International Journal of User-System Interaction 12(1) 2019, 21-38

Usability Evaluation of Gesture-Based Interactions using a Smart Watch

Ioana-Crinela Potinteu, Teodor Ștefănuț

Technical University of Cluj-Napoca
Computer Science Department
26-28 G. Baritiu, 400027, Cluj-Napoca
E-mail: crinela.potinteu@student.utcluj.ro, teodor.stefanut@cs.utcluj.ro

Abstract. This paper describes the testing methodology applied and the results obtained in evaluating the usability of a system for gesture recognition based on a smartwatch and a mobile phone. The data acquisition is performed using a single accelerometer with 3 axes, using an approach based on button segmentation. For our testing purposes, the system has been set up to recognize 4 arm gestures, that represent specific commands for a music player that is running on the mobile phone. This setup has also offered us the means to provide users with easy to recognize feedback on how their actions have been interpreted by the system. Inspired from real-life scenarios, our testing methodology has considered three major contexts in which the users should be able to easily interact with the system: while sitting, when walking and while running.

Keywords: human-computer interaction, accelerometer, gesture recognition, smartwatch, classification, dynamic time warping

1. Introduction

Gesture recognition as defined in (Ahmad et al, 2011) is considered the process of understanding and classifying purposeful people movements. Gestures described by hands, arms or upper body, facial and head movements can be tracked through different means (images, sensors, etc.), analyzed and matched on specific software commands, depending on the purpose of the system that uses the recognition.

In our research, we have focused on the capabilities of smartwatches, which evolved surprisingly in the past few years, to record hand performed gestures. Currently, wearables like smart bracelets and watches are widely used for health monitoring, playing music, browsing the internet, control other smart devices (phones, computers, automated houses), so they are constantly and naturally integrated into everyday lives. For most of the

aforementioned activities, the interaction with the user is made by touching the relatively small screen or pressing the physical buttons of the device. This approach requires specific visual attention from the user, who very often needs to interrupt the ongoing activity in order to interact with the device.

For very common situations like riding the bike, jogging, or even walking at a fast pace, this kind of interruptions can be disrupting, so relying on interaction techniques that do not require so much focused attention from the user is recommended. Consequently, we have designed a system that is using the standard accelerometer that is already integrated into almost all the wearables devices, to record the movements of the user and to identify specific predefined gestures which can later be mapped on software commands. It turned out that off-the-shelf devices can provide very good input for an arm gesture recognition system.

For gestures monitoring, we have used a smartwatch equipped with a 3-axis accelerometer and a mobile phone with an Android operating system. As running the recognition algorithm on the smartwatch has not been possible due to hardware limitations, the values obtained from the smartwatch's accelerometer are transmitted to the mobile phone where the processing algorithm is implemented. For the case study presented in this article, we have defined a dictionary of 4 gestures which have also been associated with specific actions used to control a music player that is running on the smartphone. As a result, the user can start or stop the music and skip to the next or previous song by describing the corresponding gesture with the arm on which is wearing the smartwatch.

The two devices involved in our system, the mobile phone, and the watch, are exchanging information through a Bluetooth connection which is kept active throughout the entire experiment. The application running on the smartwatch, which records the data registered by the accelerometer, must be permanently in the foreground and the application running on the smartphone can be either in the foreground or in background. In order to give a command using a gesture, the user must follow the next steps: (1) bring the forearm parallel to the ground; (2) press a certain button and (3) perform the gesture. All the gestures are chosen in such a way that at the end of the gesture the arm is in the same position as it was at the beginning (step 1). The button press is used as a record trigger in our test system. After pressing the button the accelerometer data is recorded continuously for three seconds and the information is sent to the phone. For best results, all the users that participated

in our test have been instructed to keep the arm in the finish position for the rest of the time if they finish the gesture in less time than three seconds.

When all the data recorded by the accelerometer for a gesture has been received on the phone, the recognition algorithm is launched. We have prepared a database of 240 labeled templates which have been collected from a single person. As the maximum duration of a gesture can be of three seconds, all the templates have the same length. The 240 templates represent 15 labeled templates for each of the four gestures with each arm, while the user was sitting and running. The recognition algorithm implements One Nearest Neighbor Classifier to find the best matching template in the database for the new values that come from the accelerometer. The tests we performed have shown that the number of label templates that are used by the classifier has a great impact on the system's accuracy. Using 5 templates for each gesture we obtained an accuracy of 81.75%, for 10 templates is obtained a value of 88.83% and for 15 templates the accuracy is 92%. However, using a higher number of templates has a strong negative impact on performance, so a compromise between accuracy and response time needs to be made in order to maximize user's satisfaction.

The article is structured as follows: in the next section, we have briefly discussed relevant related work. In section three we are presenting a general overview of the system used in usability evaluation, while section four details the gesture recognition approach that we have used. Section five presents a detailed analysis of the data recorded from the accelerometer, while section 6 describes the methodology implemented for the system's evaluation and discusses the obtained result. Finally, the conclusions are introduced and future improvements are proposed.

2. Related work

In 1989 (Sturman et al, 1989) describes one of the first algorithms designed for gesture recognition, which has been used data gloves as an input device. Gestures described and recorded using the gloves have been mapped on software commands used to manipulate virtual objects. Since then many approaches to gesture recognition problem that used data gloves appeared, for instance (Kim et al, 2001) and (Tubaiz et al, 2015).

Another common approach in gesture recognition research is based on computer vision methods, as described in (Das et al, 2017) and (Dokmanic et

al, 2014) where ultrasonic depth imaging is used. Monocular cameras were also used by (Bhuyan et al, 2013) while for more complex environments binocular cameras achieved a better accuracy as presented by (Jiang et al, 2018), (Chen et al, 2017) and (Feng et al, 2017).

In literature, there are several systems that use an accelerometer and a gyroscope for gesture recognition. One of the most popular systems that use only a 3-axis accelerometer is uWave, described in (Liu et al, 2009). It is a user-dependent system that requires the user to perform all the gestures before using it. It was defined as a dictionary of 8 gestures and for each one, a template is stored in the database. The system searches through the templates' database and Dynamic Time Warping algorithm is used to measure the distance between the new values and the template. The gesture is recognized by assigning the label of the template that gives the minimum distance.

uWave also includes a database adaptation phase. This phase was included because there were observed substantial variations between the gestures performed by the same user in different days. For this process, uWave will keep two templates performed on different days for each gesture. The recognition algorithm will compute the distance between the new input and both templates and will take the smaller distance as the matching cost between the input and the dictionary gesture.

In (Akl et al, 2011) it is proposed a solution that starts with a training phase that collects data from a 3-axis accelerometer and uses Dynamic Time Warping and Affinity propagation to cluster the data. Each cluster will be represented by one member called "exemplary". The article proposes a large dictionary that contains 18 gestures. In the testing phase, Dynamic Time Warping will be used to compute the similarity between the traces from the database and the real input from the accelerometer. This algorithm will select the traces that are the closest to the new input data. The selected traces and the unknown trace will be all projected onto the same lower-dimensional subspace, so they will all have the same duration. In this subspace, the recognition problem becomes an l_1-minimization problem. Both user-dependent and user-independent approaches are presented and compared, better results were obtained for the user-dependent approach.

A user-independent approach for detecting 5 unremarkable and fine-motor finger gestures is presented in (Wen et al, 2016). It combines the data from the 3 axes (x, y, and z-axis) of an accelerometer, a gyroscope and a linear accelerometer integrated into a Samsung Galaxy Gear smartwatch. From the

data 7 statistical features are computed from a 1-second sliding window: mean, standard deviation, max, min, and 3 quantiles. To these features, there will be added the lower 10 power bands resulting by applying a Fast Fourier Transform for the same 1-second window. The performance of the system was tested using different basic classifiers: support vector machine, Naive Bayes classifier, Logistic Regression, and K-Nearest Neighbors.

Another user-independent system that is based on features extraction is presented in (Khan et al, 2012). The features vectors contain Haar coefficients computed from the accelerometer data recorded for x, y and z axes. Support vector machine classifier that was trained offline using the uWave gesture library was used to build a prototype of the system.

An interesting approach based on global alignment kernels is presented in (Porzi et al, 2013). It is proposed a user-independent approach capable to recognize 8 distinct gestures, using a 3-axis accelerometer integrated into a smartwatch. The system also contains two vision-based modules, one for identifying wet floor signs and other for predefined logs. These modules use the camera of a smartphone that the user wears at his or her neck. The target of this system are people with visual impairments.

Inspired from the approach proposed by uWave, our system has been built as a user-independent gesture recognition application which uses multiple templates for each dictionary gesture. Dynamic Time Warping is used to compute the distance between all the templates and the new input gesture and One Nearest Neighbors classifier will choose the smaller distance.

3. Implementation of the Test System

As mentioned before, our gesture-based user interaction system has two main components: the smartphone and the smartwatch. In order to easily register the templates for each gesture and to configure different parameters for our usability tests, we have also added two more components (see Figure 1):

Web API + Database – is a component that allows us to store the data on a remote server and not locally on specific devices. In this manner, whenever deemed necessary, it was easy to update all the templates on all devices.

Angular application – a web-based interface that has been created for data visualization and analysis.

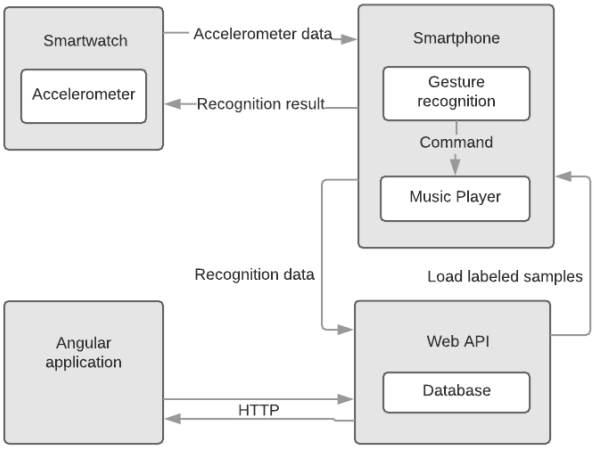

Figure 1.Communication architecture of the system

For the experiments, we have used a Garmin Forerunner 935 smartwatch that has an integrated 3-axis accelerometer. It was directly programmed to record the data from the accelerometer for 3 seconds after the user has pressed a certain button and then to send the data to the mobile phone via a Bluetooth Low Energy connection. To program the smartwatch, Garmin Company offers a free SDK called Connect IQ. The SDK has a special class to obtain the data from the accelerometer called **AccelerometerData**. Using this class, we obtain a maximum of 25 samples per second for each axis. The acceleration is measured in millig (mG) units, where 1G (1000 mG) is equal to earth's gravity acceleration, which by definition is equal to 9.78033 m/s^2.

In order to preserve energy and to ensure optimized user experience, the Connect IQ SDK imposes many constraints on the applications that can be developed for the smartwatches. One of the most important is related to the maximum execution time of a user-defined function. The device uses a watchdog timer that counts down from some initial value to zero. The embedded software selects the initial value (for Forerunner 935 is 120 000) and restarts it periodically. If the counter reaches zero before being restarted, the reset signal is asserted and the processor will be restarted. The execution time of Dynamic Time Warping algorithm is greater than the maximum execution time allowed by the watchdog counter, for this reason, the data from the accelerometer is sent to the phone where all the processing is performed.

An important aspect for data gathering and processing is related to the accelerometer axis directions when the smartwatch is worn on the left and on the right hand (see Figure 2). Analyzing the values obtained for the X-axis when the user performs the same movement but with a different arm, we can see that they are distinct.

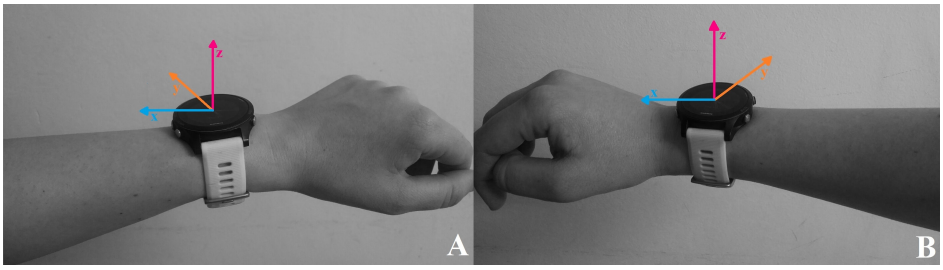

Figure 2. A: smartwatch worn on the left hand and the accelerometer axes direction B: smartwatch worn on the right hand and the accelerometer axes direction

For instance, if the user has the left arm in a position similar to the one in Figure 2A and he or she moves his arm in front along the X-axis, the accelerometer will record negative values of the acceleration on the X-axis. If the user has the right arm in a similar position and moves his arm in front along the X-axis, positive values will be recorded by the accelerometer for the X-axis. As a result, the templates that are used in the recognition process for each of the gestures are different for each hand. The user has the possibility to indicate the hand on which he or she is wearing the smartwatch in the application that is running on the smartphone, information that is used as a parameter in the gesture recognition algorithm in order to compare the registered values only with the relevant templates.

The user indicates to the application his intention to perform a gesture by pressing a predefined hardware button and thus marking the start of the three seconds interval. The smartwatch will start recording the accelerometer data continuously for the next 3 seconds and will send the information to the smartphone. In our tests, we have used a BlackBerry Priv model that runs Android OS. Both templates' databases for right and the left hand will be loaded only once, when the application on the smartphone is started, by calling a Web API that we have implemented. Using the proper templates' database, the recognition will be performed and the result will be used to control a music player implemented inside the same application.

4. Gestures recognition algorithm

For our testing purposes, we have defined a dictionary of four gestures, which have been bind to the *play*, *stop*, *skip to next song* and *skip to previous song* functionalities of a music player (see Figure 3).

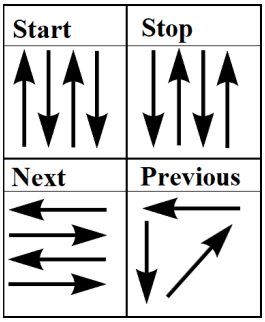

Figure 3.The dictionary of 4 gestures

For each of the above gestures, we have built a template database recording each gesture 15 times, with the help of the same person and for each hand: left and right. Each of the recordings above (*gestures* * *each hand*) has been performed twice: once while sitting and once while jogging. Using the Web API, all the templates have been saved on the remote server and downloaded on the test smartphone when the application has been started.

During the experiment, the data recorded by our test subjects have been transferred by the watch to the phone where the gesture recognition algorithm was executed. Before actual processing, the **Context detection** unit had the role to establish which templates of the available ones will be used for recognition of the current gesture: the ones recorded while the subject was sitting or the ones recorded while the subject was jogging. This component implements a very simple algorithm that analyses only the values recorded on the Z-axis. This has been proved necessary as important differences were observed for the values recorded for Z-axis when the users performed the gestures while sitting versus while they were jogging. All the components of the system and the data-flow can be visualized in Figure 4.

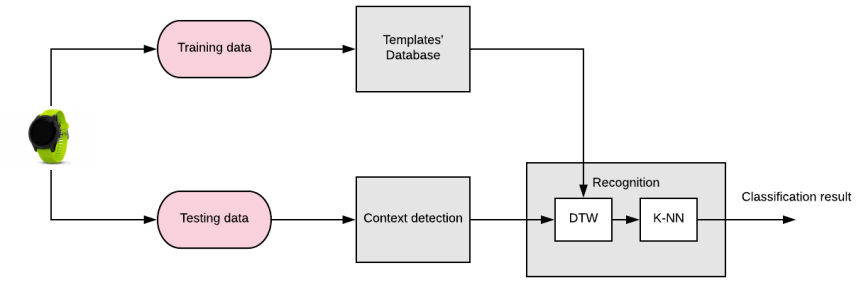

Figure 4.General overview of the recognition system

Using the aforementioned templates, One Nearest Neighbors classifier uses Dynamic Time Warping algorithm to compute the distance between each template and the new input from the accelerometer and assigns to the output the label of the best matching template.

Dynamic Time Warping (DTW) is a classical algorithm used to match two-time series. It is based on dynamic programming and it needs a function to calculate the distance between two points in the two series like it is described in (Liu et al, 2009). The system computes the distance between two points as the absolute value of the difference between the values for those points. DTW result for two series Gi and Gj will be:

$$DTW(G_i,G_j) = DTW(x) + DTW(y) + DTW(z) \quad (1)$$

where DTW(x), DTW(y), DTW(z) are the DTW distances computed for x, y and z axes.

Nearest Neighbors (K-NN) Classifier is one of the simplest classification methods. Given a new input *g*, using the database with labeled templates, K-NN classifier will find the *k* nearest neighbors of *g* from the database and among them selects the most frequent class, as described in (Duca et al, 2017). For our implementation, the distance will be measured using DTW and the value of K will be 1.

5. Accelerometer data analysis

As mentioned in the previous sections, for each gesture we are recording the

values registered by the accelerometer for three seconds, starting when the user indicates his intention to perform a gesture by pressing the hardware button on the watch. The three seconds time-frame has been empirically established based on measurements performed.

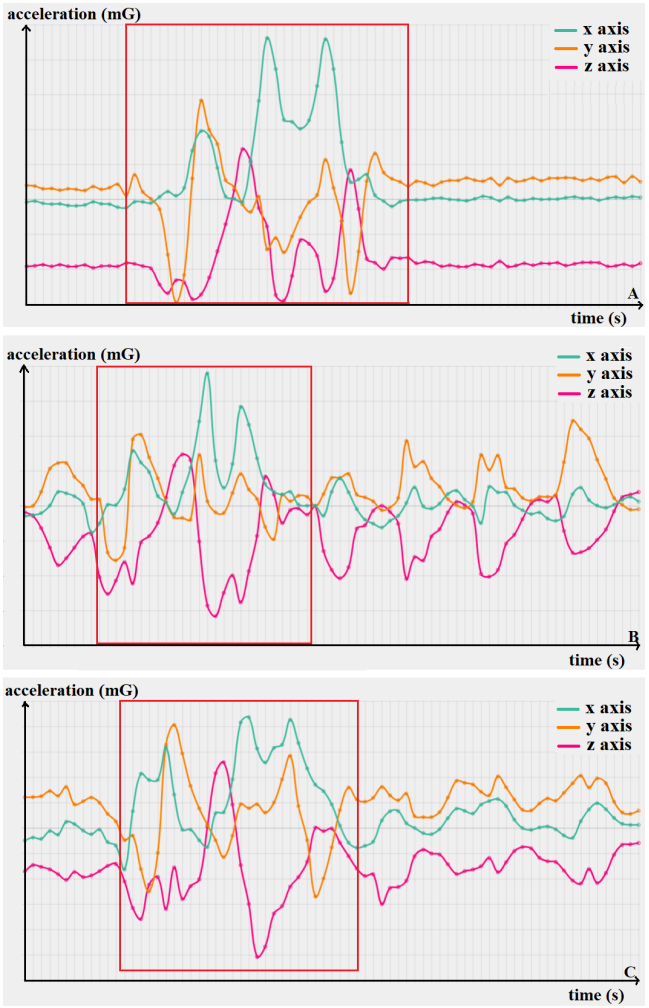

Figure 5: Previous gesture performed while A: sitting, B: running, C: walking

However, there are users who are finishing the gesture description significantly faster. In these cases, we have instructed our test subjects to maintain their hand in a horizontal position (similar to the gesture-start

position) after describing the gesture for the remaining of the time (approximately). This has helped significantly with noise reduction in recorded data which has been translated in better performance in gesture recognition.

An example of the data recorded for the Previous gesture (its shape can be seen in Figure 3) can be seen in Figure 5. We have highlighted with a red rectangle the values recorded during the actual arm movements needed to perform the gesture, the rest of the values being registered while the user was preparing to start the gesture description and after the description was complete (the user still keeping the arm horizontally). The total recorded time is of three seconds, from the moment the button was pressed.

In Figure 5A, the gesture was performed while the user was sitting. It can be observed that before and after the gesture has been performed, when the user is required to keep his forearm parallel to the ground, the values for X and Y axes are close to 0 and the values for Z are close to -1000 mG units (the value of acceleration due to gravity at the Earth's surface).

However, analyzing the data from the recording that was performed while the test subject was running, we can observe that the values registered on all three axes before and after gesture description are not close to a certain value anymore (see Figure 5B). Considerable variations of the acceleration can be seen on the Z-axis because while the user is running he cannot keep his hand at a constant height. At each step, the hand will move up and down generating significant deviation on Z-axis. Also, during running, he will move his hand from left to right and back and forth, so the acceleration on X and Y axes will also be different than 0.

In Figure 5C, we can view the data recorded when the gesture is performed while the user was walking. The same observations as to when the user was running are also valid here, but the values of the acceleration that appear while walking will be less than those for running. Based on these observations, we decided to have in the database only two categories of templates: recorded while the user was sitting and while he was jogging. We did not include a third category for templates recorded while the user was walking because we obtained good results in classifying gestures performed while the user was walking using the existing templates in the two categories.

Due to performance reasons, the recognition algorithm should compare the recorded data with a minimum of predefined templates. If the templates from both categories are taken into consideration at each gesture evaluation,

the performance drops significantly and this is not acceptable for a system that we intend to use in nearly real-time. For this reason, the **Context detection** component has been introduced, that will determine if the user was running or sitting and only the templates from that category will be used by the **Recognition** component.

The algorithm used by the **Context detection** unit to decide if the gesture is performed while the user was sitting or running is based on the number of local maxima that appear on the Z-axis. Before computing this number, a low pass filter is applied to the signal to attenuate the noise that increases the number of local maxima. Our implementation of the *low pass filter* takes a small window centered at each point of the signal, containing 2 neighbors to the left and 2 neighbors to the right, replacing the central point by the mean of all the other points in the window, including the central one. The size of the window was chosen experimentally in a way that prevents an over-smoothing of the final result. The computation of local maxima also takes a window of size 2 and verifies if the point in the center is greater than all the other points in the window.

A supplementary condition to make the local maxima computation more resistant to noise was added: the value of the point must be greater than -500 mG. After the number of local maxima is computed a simple threshold operation is employed to determine if the gesture is made while sitting or running. If the number of local maxima is greater than 5 then the user is running, otherwise is sitting or walking. The value of the threshold was determined experimentally.

6. Usability and performance evaluation

The system is designed to be used by people having different ages, that practice sports on a regular basis or only occasionally. For these reasons 10 people covering all these categories were asked to use the system. In order to evaluate the gesture recognition performance every user repeated each of the 4 gestures 5 times under different contexts: sitting, walking and running, wearing the smartwatch on the left hand and on the right hand. As a result, a total of 120 gestures have been recorded for each user.

Our test users belong to different age groups: 1 person is under 20 years, 6 persons are between 20 and 30 years, 2 persons are between 30 and 40 years and one person is over 50 years. Only two of them wore a smartwatch before.

Taking into consideration the fact that 40 gestures should be performed while running we also considered as segmentation criterion how much they practice sports. From the chosen users: 4 practice sport on a regular basis and 6 only occasionally. None of the users had used a gesture recognition system before.

The gestures and the contexts under which the gestures should be performed were presented before the experiment to each participant. After they found out the content of the experiment they had the ability to choose the order in which they will perform the gestures. A part of them chose to start doing the gestures with the right hand because they thought that it would be more uncomfortable. Some of them assumed that performing the gestures sitting and then walking would be easier, so they decided to start with the easy part of the experiment. Some of the participants that practice sport every day preferred to start by running, considering this part as being the most interesting from the experiment.

We have summarized the most relevant aspects related to the experiments that were performed:

- **The attitude against the experiment**: none of the users had used a gesture recognition system before and all were very curious to see such a system. Some users have not seen a smartwatch before and they were very enthusiastic to wear one for the first time. A part of them required a short training phase, during which they did all the gestures without recording them into the database because they were nervous. The people that do not practice sport on a regular basis were afraid that they would not be able to run and also to perform the gesture at the same time.
- **The order that he/she has chosen for the contexts**: all the users are right-handed and they wear the watch on the left hand. Some of them started to perform the gestures with the right hand because they thought that it would be more uncomfortable and they wanted to start with the most uncomfortable part. Also, some of them considered that it would be harder to perform the gestures while running, so they decided to start by performing the gestures while sitting and walking in order to get used with the gestures before doing them while they are running.
- **The order of gestures in the chosen context**: all the users preserved the order in which we have first shown them the gestures.
- **Memorability of gestures**: at the first rounds of repetitions we have

been asked to showcase again the gestures before they started the round, then they remembered correctly the gestures without any help.

- **The opinion about performing the gestures with left and right hand**: the majority of the users did not find any difference between performing a gesture with the right hand or with the left hand. The rest of them said that it is more uncomfortable to perform the gestures with the right hand.
- **The difficulty of gestures**: four users considered that Previous gesture is a little bit more complicated than the other gestures. The rest considered that the gestures have equal difficulty.
- **The way of recording the samples while running**: some of the users were running continuously at a moderate rate, some of them were running continuously at a high rate and some of them stopped running after a gesture was recorded and sent to the phone for the recognition.
- **Improvement suggestions**: after the experiment, we have asked all the users to give at least one suggestion about how to improve the system. Some of the received suggestions are: add new gestures to control the volume of the song; add the possibility to control the YouTube playlist, not only the music player integrated into the application running on the smartphone; add information about the distance that a user ran/walked and the average running speed.

For the gesture recognition algorithm, we have used a different number of templates in the database for all the gestures performed by the users. The database had 20, 40 or 60 templates, resulting 5, 10 respectively 15 templates for each gesture. The results of classifying the new input gestures using a variable number of templates in the database, together with the execution time for each number have been saved for each user.

The accuracy was computed as the ratio between the total number of gestures correctly classified and the total number of performed gestures. We obtained different values for the execution time of the recognition algorithm when the application was running in the background (another application was in the foreground, or the screen was locked) or the application was in the foreground and the screen was unlocked. Table 1 shows the change in accuracy as the number of templates per gestures in the database is increased, but this will also increase the execution time of the recognition algorithm. It can be observed that the accuracy is increasing as more templates are added

for each gesture, but the execution time is also increasing. We want to use our system in nearly real-time, so is very important to make a trade-off between the accuracy and the recognition time.

Table 1.Change in accuracy and execution time as the number of templates per gesture is increased

Templates per gesture	Accuracy	Foreground execution time	Background execution time
5	81.75	0.85	1.88
10	88.83	1.51	3.72
15	92	2.29	5.9

The **Context detection** algorithm had an accuracy of 67.75%, for 99.25% of the gestures performed while sitting the context was detected correctly and only for 36.25% of the gestures done while running the context was correct. We run the recognition algorithm on all the gestures performed while running using only templates from this category. In Table 2, the second column shows the accuracy of the recognition algorithm that used templates according to the result of **Context detection** unit (only the gestures performed while running were taken into consideration). Also, for the gestures performed while running the third column shows the accuracy of the recognition algorithm when only templates belonging to the running category were used.

It can be observed that better results are obtained if we use the output of the **Context detection** unit. We detected the running activity using the local maxima that appear on z-axis values based on the fact that while running the user will move his arm up and down. This is true in the situation when the user is running fast, but if the user is running slowly the movement of the arm will be more smooth, the local maxima will not appear or will be less than 5 (the threshold that we found experimentally) so the gestures will be closer to the templates recorded while sitting. This explains better results which were obtained using the output of the **Context detection** component.

Table 2.Accuracy of recognition the gestures performed while running

Templates per gesture	Accuracy based on the detected context	Accuracy based on ideal context
5	71.75	66
10	81.25	69
15	86.25	72.75

Looking carefully at the templates recorded during the testing time we observed that the majority of the wrong classified gestures were mistaken with gesture Previous. This happened when the gesture was performed too late after pressing the button and only a short part of the gesture was saved and sent to the phone. For the situations when then user performs the gesture too late after pressing the button, the accuracy of the algorithm is affected. For an user-dependent approach, this issue is less frequent because the person that records all the training templates becomes familiar with the system, he/she knows that the gesture should be performed shortly after the button was pressed.

7. Conclusion

This article describes the testing methodology used for the evaluation of a simple gesture recognition system that employs a single 3-axis accelerometer integrated on a smartwatch and leverages a mobile phone to perform the needed computations. We have built a database of templates for each gesture that were recorded sitting or running with the left and right hand. The core of our recognition approach is the Dynamic Time Warping (DTW) algorithm. In order to reduce response time, only a subset of templates from the database will be used for the processing of every new input, the subset being selected based on the hand used for performing the gesture, indicated by the user, and on the context in which the gesture has been used (while sitting or running) which is established dynamically.

For testing purposes, the system has been set up to recognize 4 gestures and it was tested by 10 users that repeated each gesture 5 times under different contexts: running, walking, sitting, with the right and left hand. We achieved different values of accuracy depending on the number of templates for each gesture: using 5 templates per gesture we achieve 81.75%, for 10 templates it is achieved 88.83% accuracy and for 15 templates the accuracy is 92%.

Further development will involve finding a more flexible way of recording the gesture, the user will not need to keep the hand parallel to the ground for the remaining time until 3 seconds have passed since he has pressed the button for recording. Extending the set of possible gestures in order to increase the number of commands, will be also considered. Moreover, for decreasing the recognition time a user-dependent approach will be investigated.

References

Ahmad, A., Chen, F., Shahrokh, V. A novel accelerometer-based gesture recognition system. *IEEE Transactions on Signal Processing*, 59(12):6197–6205, 2011.

Amit, D., Ivan, T., Shoaib, M. Ultrasound based gesture recognition. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017. p. 406-410.

Angelica Lo Duca, Clara Bacciu, and Andrea Marchetti. A K-nearest neighbor classifier for ship route prediction. In *OCEANS 2017-Aberdeen*. IEEE, 2017. p. 1-6.

David J Sturman, David Zeltzer, and Steve Pieper. Hands-on interaction with virtual environments. In Proceedings of the 2nd annual ACM SIGGRAPH symposium on User interface software and technology, pages 19– 24. ACM, 1989.

Disi Chen, Gongfa Li, Ying Sun, Jianyi Kong, Guozhang Jiang, Heng Tang, Zhaojie Ju, Hui Yu, and Honghai Liu. An interactive image segmentation method in hand gesture recognition. *Sensors*, 17(2):253, 2017.

Du Jiang, Zujia Zheng, Gongfa Li, Ying Sun, Jianyi Kong, Guozhang Jiang, Hegen Xiong, Bo Tao, Shuang Xu, Hui Yu, Honghai Liu, and Zhaojie Ju. Gesture recognition based on binocular vision. *Cluster Computing*, 2018, 1-11.

Hongyi Wen, Julian Ramos Rojas, and Anind K Dey. Serendipity: Finger gesture recognition using an off-the-shelf smartwatch. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 3847–3851. ACM, 2016.

In-Cheol Kim and Sung-Il Chien. Analysis of 3d hand trajectory gestures using stroke-based composite hidden Markov models. *Applied Intelligence*, 15(2):131–143, 2001.

Ivan Dokmanic and Ivan Tashev. Hardware and algorithms for ultrasonic depth imaging. *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6702–6706. Ieee, 2014.

Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. Uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

Liqian Feng, Sheng Bi, Min Dong, and Yunda Liu. A gesture recognition method based on binocular vision system. In *International Conference on Computer Vision Systems*, pages 257–267. Springer, 2017.

Lorenzo Porzi, Stefano Messelodi, Carla Mara Modena, and Elisa Ricci. A smart watch-based gesture recognition system for assisting people with visual impairments. In *Proceedings of the 3rd ACM international workshop on Interactive multimedia on mobile & portable devices*, pages 19–24. ACM, 2013.

Manas Kamal Bhuyan, Karl F MacDorman, Mithun Kumar Kar, Debanga Raj Neog, Brian C Lovell, and Prathik Gadde. Hand pose recognition from monocular images by geometrical and texture analysis. *Journal of Visual Languages & Computing*, 28:39–55, 2015.

Mridul Khan, Sheikh Iqbal Ahamed, Miftahur Rahman, and Ji-Jiang Yang. Gesthaar: An accelerometer-based gesture recognition method and its application in nui driven

pervasive healthcare. *2012 IEEE International Conference on Emerging Signal Processing Applications (ESPA)*, pages 163–166. IEEE, 2012.

Noor Tubaiz, Tamer Shanableh, and Khaled Assaleh. Glove-based continuous Arabic sign language recognition in user-dependent mode. *IEEE Transactions on Human-Machine Systems*, 45(4):526–533, 2015.