Creating, Composing, and Recognizing Multisensor Gestures in Mobile Devices

Miroslav Takács and Valentino Vranić
Institute of Informatics and Software Engineering
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
Ilkovičova 2, Bratislava, Slovakia
t.miro99@gmail.com, vranic@stuba.sk

Abstract—**The work reported here focuses on creating, composing, and recognizing multisensor gestures and their use in terms of remotely controlling another device. The approach allows to compose gestures synthetically, i.e., by recording complex gestures directly as they are performed, and analytically, i.e., by selecting and composing existing gestures to define more complex ones. Analytical gesture composition doesn't have to be sequential: the gestures can be superposed. We explored this in a setting of using a mobile device to control a remote device. In our case, this was a smartphone controlling a computer, which we assume to be most probable setting in practice. For this, a client-server system was developed consisting of the application running on the Android mobile device using the accelerometer, proximity sensor, touchscreen, and ambient light sensor to capture gestures. The server-side application that runs on a computer was developed to store and recognize gestures. The preliminary evaluation results are promising exhibiting the 85% rate in the ability of users to replicate gestures not created by them. According to informal observations, the user seem to be quite comfortable working with their own multisensor gestures.**

Keywords*—*mobile device; multisensor gesture; gesture composition; gesture recognition; dynamic time warping; Android

I. Introduction

Today's mobile devices contain a number of integrated sensors. Typically, these include accelerometer, gyroscope, GPS, compass, and proximity sensor. The touch display can be considered as yet another sensor: perhaps the one mostly used to control applications. The input to other sensors is also used to control applications implicitly, such as when the application changes its behavior based on location obtained from the GPS sensor, or explicitly, such as controlling a car racing game using the accelerometer or magnetometer.

The mobile device sensors work simultaneously and this can be used as a basis for establishing a complex, multisensor control. In effect, this requires users to become comfortable with quite sophisticated actions with their devices such as moving their device down while sliding with a finger across the display. These are actually complex, multisensor gestures.

The idea of allowing users to define their own gestures and attach the behavior of their choice to them is not new and not even limited to mobile devices. One or multifinger gestures can be defined for most contemporary touchpad devices. However, employing multiple sensors at once to activate desired behavior is explored only to a limited extent.

Here we go one step further and propose an approach to defining complex multisensor gestures in mobile devices and their recognition. The approach allows to compose gestures synthetically, i.e., by recording complex gestures directly as they are performed, and analytically, i.e., by selecting and composing existing gestures to define more complex ones. We explored this in a setting of using a mobile device to control a remote device. In our case, this was a smartphone controlling a computer, which we assume to be most probable setting in practice.

Section II takes a look at gestures in mobile devices. Section III proposes a new approach to treating gesture composition. Section IV explains the recognition of composed gestures. Section V presents the evaluation results. Section VI discusses related work. Section VII concludes the paper.

II. Gestures in Mobile Devices

Mobile devices receive the input from the user using sensors. Many smartphones do not have hardware keyboard and they use a software keyboard on the touch screen instead. Thus, the touch screen is the most used and most important sensor in mobile devices. It responds to two-dimensional gestures, and provides output in the form of images or text. Another possible way of obtaining input from the user is by using motion gestures [1].

A gesture should be easy to remember and easy to use. It should not be overly complicated, so that users can replicate it quick enough. Gesture recognition should be efficient, so that the system can make actions in real time. The error rate in gesture recognition must be minimal, i.e., the system must be capable of differentiating similar gestures.

A. *Sensors for Multisensor Gestures*

Multisensor gestures are virtually superposed single-sensor gestures. The accelerometer as a very common sensor in mobile devices provides a plenty of opportunities to be employed in gestures. A gesture based on the accelerometer activation is in fact a trajectory obtained by a continuous measurement of the orientation of the mobile device in space. More specifically, the accelerometer measures the effect of the gravitational force on the device in three coordinate axes. The measured values are relative to the free fall. A typical use of the accelerometer is to automatically adjust the display orientation depending on how the user holds the mobile device.

Other sensors can also be used in building complex, multisensor gestures. The proximity sensor detects whether there is an object near the mobile device without the need for a physical contact. The output of this sensor is the distance of the object, which is typically zero if an object was detected, and non-zero if it wasn't. The touch sensor can be used for obtaining the information that indicates the number of fingers on the touchpad. The ambient light sensor detects the surrounding brightness near the mobile device. Gestures using this sensor are dependent on the level of light around the device. The output of the sensor could be used as a parameter, for example setting the brightness level. Gyroscope can be used similarly. Output of this sensor is value depending on the rotation of the device. For example, gyroscope sensor can be used for setting volume, depending on rotation of the device.

B. *Gesture Types*

According to the type of the sensor employed, the gestures can be categorized as touch (two-dimensional) or motion (three-dimensional). The touchscreen sensor is used as the main input device. It detects the position of the finger or other touching object (stylus) on the screen. Using the touch screen is simple and intuitive and therefore it has become very popular in mobile devices.

Touch gestures can also use multiple touches at once, which is known as multi-touch: "an interface technology that enables input through pressure and gestures on multiple points on the surface of a device" [2]. This allows users to use the touch screen gestures. The simplest is the tap gesture. Other commonly used gestures are scrolling, zooming, and rotating. The term gesture can also refer to making a specific shape with the finger on the touchscreen. The mobile device touchscreen can be used to control the pointer on a computer, i.e., as a remote touchpad, or for detecting two-dimensional gestures that start different actions.

Motion gestures are created by moving or rotating the mobile device in space. Motion gestures have an advantage in users not having to focus their sight on the mobile device screen. As we already pointed out, the accelerometer is probably the best sensor to use with motion gestures. A typical motion gesture is the shake gesture.

According to the complexity, the gestures can be categorized as elementary or composed. From the perspective of gesture recognition, an elementary gesture can be defined as a sequence of the output values of one sensor that constitutes one indivisible action. For example, moving a mobile device to the right is an elementary gesture. Elementary gestures are usually short. They can be based on any sensor, not only accelerometer.

A composed gesture consists of other gestures, be they elementary or composed gestures. There are two ways to compose gesture: serial and parallel (superposed) composition. These are going to be explained in the further text.

III. Composing Gestures

As has been shown, users can perform the gesture they created with a greater success than the predefined ones [1]. A new gesture can be created directly by recording it or by composing already recorded gestures.

A. *Gesture Representation*

Analogically to a one-sensor gesture (Section II), a multisensor gesture can be defined as a combination of the output values of the sensors within a certain time period. These output values are recorded in regular time intervals. The saved gesture must contain the information about the sensors that have been used. This is important for sending the gesture over a network and for comparing gestures. Furthermore, the gesture has to contain the actual output values of the sensors represented as decimal values. Optionally, gestures can have a name attribute for easier identification.

Figure 1 shows output values of the accelerometer sensor. The y-axis in graphs shows the acceleration in space detected by the sensor measured in m/s^2. The x-axis represents the time in seconds. The output values of the sensors are recorded in regular time intervals. This figure is a screenshot taken from the created application. In the x-axis, there is visible activity of the accelerometer caused by moving the mobile device to the right.

B. *Composing Gestures Synthetically*

Multisensor gestures can be recorded directly on a mobile device by simultaneously recording the output of the involved sensors. This way, individual sensor gestures are composed in a synthetic manner.

In our application, before the actual recording, the sensors that are going to be used have to be selected. The recording can start by pressing a button, but a better alternative is to start it automatically when device detects the activity such as motion. The recording can be stopped the same way: when device is no longer moving. The output values of the sensors are stored in an array or some collection in regular intervals. Our experiments indicate that the interval of 50 ms is quite usable. Longer intervals decrease the rate of success gesture recognition, while smaller intervals cause higher battery and network usage.

There is no guarantee that the sensors will provide data at the same time, so there is a need to synchronize the obtained data. The output of the touch screen sensor can be obtained directly at any time. Obtaining data from others sensors works differently. The accelerometer and ambient light sensors provide data at intervals, which cannot be set to a specified value. The solution is to retrieve the data from these sensors at shorter intervals and get the average of the values every 50 ms. The proximity sensor is interrupt based. This means that we get a proximity event only when the proximity changes, so we use the last detected value. The output of this sensor is usually only a value of 0 or 10. In our experience, this approach worked well and, in addition, it helped to reduce noise (a deviation from the real value caused by the sensor inaccuracy) in the accelerometer sensor.

C. *Composing Gestures Analytically*

Two existing gestures can be composed to create a new gesture by picking them from the database and declaring their composition analytically. In a simple case, the gestures to be composed would be based on one and the same sensor. For example, the result of composing the gesture of moving the

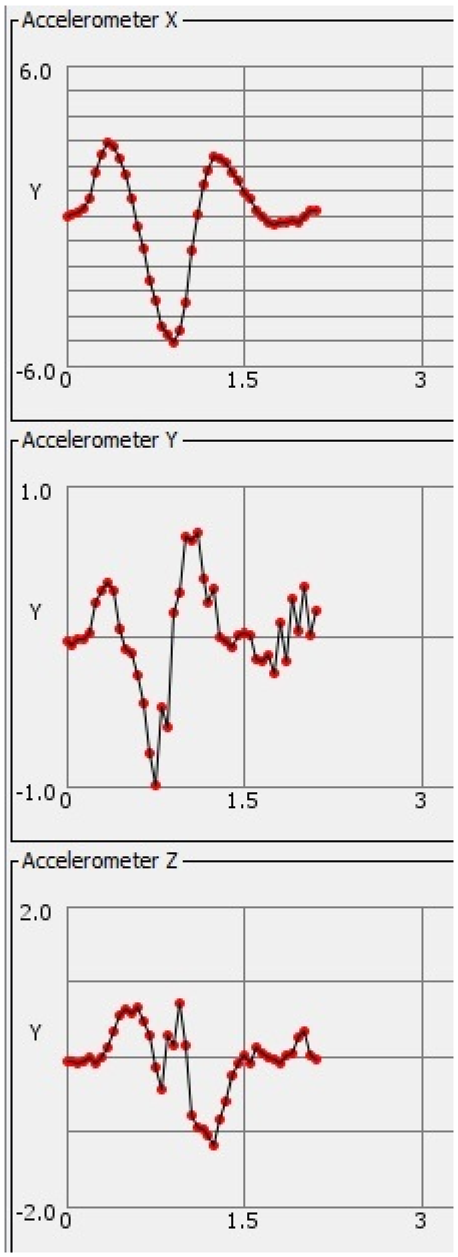

Fig. 1. The accelerometer output when moving the mobile device to the right.

device down with the gesture of moving the device to the right is a composed gesture of moving the device in a letter L fashion. However, in a general case, each of the gestures to be composed may involve multiple, possibly different sensors. An example is composing the movement to the right with the proximity sensor detecting (or not detecting) a near object.

Analytical gesture composition doesn't have to be sequential: the gestures can be superposed. In this case, the output of each sensor involved in all gestures is considered in parallel in time to all other sensors. For this, it is necessary the gestures are based on different sets of sensors.

Figure 2 shows an example of a composed gesture. The first part of the gesture is the movement to the right, the second part is the movement to the left, but with one finger on touchscreen. The y-axis in the first three graphs shows the acceleration detected by the sensor, measured in m/s^2, while in the fourth graph it shows the number of detected touches on the touchscreen. The x-axis represents the time in seconds.

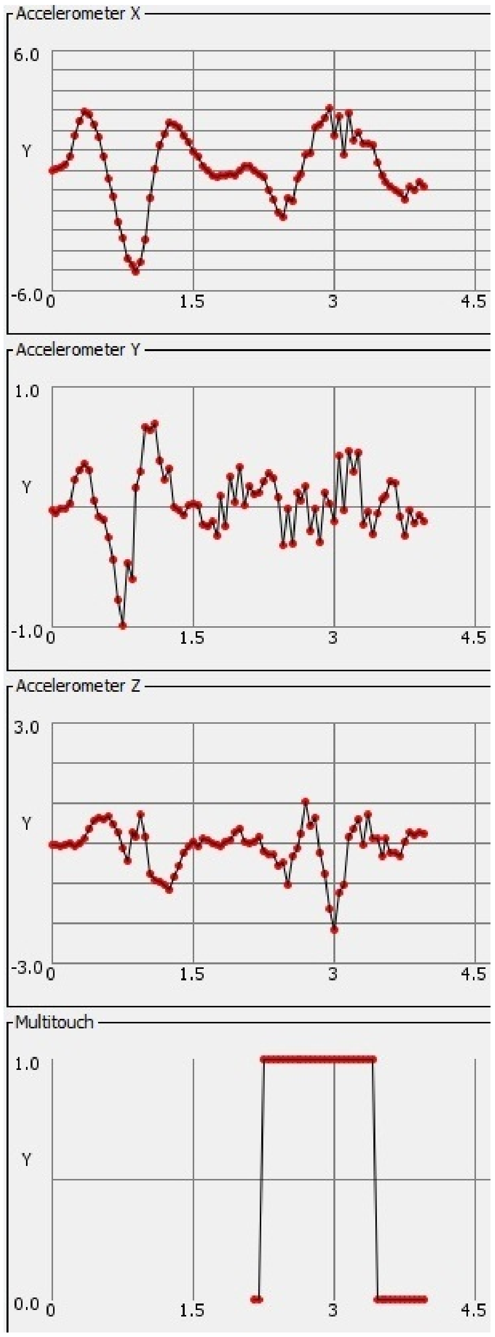

Fig. 2. An example of a composed gesture.

IV. Recognition of Composed Multisensor Gestures

The recognition of single-sensor gestures isn't trivial either, but the recognition of composed multisensor gestures is far more complicated. It involves several aspects that are going to be discussed in this section.

A. Filtering Out the Gravity Noise

The values produced by the accelerometer are equal to almost 1 G even when the mobile device remains motionless because it is affected by the gravity. The gravity noise has to be removed out of the raw acceleration data coming from the 3-axis accelerometer. The solution is to use a high-pass filter, which will eliminate the gravitational force from the output [3]. The Android platform offers an easy solution to this problem in a form of a virtual sensor that produces data from the accelerometer without the gravity force.

B. Identifying the Gesture Start and End

To recognize gestures, we need to determine the time of its start and end. Here, the technique by Hwang and Lee [4] can be employed. In a motion gesture, it is necessary to calculate the kinetic energy of the device E_i at a given moment i by the following equation:

$$E_i = (x_i)^2 + (y_i)^2 + (z_i)^2$$

where x_i, y_i, and z_i indicate the acceleration in the corresponding coordinate. We calculate an average of N last values. If this value is greater than the certain threshold, the beginning of the gesture if found. Similarly, we can determine the end of the gesture by the average value of the energy falling below the threshold.

If the gesture does not involve the accelerometer, but the touch screen, the gesture lasts as long as at least one finger is on the touch screen.

C. Comparing Gestures Using Dynamic Time Warping

Dynamic time warping (DTW) is an algorithm for comparing two temporal sequences that may vary in time or speed used, for instance, in speech recognition [5]. This algorithm is suitable also for gesture recognition. DTW compares the input gesture with the stored gestures and evaluates which of the stored gestures is the most similar to the input gesture.

Constantly recording the sensor data, gesture recognition, and communication with the remote device (to be controlled by gestures) consumes battery and engages significant processing power of the mobile device. The optimal solution appears to be to record sensor data with the mobile device, send it to the computer, and recognize gestures there.

DTW can be sped up by adhering to some constraints. When constraints are employed, the DTW algorithm finds the optimal warp path through the constraint window. However, the globally optimal warp path will not be found if it is not entirely inside the window. But this is not a problem when comparing gestures. In addition, by the use of the band increased recognition accuracy, different gestures achieved a much greater distance.

When searching for the shortest distance between two sequences, DTW uses a two-dimensional array for storing calculations. In our case, this can be reduced: to calculate a row in this array, the algorithm needs only the values from the previous row. That means that when comparing two gestures with lengths n and m, we can reduce the space complexity from $O(n \times m)$ to $O(2 \times n)$.

The gesture comparison is performed using the DTW algorithm. The similarity between two points in a time span of gestures (the output values of the sensors at a given time) is computed as an average of the distances of the data from all sensors that are employed. The result of the DTW algorithm is a value that represents the similarity between the input gesture and one stored gesture. Our application stores this result and after comparison with all stored gestures, the gesture with the lowest similarity value is evaluated as the most similar to the input gesture, but only when this similarity value does not exceed a given threshold. This threshold is useful when none of the stored gestures is actually similar to the input one, so in that case no similar gesture is found.

Two gestures are automatically (without employing DTW) considered to be different when the input gesture does not involve a sensor that the stored gesture involves or when the difference in the gesture duration is more than two times. This prevents useless comparisons.

The value of the similarity between the two accelerometer output values can be expressed by the following formula:

$$result = (x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2$$

For a more precise comparison, the similarity between the output values of the other sensors should be in a similar range. For the ambient light sensor, the following formula is used:

$$result = \sqrt{|l_1 - l_2|}$$

where l_1 a l_2 are the output values of the ambient light sensors.

For the proximity sensor, the following formula is used:

$$result = |p_1 - p_2|$$

where, for most devices, the output values p_1 and p_2 may take only the value of 0 or 10.

The touchscreen sensor is used for detecting how many fingers are present on the screen. The similarity between two output values of this sensor is a difference between the touches multiplied by 10, in order to provide this sensor sufficient weight when comparing gestures. The following formula is used:

$$result = 10 \times |m_1 - m_2|$$

D. Communication Between Devices

Our experiments have been conducted in a setting of using a mobile device to control a remote device. More specifically, we used a smartphone to control a computer, which we assume to be most probable setting in practice. The mobile and remote devices must exchange information and Wi-Fi seems to be the best for this purpose. Communication can take place using the UDP protocol, which is faster, but does not guarantee delivery, or TCP, which is more complex, but reliable. As a compromise, in the approach presented here, both protocols are used: TCP for important messages, and UDP for data.

In order to send messages to the remote device, a server application must be running on it and listening at a specific port. The client application on the mobile device can then find the remote device running the server application using broadcast messages. The computer will respond to the broadcast with a message that includes its name and IP address. Subsequently, the user selects one of the available servers from the client application establishing the communication with it.

The largest amount of data that is transmitted between devices are in the sensor data. These data are recorded 20 times per second and stored as a byte array. They can be sent continuously using the UDP protocol and gesture recognition can be performed on the computer. Another option is to recognize the start and end of the gesture on a mobile device and send only the data that corresponds to the gesture using the TCP protocol. This reduces the number of sent messages and guarantees that they are received by the server. Furthermore, the number of transmitted messages is reduced thanks to Nagle's algorithm, which is employed to combine small messages and send them all at once [6].

V. Evaluation

A mobile application was developed as a demonstration and experimental setting for the approach of creation and recognition of multisensor gestures and their composition. The application was developed for mobile devices with the Android operating system and uses the accelerometer, proximity sensor, touchscreen, and ambient light sensor.

The mobile application has two modes. The first one is gesture recording, which allows user to record gestures with their mobile devices and store the gestures within the server application on a computer.

The second mode is gesture recognition. When a gesture is performed, the application will find the most similar gesture among the stored ones and activate the behavior assigned to that gesture, if any.

The server application allows user to modify or remove the stored gestures. There is also an option to analytically compose gestures (two at a time), including the newly created ones. Each gesture can be assigned an action, such as to show a notification or emulate pressing a key. Gestures are visually represented in application by graphs of sensors data. The application also contains a tab that shows the real-time data received from the mobile device.

To evaluate the usability of the approach, four participants were asked to perform ten already recorded gestures. First seven gestures were motion gestures, which used only the accelerometer sensor: moving in four directions, and drawing the letters L, O, and M. The next three gestures were multisensor gestures: holding a finger on the touch screen while moving left and right, and drawing the letters L while the proximity sensor detects the proximity of an object. All participants had a moderate experience with mobile devices.

All gestures were repeated until they were successfully recognized. The process of performing gestures was explained to each participant. The results summarized in Table I show that the participants were able to perform gestures with a success rate of 85%. The numbers in the cells represents how many attempts were needed until a successful gesture performance has been achieved. The numbers in the last column represent the success rate of each user individually.

After performing the predefined gestures, the participants had an opportunity to create their own gestures. The participants had no problems in performing their own gestures.

The gesture recognition can be improved by experimenting with different thresholds for determining the gesture start and end. The thresholds should be increased if the user is moving the mobile device without intention to perform a gesture, but system detects this as a gesture. The threshold value for similarity can also be set by users. When the most similar gesture is found, this value is used to determine whether the gesture that was found is evaluated as identical.

The recognition accuracy can by increased greatly by using multiple gesture samples in gesture creation. However, this would increase the number of gesture comparisons, but the running time of DTW with the gesture database of ten gestures was found to be only 16 ms.

VI. Related Work

Hwang and Lee [4] addressed accelerometer based gestures. They managed to treat simple spatial gestures such as moving a mobile device in one direction. Similar results are reported by Liu et al. [7]. Parikh [8] presents a more complex treatment of accelerometer based gestures for zooming, rotating, scrolling, and shaking.

Ruiz, Li, and Lank [1] describe the gestures for some typical mobile device actions, such as receiving or declining a call proposed by volunteers, but without taking into account the technical limitations of sensors. Unlike in other approaches, the approach proposed in this paper takes into account multiple sensors and enables to create gestures based on the input from these sensors in both analytic and synthetic way.

VII. Conclusions

Today's mobile devices are easily portable and include a number of sensors. Mobile applications make extensive use of individual sensors. The ability to capture gestures by multiple sensors in mobile devices opens new opportunities in the field of device remote control and improves users comfort. The main advantage of using gestures is that the user can focus directly on the point of interest instead of having to follow the screen of the mobile device.

The work reported here focuses on creating, composing, and recognizing multisensor gestures and their use in terms of remotely controlling other devices. The approach allows to compose gestures synthetically, i.e., by recording complex gestures directly as they are performed, and analytically, i.e., by selecting and composing existing gestures to define more complex ones. Analytical gesture composition doesn't have to be sequential: the gestures can be superposed.

We explored this in a setting of using a mobile device to control a remote device. In our case, this was a smartphone controlling a computer, which we assume to be most probable setting in practice. For this, a client-server system was developed consisting of the application running on the Android mobile device using the accelerometer, proximity sensor, touchscreen, and ambient light sensor to capture gestures. The server-side application that runs on a computer was developed to store and recognize gestures.

TABLE I. THE RESULTS OF THE GESTURE RECOGNITION.

gesture	right	left	up	down	L	O	M	right + touch	left + touch	L + far	accuracy
User A	1	1	1	1	1	1	2	1	1	1	90%
User B	1	1	1	1	1	3	1	1	2	1	76%
User C	1	1	1	1	1	1	3	1	1	1	83%
User D	1	1	1	1	1	1	2	1	1	1	90%

The preliminary evaluation results are promising exhibiting the 85% rate in the ability of users to replicate gestures not created by them. According to informal observations, users seem to be quite comfortable working with their own multisensor gestures. We expect a more thorough evaluation to bring new findings that could be fed into the gesture composition and recognition process.

The multisensor gestures can be used to control remote devices, as demonstrated in our setting. The actions controlled this way be as simple as controlling slide switching in a presentation, but gestures have a greater potential, e.g., to be used for complex actions in computer games or in complex event processing in general [9], [10], in sophisticated authentication, or to simplify certain tasks for physically disabled persons. To target different mobile platforms, multiplatform (cross-platform) approaches to mobile application development could be used [11].

ACKNOWLEDGMENT

The work reported here was supported by the Scientific Grant Agency of Slovak Republic (VEGA) under the grant No. VG 1/1221/12. This contribution/publication is also a partial result of the Research & Development Operational Programme for the project Research of Methods for Acquisition, Analysis and Personalized Conveying of Information and Knowledge, ITMS 26240220039, co-funded by the ERDF.

REFERENCES

[1] J. Ruiz, Y. Li, and E. Lank, "User-defined motion gestures for mobile interaction," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2011, pp. 197–206.

[2] M. Rouse. (2011) Multi-touch definition. [Online]. Available: http://searchconsumerization.techtarget.com/definition/multi-touch

[3] Android Open Source Project. (2014) Android developers – reference. [Online]. Available: http://developer.android.com/reference/

[4] K. Hwang and J. M. Lee, "An implementation experience of accelerometer-based gesture recognition with android smartphone," *International Journal of Advancements in Computing Technology*, vol. 5, no. 12, 2013.

[5] G. Al-Naymat, S. Chawla, and J. Taheri, "SparseDTW: A novel approach to speed up dynamic time warping," in *Proceeding of 8th Australasian Data Mining Conference, AusDM '09*. Melbourne, Australia: ACM, 2012.

[6] J. Nagle, "Congestion control in ip/tcp internetworks," *ACM SIGCOMM Computer Communication Review*, vol. 14, no. 4, pp. 11–17, Oct. 1984.

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

[8] N. Parikh, "Accelerometer based motion gestures for mobile devices," Master's Project, San Jose State University, 2008, http://scholarworks.sjsu.edu/etd˙projects/103/.

[9] J. Lang and J. Janík, "Reactive distributed system modeling supported by complex event processing," in *Proceedings of 3rd Eastern European Regional Conference on the Engineering of Computer Based Systems, ECBS-EERC 2013*. Budapest, Hungary: IEEE Computer Society, 2013.

[10] J. Lang, M. Jantošovič, and I. Polášek, "Re-usability in complex event pattern monitoring," in *Proceedings of 10th Jubilee International Symposium on Aplied Machine Intelligence and Informatics, SAMI 2012*. Herľany, Slovakia: IEEE, 2012.

[11] Ľ. Staráček and V. Vranić, "MDA based multiplatform mobile application modeling with platform compliant user interfaces," *INFOCOMP Journal of Computer Science*, vol. 13, no. 2, pp. 34–43, 2014.