Gesture recognition using mobile phone's inertial sensors

Xian Wang, Paula Tarrío, Eduardo Metola, Ana M. Bernardos, José R Casar

Data Processing and Simulation Group, ETSI. Telecomunicación

Universidad Politécnica de Madrid, Madrid, Spain

{wang.xian, paula, eduardo.metola, abernardos, jramon}@grpss.ssr.upm.es

Abstract The availability of inertial sensors embedded in mobile devices has enabled a new type of interaction based on the movements or "gestures" made by the users when holding the device. In this paper we propose a gesture recognition system for mobile devices based on accelerometer and gyroscope measurements. The system is capable of recognizing a set of predefined gestures in a user-independent way, without the need of a training phase. Furthermore, it was designed to be executed in real-time in resource-constrained devices, and therefore has a low computational complexity. The performance of the system is evaluated offline using a dataset of gestures, and also online, through some user tests with the system running in a smart phone.

1. Introduction

Mobile phones have become wearable computers equipped with various sensors due to the advance in microelectronics [1, 2], which not only increased the processing power of such devices but also made possible new forms of input interfaces, such as touch screen devices and gesture-based user interfaces [3].

Hand gesture is a powerful, natural means of communication between human beings. Now it can be a promising way to interact with computers, where gesture recognition is the core of such technique. Classically, hand gestures have been detected and recognized using camera-based computer vision algorithms. However, these techniques can be slow and require a high computational power, which leads to a significant energy consumption [2]. A more suitable approach for resource-constrained devices is to use their embedded sensors (such as magnetometers, gyroscopes, or accelerometers) to perform the recognition. Using embedded sensors has the advantage that the gesture recognition can be done in the own device and that the accuracy is not affected by lighting conditions or camera calibration. Furthermore, the cost and power consumption are lower [4].

There exist multiple challenges in hand gesture recognition for mobile phones, for example, a standardized "vocabulary" is missing, the interaction needs to be real time and the algorithms should be able to run on a platform highly constrained in terms of cost and system resources. What's more, user acceptability is also a key consideration: will they feel comfortable waving arms in a public place [2]?

Many gesture recognition systems based on accelerometers have been developed. One example is Georgia Tech Gesture Toolkit [5], that provides tools to support gesture recognition and has been used in several ongoing projects, such as an automobile gesture panel, patterned blink recognition and mobile sign language recognition. The work in [6], where 8 predefined gestures including translations, circles, and pentagram are defined and more than 98% recognition accuracy is achieved, provides a natural and intuitive way to control the browser application on a large screen. These techniques are also used in mobile gaming systems to enhance user experiences [2], where a set of 8 gestures such as single-circles, squares, triangles and double-circles are provided and an overall accuracy of 96.25% is obtained. The authors in [7] developed the uWave algorithm, which is utilized in gesture-based user authentication and interaction with a three-dimensional mobile user interface. In this work, they employ a set of 8 gestures identified by Nokia research study and the recognition accuracy is about 98%.

Gesture recognition is a type of pattern recognition. Various methods could be utilized, such as conditional Gaussian models, support vector machines [8], Bayesian networks [9], dynamic time warping (DTW) and hidden Markov models (HMMs). DTW [2, 7] and HMMs [1-3, 5, 6, 10-13] are two of the most popular approaches adopted, both of which were investigated in speech recognition area.

Only a few [2, 3, 4, 7] of the techniques mentioned above are implemented on a resource-constrained platform such as a mobile phone. What's more, most of these proposals target at user-dependent gesture recognition because of the difficulty of user-independent gesture recognition and because user-independent gesture recognition may not be that attractive as speaker-independent speech recognition, as there are no standard gestures for interaction [7]. In this work, however, our goal is to develop a user-independent system which is light, real-time, and with low consumption. To this end, we define a set of simple gestures, which are intuitive and comfortable to perform, and recognize them in a user-independent manner which is user friendly and does not require training the system.

In summary, we make the following contributions: We define a set of rotation and translation gestures and develop real-time methods to recognize them with very low computational cost. As this is a work in progress, the recognition of rotation gestures is implemented on a mobile phone and tested in real-time, whereas the recognition of translation gestures is implemented on a PC and tested offline with a dataset of gestures performed by real users. Preliminary results show a high recognition accuracy and low time complexities.

The rest of the paper is organized as follows: Section 2 enumerates possible application scenarios of our gesture recognition system. Section 3 describes the set of gestures we want to recognize. Section 4 presents the architecture of the recognition system. Section 5 reports the validation results of the rotation and translation gesture recognition systems with real experiments and simulations, respectively. Finally, we conclude in section 6 and point out future work.

2. Application scenarios

Gesture recognition systems for mobile devices have a wide variety of applications in several scenarios, such as smart environments, teaching/learning, robot control, etc. For example, gestures performed with the user's mobile phone can be used to control the equipment of a smart room in a hotel (e.g. switching off the light when the position of the phone changes from facing up to facing down, switching on the lights by making the opposite movement, raising/lowering the blinds by doing an up/down movement while pointing to the window, opening/closing the curtains with a left/right movement, raising/lowering the room temperature by making an up/down movement while pointing to the air conditioner, calling the room service by shaking the phone, etc.). Gestures can also be used in a teaching scenario, or even in conferences or business presentations to control the computer and the slides. For example, a movement to the left/right goes to the previous/next slide, a spiral to the left/right goes to the first/last slide, moving up/down the phone will control the volume of the computer if a video/audio file is played, shaking the phone can switch on/off the projector and once it is off, an up/down movement can raise/lower the screen, etc. Another practical application of mobile gesture recognition is to control robots, toys or vehicles: turning the phone to the left or right can control the direction of the robot, turning it up and down can control the speed and shaking the phone can switch on/off the robot.

3. Selection of gestures

The selection of the gesture set affects both the experience of user interaction and the recognition accuracy. The author of [1] argues that an extensive set of gestures becomes unpractical because too many gestures have to be learned by the users. The recognition results in [7] highlight the importance of selecting the right gesture vocabulary for high accuracy. More complicated gestures provide more features to distinguish them, resulting in higher recognition accuracy. However, complicated gestures also force users to remember how to perform them and how they are related to functions/actions. Furthermore, in [14], user studies indicate that users tend to use spatial two-dimensional gestures and that utilizing all three dimensions in one gesture is rare. In this section, we describe the gesture set selected for our recognition system, which is divided into two groups: turns and translations.

3.1 Turn gestures

Taking into account that there are three axes in the device, we can define six possible elementary turns, as we can see in Fig. 1. In our system, we target at recognizing 45º and 90º turns of the six possible types.

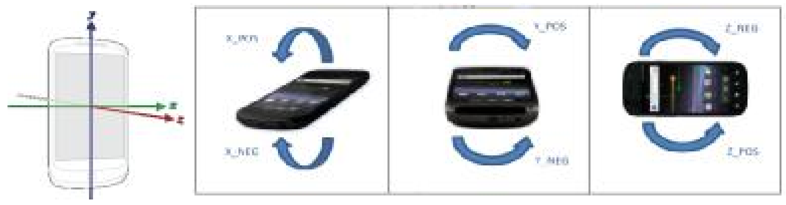


Fig. 1. Six elementary turns around the device's axes.

The complete gesture will be defined by the type of turn and by the initial or final orientation of the device. We have defined six possible orientations with respect to the ground (shown in Fig. 2).

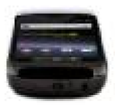
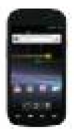
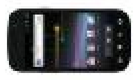
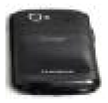
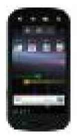
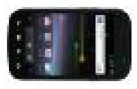
Horizontal_Up	Vertical_Up	Left	Horizontal_Down	Vertical_Down	Right
					

Fig. 2. Main six orientations of the device.

If we make a 90º-turn from one of the orientations in Fig. 2, we will obtain another orientation of Fig. 2. In addition, if a 45º-turn is made intermediate orientations are obtained. We classify the possible orientations in three groups:
- Type I orientations: the gravity only appears in one axis (those in Fig. 2).
- Type II orientations: the gravity appears in two axes. This happens when, from a type I position, there is a 45º-turn.
- Type III orientations: the gravity appears in all the axes. It happens when, from type II positions, there is a 45º-turn over the axis that has not been turned yet.

3.2 Translation gestures

Following the previous research [14], in our work, we have defined six simple translations named backward, forward, left, right, up and down. The following table describes these gestures. The orientation of the device when doing the translations remains the same: in a 'Vertical_Up' orientation.

Table 1. Description of translation gestures

Gestures	**Description (performed with the same fixed pose)**
Backward	Pull the mobile horizontally.
Forward	Push the mobile horizontally.
Left	Move the mobile horizontally from right to left.
Right	Move the mobile horizontally from left to right.
Up	Move the mobile vertically from low position to high.
Down	Move the mobile vertically from high position to low.

4. System architecture

The current system architecture consists of three different layers, a *sensor layer*, which acquires the information provided by the inertial sensors, a *fusion layer*, which includes the recognition algorithms to recognize the gestures, and a *communications layer*, which offers the recognition result to external devices.

4.1 Sensor layer

The gesture detection system relies on the data provided by the gyroscope and the accelerometer embedded in the device. In our case, the acquisition process is done using a Google Nexus S smartphone running Android O.S, where data communication is done by event notifications. As a result we cannot talk about a real sampling frequency, but we have seen that the approximate sampling rates for the gyroscope and the accelerometer are, respectively, 0.0096 seconds and 0.02 seconds per measurement.

In the case of turn gestures, the information collected by the gyroscope allows us to distinguish among the six elementary turns, whereas the accelerometer gives us information about the orientation of the device. Fig. 3 shows some examples of the gyroscope and accelerometer signals for different gestures.

Fig. 3a and 3b show the values acquired from the gyroscope (angular speed around each axis) and accelerometer (acceleration on each axis), for a movement from a ‘Horizontal_Up’ position to a ‘Vertical_Up’ position. Fig. 3c and 3d correspond to a 90º-turn around the vertical axis. As it can be seen, the value of the gyroscope signal in the axis of rotation increases/decreases significantly, which allow us to recognize the type of elementary turn. The acceleration signal after the turn enables recognizing the final position (the axis with the gravity component is the one pointing to the ground).

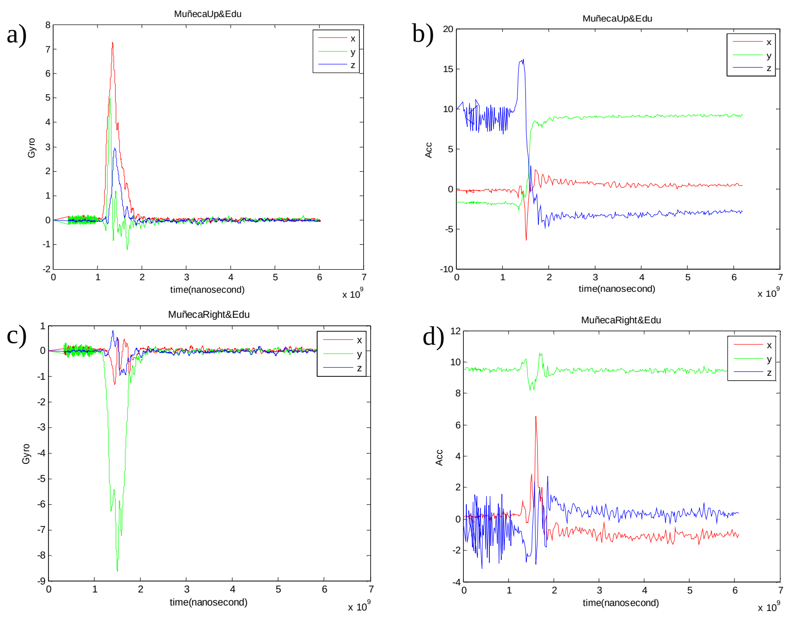


Fig. 3. Examples of gyroscope and accelerometer signals for rotation gestures.

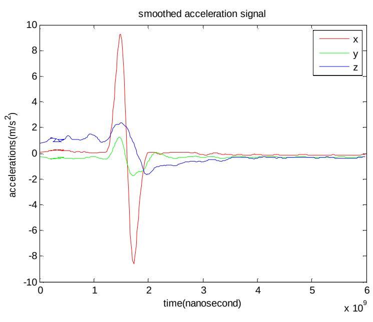


Fig. 4. Example of acceleration signals for a correctly performed ‘Right’ gesture

The translation gesture is a variable force movement, starting and ending with a stationary state. Essentially, the 6 translation gestures are the same but moving in different directions. After the gesture starts, the acceleration value in the direction of movement increases quickly, then changes its direction, and finally returns to zero. So ideally, the acceleration in the axis along the movement direction should look like a sinusoid in one period, and the acceleration data in the other two axes should be close to zero (see Fig. 4). The axis where this pattern appears and the order of the appearance of the curve’s peak and valley indicate the direction of the movement. For example, Fig. 4 represents a ‘Right’ movement (sinusoid-like curve in the x axis, with the peak appearing earlier than the valley).

4.2 Fusion layer

a. Recognition of turn gestures

In order to detect when a turn happens, the fusion layer is permanently gathering the data from the gyroscope sensor. The data obtained by the fusion layer is an array of three elements, each element related to one axis. If one of the elements of the array increases over a predefined threshold, a turn is detected on the corresponding axis. By also taking into account its sign, it is possible to distinguish among the six types of elementary turns. The system is able to recognize 45º and 90º turns, by comparing the initial position of the smartphone and the final position after a turn.

When the value in the axis that has increased, decreases under another threshold (set to indicate that the turn is finishing), the accelerometer sensor is enabled and a fixed number of measurements are taken to estimate the final position. These measurements are averaged, giving a higher weight to later positions in the sequence of accelerometer's data. Then, depending on the magnitude and the sign of this weighted average, a final orientation of the device is selected.

In a type I position, the gravity component relies mostly over one of the three axes. In a type II position, it will be spread over two of the three axes. And finally, a type III position has a significant component of the gravity over the three axes. So, analyzing the three components of the acceleration, if the lowest value is over 3 m/s2, a type III position is estimated. Otherwise, if the second lowest value is over 3 m/s2, a type II position is estimated. Otherwise, mobile position is included in the type I group.

For type I positions, one of the six orientations of Fig. 2 is selected (according to which axis has the gravity component and its sign). For type II/III positions, a combination of two/three type I positions is chosen, e.g. 'Horizontal_Up-Left'.

b. Recognition of translation gestures

For translation gesture recognition, we have developed a method based on analysis of acceleration data and feature extraction. The input of the algorithm is a time series provided by the three-axis accelerometer. Each time sample is a vector of three elements corresponding to the sampled data in the three axes.

As we said before, ideally, the curve in the axis along the movement direction should look like a sinusoid in one period (as in Fig. 4). But in practice, the signals almost never show such a clear shape because of noise and performance of the users (we should never expect the users to make gestures ideally). As a result, the sinusoid-like curves are distorted and appear in all axes with different amplitudes, as shown in Fig. 5. The algorithm of translation gesture recognition in our work consists of three steps, which are described next:

A. Preprocessing

According to our definition of gestures, the mobile phone should be held vertically when doing the gesture. But in practice, a tilt of a few degrees is quite common. As gravity is a relatively large value compared to other acceleration components forced by the user, a small tilt will introduce a significant acceleration component along x and z axes, which would result in wrong recognition. In our work, the normalizing method provided in [1] is utilized to correct small amounts of tilt. Then the rectified data is passed through a Butterworth low pass filter to reduce the effect of noise. The signal is further smoothed by a moving average method where the smoothed value is the un-weighted mean of its previous *n* data points, where *n* is the size of the sample window. The order and cutoff frequency of the filter and *n* are all determined empirically.

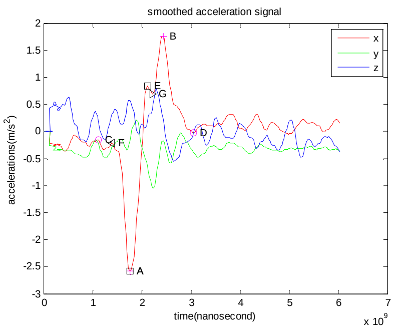


Fig. 5. Pattern generalization in the x direction. F, A, E, G are key points of the original recognized pattern. C, A, B, D are the generalized pattern.

B. Pattern extraction for each axis

The key points of the typical translation pattern are the starting point, the ending point, the peak and the valley of the sinusoid-like curve. Because of noise or the imperfect gesture performance of users, there are more than two local extrema, so first, all the local extrema are detected, and then we find all pairs of adjacent extrema with different signs. The peak and valley of the pattern should be in the pair with the largest absolute difference of the two elements, for example, the pair A-E in Fig. 5. The starting and ending point of the pattern are the two adjacent extrema to peak and valley (F and G in Fig. 5).

Considering the noisy acceleration measurement and that this method is sensitive to fluctuations of the signal, an adjustment of the key points is employed: we look forward or backward several extrema points from the peak and valley points and check whether there are higher extrema (for the peak) or lower extrema (for the valley). This way, the method is robust to small fluctuations. In the same way, the starting and ending of the recognized pattern are the adjacent extrema points of the new peak and valley. For example, in Fig. 5, C, A, B and D are the new four key points of the pattern, which represent the shape of a sinusoid curve.

C. Movement detection

With the pattern extraction mentioned above, the sinusoid-like curve pattern in each axis is obtained. Then we first check the symmetry of the pattern by calculating the following ratios:

$$r_1 = \begin{cases} \dfrac{C-A}{\frac{1}{2}(B-A)}, \text{ if C-A} \leq \frac{1}{2}(B-A) \\ \dfrac{\frac{1}{2}(B-A)}{C-A}, \text{ if C-A} > \frac{1}{2}(B-A) \end{cases} \qquad r_2 = \begin{cases} \dfrac{B-D}{\frac{1}{2}(B-A)}, \text{ if B-D} \leq \frac{1}{2}(B-A) \\ \dfrac{\frac{1}{2}(B-A)}{B-D}, \text{ if B-D} > \frac{1}{2}(B-A) \end{cases}$$

where A is the acceleration value at the valley, B at the peak, C at the starting point and D at the end. To be a “good” pattern, r_1 and r_2 should be both greater than a predefined threshold.

Then we select the axis where there is the largest drop from peak to valley of the pattern and that agrees with our symmetry requirement, as the dimension where the movement occurred. The appearance order of peak and valley further determines the direction of the movement along this dimension. For example, Fig. 5 represents a ‘Left’ gesture.

The time complexity of Butterworth low pass filtering is O(n*logn) due to the Fourier transform. The rest of processing has linear time complexity. Therefore, the method presented above has a complexity of O(n*logn), where n is the length of acceleration signal. In [3], a DTW algorithm is implemented on a mobile phone and has time complexity of O(n*n). The researchers of that paper showed that DTW implementation has lower computational load than HMMs. So hopefully, our algorithm can run faster on a mobile phone platform.

4.3 Communications layer

The gesture recognition application runs autonomously in the device, but a communications layer has been integrated in order to transmit the detected gestures to external devices (either for a more comfortable display of the recognition results or to take the appropriate actions corresponding to the performed gestures).

To this end, a NetworkCommunication component is created in the main component of the application, so that when a gesture is detected, the information is sent through a previously created socket. This information can be gathered from an external element, like a PC, by listening to the data on an already-known IP address and port. In this way, the gesture can be shown in the screen or used to perform the appropriate action.

5. Validation

This section presents some preliminary results of the validation of the gesture recognition system. The recognition of turn gestures has been completely implemented and tested in a Google Nexus S smartphone, running Android O.S, whereas the recognition of translation gestures was implemented in Matlab and tested with a dataset of gestures, collected by several users with the same smartphone.

5.1 Online validation of turn gestures with the mobile phone

a. Performance statistics

The performance of the proposed system is evaluated according to its ability to correctly detect the type of gesture. The speed at which the user performs the gesture determines if it is detectable or not, as we have used an angular speed threshold to filter out random movements. In particular, we have used a threshold of 2 rad/s, which assures that usual wrist movements are detected (with detection probability nearly equal to 100%). Decreasing this threshold would allow detecting slower movements, but would increase the probability of false detections.

Supposing that the system has detected a gesture, the system quality can be described in terms of the probability of correctly recognizing the type of gesture and in terms of time delay in the recognition process. Both parameters can be tuned by modifying the number of accelerometer measurements that are analyzed. Longer data windows will increase the recognition capability, but will also increase the delay. Next table shows the delay (time interval between the instant in which the turn has finished and the instant in which the orientation is estimated) for different values of the window size:

Table 2. Time delay for different window sizes

Window size (Number of measurements)	5	10	15	20
Interval of time	0,11 s	0,21 s	0,31 s	0,41 s

After some tests, we have seen that using a window size of 10 is enough to correctly recognize most of the gestures (nearly 100% recognition accuracy) and still provides a reasonable recognition speed.

b. Computing time

The computing time of the algorithm has been measured for type I positions. Using 15 measurements from the acceleration sensor to estimate the position of the smartphone, the average computing time was 273µs. This time is negligible compared to the acquisition time (0.02 s per measurement). So we can conclude that the computation of gesture recognition algorithm will not slow down the entire process and can be done in real time.

If the number of measurements increases, the computing time will increase almost proportionally. For 150 measurements, the computing time rises to 2.26 ms.

5.2 Offline validation of translation gestures with a dataset

a. Description of the dataset

To test the gesture recognition method, a dataset was collected with 14 subjects (8 males and 4 females), who repeated 10 times each gesture using a Google Nexus S smartphone. The starting and ending of each gesture was marked by pressing a button of the smartphone. Before collecting the gesture data, they were given a brief introduction about how to perform each gesture.

b. Performance statistics

Table 2 summarizes the recognition result of our approach over the collected dataset. An average of 93.2% is achieved. Most of the errors are due to the sensitivity of our approach to fluctuations of the signal. In particular, the test result demonstrates that more than half of the errors are due to the presence of a small fluctuation around zero (see an example in Fig. 6).

Table 3. Confusion matrix for the 6 translation gestures. The columns are recognized gestures and the rows are actual gestures. Average accuracy is 93.2%.

	Backward	**Forward**	**Left**	**Right**	**Up**	**Down**
Backward	94.3	0.7	2.9	0.7	0	1.4
Forward	0.7	94.3	1.4	2.1	0.7	0.7
Left	3.6	2.9	93.6	0	0	0
Right	3.6	0	0.7	94.3	0.7	0.7
Up	5.0	1.4	0	0	93.6	0
Right	0.7	5.0	4.3	0	0.7	89.3

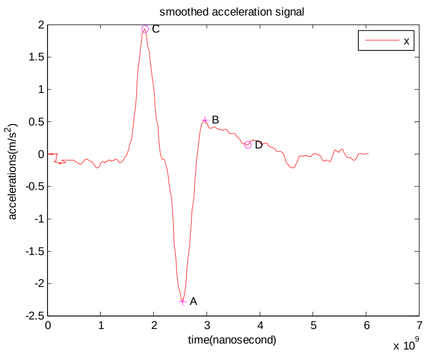


Fig. 6. Failure of pattern recognition due to fluctuation around zero between the real peak and valley (B and A are detected as peak and valley, instead of C and A)

6. Conclusion and future work

We focus on resource-constrained mobile phones and present algorithms to recognize turning and translation gestures in a user-independent manner. Our method utilizes embedded accelerometers and gyroscopes, which are commercially available in many mobile devices. The evaluation experiments of the turning gestures recognition system indicate that the time spent in the recognition process could be negligible compared with data sensing. The translation gesture recognition system has theoretically lower computational complexity than previous proposals implemented on mobile phones. The preliminary validation of both types of gesture recognition has shown a high recognition accuracy.

In order to further improve the recognition accuracy, we are currently making the translation gesture recognition more robust to fluctuations of the signals. Then it will be implemented on the mobile phone platform, to test its performance in real-time. As further work we are also planning to add some more gestures to our set (circles and spirals) to finally test the complete system with real users and get some feedback about the usability, accuracy, delay and energy consumption in real situations. Another possibility is to add some feedback to enhance user experience (adding sound, vibration or graphics).

References

[1] Pylvänäinen T (2005) Accelerometer Based Gesture Recognition Using Continuous HMMs. Pattern Recognition and Image Analysis, LNCS 3522: 413-430, Springer Berlin / Heidelberg

[2] Niezen G, Hancke GP (2008) Gesture recognition as ubiquitous input for mobile phones. In: Proc. of the Workshop on Devices that Alter Perception

[3] Joselli M, Clua E (2009) gRmobile: A Framework for Touch and Accelerometer Gesture Recognition for Mobile Games. In: 2009 VIII Brazilian Symposium on Games and Digital Entertainment, 141-150, IEEE

[4] Niezen G, Hancke GP (2009) Evaluating and optimising accelerometer-based gesture recognition techniques for mobile devices. In: AFRICON 2009, 1-6, IEEE

[5] Westeyn T, Brashear H, Atrash A, Starner T (2003) Georgia Tech gesture toolkit: supporting experiments in gesture recognition. In: Proc. of the 5th Int. Conf. on Multimodal Interfaces, 85-92, ACM

[6] Kauppila M, and Pirttikangas S, and Su X, and Riekki J. (2007) Accelerometer Based Gestural Control of Browser Application. In: Int. Workshop on Real Field Identification, 2-17

[7] Liu J, Wang Z, Zhong L, Wickramasuriya J, Vasudevan V (2009) uWave: Accelerometer-based personalized gesture recognition and its applications. Pervasive and Mobile Computing 5(6): 657-675, Elsevier

[8] Wu J, Pan G, Zhang D, Qi G, Li S (2009) Gesture recognition with a 3-d accelerometer. Ubiquitous Intelligence and Computing, LNCS 5585/2009: 25-38, Springer

[9] Cho SJ, Oh JK, Bang WC, Chang W, Choi E, Jing Y, Cho J, Kim DY (2004) Magic wand: a hand-drawn gesture input device in 3-D space with inertial sensors. In: 9th Int. Workshop on Frontiers in Handwriting Recognition, 106-111, IEEE

[10] Kauppila M, Inkeroinen T, Pirttikangas S and Riekki J (2008) Mobile phone controller based on accelerative gesturing. Adjunct Proceedings Pervasive: 130-133

[11] Hofmann FG, Heyer P, Hommel G (1998) Velocity profile based recognition of dynamic gestures with discrete hidden markov models. Gesture and Sign Language in Human-Computer Interaction, LNCS 1371/1998: 81-95, Springer

[12] Schlömer T, Poppinga B, Henze N, Boll S, (2008) Gesture recognition with a Wii controller. In: Proc. of the 2nd Int. Conf. on Tangible and embedded interaction, 11-14, ACM

[13] Mäntyjärvi J, Kela J, Korpipää P, Kallio S (2004) Enabling fast and effortless customisation in accelerometer based gesture interaction. In: Proc. of the 3rd Int. Conf. on Mobile and Ubiquitous Multimedia, 25-31, ACM

[14] Kela J, Korpipää P, Mäntyjärvi J, Kallio S, Savino G, Jozzo L, Di Marca S, (2006) Accelerometer-based gesture control for a design environment. Personal and Ubiquitous Computing 10(5): 285-299, Springer-Verlag