described in Section III.

C. Performance Measure

The performance of the proposed system was evaluated in terms of recognition accuracy, the percentage of gestures classified correctly. In particular, the accuracy was measured for two different cases. In the first one, the goal was to recognize every gesture separately (all vs all gestures), while in the second one, it was to discriminate between pointing (`lpointf`, `lpointl`, `lpointr`, `rpointf`, `rpointl` and `rpointr`) and other gestures (`lcall`, `lscreen`, `ltime`, `rcall`, `rscreen` and `rtime`) (pointing vs other gestures). The second case was considered since one of the main objectives of this project was to detect when a user is pointing with his mobile phone.

Two different strategies were used for cross-validation. In the subject-dependent case, the signals were partitioned into five datasets, one for each day. The gesture recognition system was trained with signals from four datasets and tested on signals from the remaining one. This was repeated so that each dataset was used once for testing. The accuracy estimated from each test set were averaged for each subject and a global measure of accuracy was computed by averaging the accuracy values of all subjects.

In the subject-independent case, a leave-one-subject-out strategy was used. Namely, each test set was built by grouping together all recording from one subject. Therefore, recognition accuracy was evaluated by training the gesture recognition system with data from five subjects and testing it with the remaining dataset. As before, this was repeated so that each dataset was used once for testing. The global accuracy was obtained by averaging the accuracy measurements for all subjects.

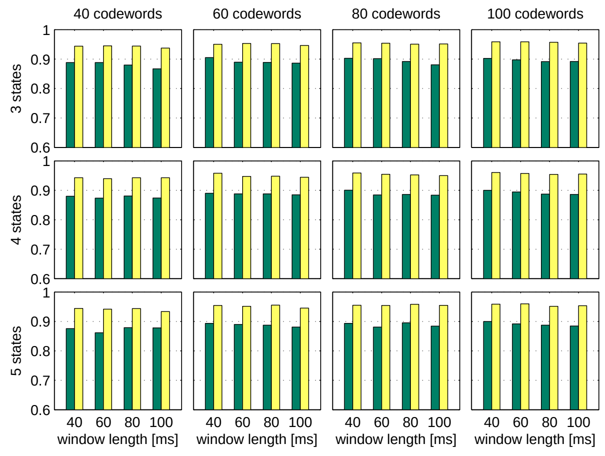
D. Results

Recognition accuracy for all parameter configurations measured in subject-dependent tests and subject-independent tests are reported in Fig. 4 and Fig. 5 respectively. Furthermore, the five best configurations in terms of accuracy are summarized in Tables II and III for all cases.

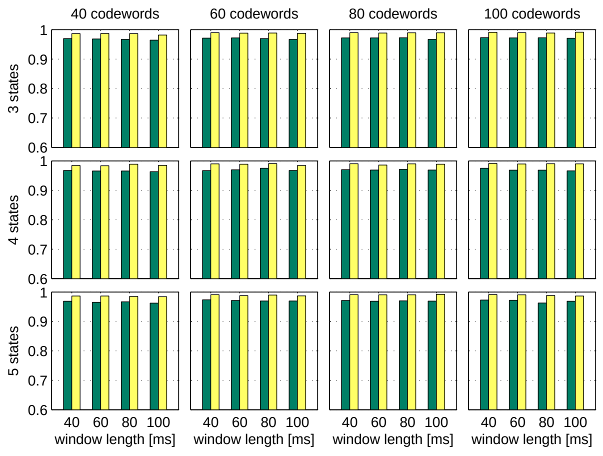
In the subject-dependent scenario, accuracy values around 96% were obtained when recognizing gestures separately. Thus, even with similar gestures, the recognition system showed reliable performance, Furthermore, when discriminating only pointing gestures from the others, the accuracy raised above 99%, almost perfect recognition. In fact, for some subjects, perfect recognition (100% accuracy) was achieved. The effects of window length, number of codewords and number of states were limited on recognition accuracy, even though more codewords often led to slightly better performance. By contrast, the main factor is clearly the inclusion of gyroscope data. Indeed, using the gyroscope in addition to the accelerometer increased the recognition accuracy by more than 5% in the all vs all gestures case, and by more than 1.5% in the pointing vs other gestures case.

In the subject-independent scenario, the overall performance decreased drastically when recognizing all gestures separately. In particular, when using accelerometer data only, accuracy dropped below 70%. Using the additional information provided by the gyroscope increased the accuracy up to 82%, which is much better but still insufficient to be practical. These results are not surprising as some of the gestures (Table I) are very similar. However, when discriminating pointing and other gestures, the recognition performance improved substantially. Indeed, with the best parameter configurations, accuracy raised above 91% for accelerometer data only. Moreover, when using both the accelerometer and the gyroscope, the accuracy raised above 97%, even reaching 98% for one specific configuration. The same remarks about the influence of the different parameters as for the subject-dependent scenario can be repeated in this case. Indeed, there were no clear difference for different window lengths, numbers of codewords or states. As before, the only real difference was using gyroscope data or not.

Taken together, these results suggest that, with such a system and for specific tasks, gesture recognition can be considered as a reliable and practical input method for interacting with a mobile phone. Furthermore, it is also important to mention that no noticeable delay was observed during recognition when this system was running on a Samsung Galaxy S4 smartphone. However, training is computationally intensive and should not be performed on a mobile device, especially in subject-independent scenarios where the amount of training data can be very large.

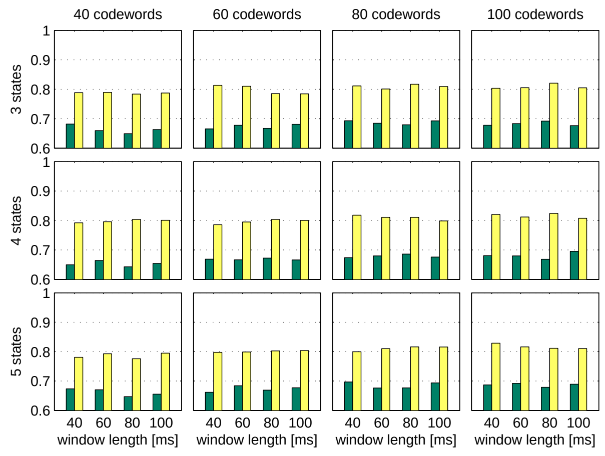


(a) All vs all gestures.

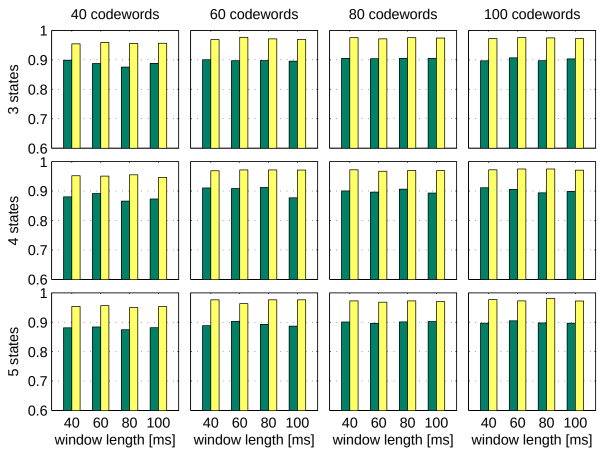


(b) Pointing vs other gestures.

Fig. 4. Measured accuracy of subject-dependent tests for (a) all vs all gestures and (b) pointing vs other gestures recognition. Recognition accuracy is reported in each chart for the tested window lengths. Each column corresponds to a number of states, while each row corresponds to a number of codewords. Accuracy in green was measured with accelerometer data only, whereas the one in yellow was measured with both accelerometer and gyroscope data.



(a) All vs all gestures.



(b) Pointing vs other gestures.

Fig. 5. Measured accuracy of subject-independent tests for (a) all vs all gestures and (b) pointing vs other gestures recognition. The same remarks as for Fig. 4 apply.

V. Conclusion

Gesture recognition is a promising approach to detect user interest as it does not require visual attention from the user compared to traditional techniques. This is especially true for mobile devices which usually are not the focus of user attention and whose screens can be quite small. Furthermore, it is often faster and more efficient to perform a gesture than to navigate through interface menus. Experimental tests have shown that our system achieves high recognition accuracy in both subject-dependent and subject-independent scenarios when detecting pointing gestures. Subject-independent performance was not as satisfactory when recognizing all gestures separately. However, this is not surprising as some of the considered gestures are very similar and thus difficult to discriminate. The test results also revealed the importance of using a gyroscope, if available, to improve the overall performance. Indeed, this sensor was a key point to obtain high detection accuracy. By contrast, the other parameters (window length, number of codewords and states) were not as decisive. Indeed, different values only caused slight differences in recognition performance.

In the future, we will explore how to extend the proposed system to address gesture recognition in a continuous stream of sensor data, also known as gesture spotting. Indeed, the fact that the user needs to indicate when a gesture starts and ends can have a negative impact on usability. The main challenges will be to achieve high recognition performance while keeping false detections at minimum and to maintain a reasonable computational load for the limited processing capabilities of mobile devices [16]. In addition, it would be interesting for the system to adapt to a particular user for improved performance or even to learn new gestures online while limiting as much as possible the number of needed training samples.

Acknowledgment

This work was supported by the Swiss Commission for Technology and Innovation (CTI grant 13943.1). The authors would like to thank Adrian Hindle and Jonathan Charnas who helped to improve the English quality of this paper.

TABLE II
BEST RESULTS OF SUBJECT-DEPENDENT TESTS

Accuracy [%]	Window length [ms]	Codewords	States
90.53	40	60	3
90.28	40	80	3
90.25	40	100	3
90.17	60	80	3
90.06	40	80	4

(a) All vs all gestures, accelerometer.

Accuracy [%]	Window length [ms]	Codewords	States
96.06	40	100	4
95.97	60	100	5
95.89	60	100	3
95.86	40	80	4
95.86	40	100	5

(b) All vs all gestures, accelerometer and gyroscope.

Accuracy [%]	Window length [ms]	Codewords	States
97.47	40	100	4
97.47	80	60	4
97.36	40	60	5
97.28	40	100	5
97.28	40	100	3

(c) Pointing vs other gestures, accelerometer.

Accuracy [%]	Window length [ms]	Codewords	States
99.19	100	80	5
99.14	100	100	3
99.14	40	100	5
99.11	40	60	5
99.08	40	100	3

(d) Pointing vs other gestures, accelerometer and gyroscope.

TABLE III
BEST RESULTS OF SUBJECT-INDEPENDENT TESTS

Accuracy [%]	Window length [ms]	Codewords	States
69.69	40	80	5
69.50	100	100	4
69.36	100	80	5
69.31	40	80	3
69.28	100	80	3

(a) All vs all gestures, accelerometer.

Accuracy [%]	Window length [ms]	Codewords	States
82.86	40	100	5
82.39	80	100	4
82.08	80	100	3
82.03	40	100	4
81.81	40	80	4

(b) All vs all gestures, accelerometer and gyroscope.

Accuracy [%]	Window length [ms]	Codewords	States
91.19	80	60	4
91.11	40	100	4
91.03	40	60	4
90.86	60	60	4
90.67	60	100	3

(c) Pointing vs other gestures, accelerometer.

Accuracy [%]	Window length [ms]	Codewords	States
98.06	80	100	5
97.69	40	100	5
97.64	60	60	3
97.61	40	60	5
97.61	100	60	5

(d) Pointing vs other gestures, accelerometer and gyroscope.

REFERENCES

[1] S. Mitra and T. Acharya, "Gesture recognition: a survey," *IEEE T Syst Man Cyb*, vol. 37, no. 3, pp. 311–324, 2007.

[2] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," *IEEE Commun Mag*, vol. 48, no. 9, pp. 140–150, 2010.

[3] B. Choe, J.-K. Min, and S.-B. Cho, "Online gesture recognition for user interface on accelerometer built-in mobile phones," in *Neural Information Processing. Models and Applications*, K. W. Wong, B. U. Mendis, and A. Bouzerdoum, Eds. Springer, 2010, pp. 650–657.

[4] J. Ruiz and Y. Li, "DoubleFlip: a motion gesture delimiter for mobile interaction," in *Proc SIGCHI Conf on Hum Factors in Comput Syst*, 2011, pp. 2717–2720.

[5] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *SIGKDD Explor*, vol. 12, no. 2, pp. 74–82, 2011.

[6] H. Junker, O. Amft, P. Lukowicz, and G. Tröster, "Gesture spotting with body-worn inertial sensors to detect user activities," *Pattern Recogn*, vol. 41, no. 6, pp. 2010–2024, 2008.

[7] C. Zhu and W. Sheng, "Wearable sensor-based hand gesture and daily activity recognition for robot-assisted living," *IEEE T Syst Man Cy A*, vol. 41, no. 3, pp. 569–573, 2011.

[8] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "uWave: accelerometer-based personalized gesture recognition and its applications," *Pervasive Mob Comput*, vol. 5, no. 6, pp. 657–675, 2009.

[9] T. Starner, J. Weaver, and A. Pentland, "Real-time American sign language recognition using desk and wearable computer based video," *IEEE T Pattern Anal*, vol. 20, no. 12, pp. 1371–1375, 1998.

[10] R.-H. Liang and M. Ouhyoung, "A real-time continuous gesture recognition system for sign language," in *IEEE Int Automat Face Gest Recogn*, 1998, pp. 558–567.

[11] J. Kela, P. Korpipää, J. Mäntyjärvi, S. Kallio, G. Savino, L. Jozzo, and S. Di Marca, "Accelerometer-based gesture control for a design environment," *Pers Ubiquit Comput*, vol. 10, no. 5, pp. 285–299, 2006.

[12] T. Schlömer, B. Poppinga, N. Henze, and S. Boll, "Gesture recognition with a wii controller," in *Proc Int Conf Tangible Embedded Interact*, 2008, pp. 11–14.

[13] C.-B. Park and S.-W. Lee, "Real-time 3d pointing gesture recognition for mobile robots with cascade hmm and particle filter," *Image Vision Comput*, vol. 29, no. 1, pp. 51–63, 2011.

[14] J. Wu, G. Pan, D. Zhang, G. Qi, and S. Li, "Gesture recognition with a 3-D accelerometer," in *Ubiquitous Intelligence and Computing*. Springer, 2009, pp. 25–38.

[15] M. Hoffman, P. Varcholik, and J. J. LaViola, "Breaking the status quo: improving 3D gesture recognition with spatially convenient input devices," in *Proc IEEE Virt Real Conf*, 2010, pp. 59–66.

[16] G. Raffa, J. Lee, L. Nachman, and J. Song, "Don't slow me down: bringing energy efficiency to continuous gesture recognition," in *Int Symp Wearable Comput*, 2010, pp. 1–8.

[17] L. Porzi, S. Messelodi, C. M. Modena, and E. Ricci, "A smart watch-based gesture recognition system for assisting people with visual impairments," in *Proc ACM Int Work Interact Multimedia Mob Portable Dev*, 2013, pp. 19–24.

[18] L.-P. Morency, A. Quattoni, and T. Darrell, "Latent-dynamic discriminative models for continuous gesture recognition," in *Proc CVPR IEEE*, 2007, pp. 1–8.

[19] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in *Proc ACM-SIAM Symp Discrete Algorithms*, 2007, pp. 1027–1035.

[20] S. P. Lloyd, "Least squares quantization in PCM," *IEEE T Inform Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[21] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *P IEEE*, vol. 77, no. 2, pp. 257–286, 1989.