Smart Watch-based Gesture Recognition to Control a Music Player

Ioana-Crinela Potinteu
Technical University of Cluj-Napoca, Computer Science Department
Cluj-Napoca, Romania
crinela.potinteu@student.utcluj.ro

Teodor Ștefănuț
Technical University of Cluj-Napoca, Computer Science Department
Cluj-Napoca, Romania
teodor.stefanut@cs.utcluj.ro

ABSTRACT
This paper describes a system for gesture recognition based on a smart watch and a mobile phone. Our system also integrates a Web API for saving, retrieving and processing data about the gestures and a web application to visualize the saved data. We propose a button-based segmentation method that uses Dynamic Time Warping and One Nearest Neighbors to classify the data from a single 3-axis accelerometer. The system distinguishes 4 arm gestures using a variable number of labeled templates for the classifier. To emphasize the practical utility of the system the recognized gestures are used to control a music player. This is just an example. This form of user interaction can be exploited by many different types of applications, for instance applications for home automation.

Author Keywords
Human-computer interaction; gesture recognition; smart watch; accelerometer; classification; dynamic time warping

ACM Classification Keywords
(H.5.2): User Interfaces

INTRODUCTION

Gesture recognition is considered as the process of understanding and classifying purposeful people movements [1]. Face, head, arms or hands movements are analyzed and identified depending on the purpose of the system that uses the recognition. In the last few years, the capabilities of smart watches have evolved surprisingly. Now using a smart watch people have the possibility to monitor health, play music, browse the Internet and so on. All these activities still need to be made by touching the relatively small screen or pressing the physical buttons. The interaction can be significantly improved using the accelerometer that is already integrated in almost all the wearables devices. The values obtained from the accelerometer are a very good input for an arm gesture recognition system.

In this work we present a 3-axis accelerometer-based gesture recognition system that is composed of a smart watch and a mobile phone. The values obtained from the smart-watch's accelerometer are transmitted to the mobile phone where the recognition algorithm is implemented. A dictionary of 4 gestures was established and for each one it was associated an action in order to control a music player that is running on the smart phone. The actions are the following: start or stop the music on the phone, play the next or the previous song.

The smart watch and the mobile phone must be permanently connected through Bluetooth, the application running on the smart watch must be permanently in foreground and the application running on the smart phone can be either in foreground or in background. The system recognizes the gesture after the user follows the next steps: bring the forearm parallel to the ground, press a certain button and 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 before. After pressing the button the user has 3 seconds to perform the gesture. If the user does the gesture earlier than 3 seconds he or she needs to keep the arm in the finish position for the rest of the time.

A database of 240 labeled templates was built, they were collected from a single person. All the templates have the same length, because the user has 3 seconds to perform any gesture. There were recorded 15 labeled templates for each gesture with each arm, while sitting and running. One Nearest Neighbor Classifier will find the best matching template in the database for the new values that come from the accelerometer. The system achieves different values of accuracy depending on the number of labeled templates that are used by the classifier. Using 5 templates for each gesture is obtained an accuracy of 81.75%, for 10 templates is obtained a value of 88.83% and for 15 templates the accuracy is 92%.

In the next section it is discussed the related work. Then it is provided a general overview of the proposed system. After that it is presented the gesture recognition approach that we have used. It is followed by a section that describes the methodology employed to evaluate the system and finally the conclusions are introduced and future improvements are proposed.

RELATED WORK

The first algorithms for gesture recognition were designed for data gloves. In 1989, Surman used data gloves to identify gestures in order to manipulate virtual objects [4]. Since then many approaches to gesture recognition problem that used data gloves appeared, for instance [8, 15].

Computer vision methods are also employed for gesture recognition. In [2, 9] is used ultrasonic depth imaging. Monocular cameras were also used [13] and for more complex environments binocular cameras achieved a better accuracy [6, 5, 11].

In literature, there are several systems that use an accelerometer and a gyroscope for gesture recognition. One of the most popular systems that uses only a 3-axis accelerometer is uWave [10]. It is a user-dependent system that requires the user to perform all the gestures before using it. It was defined 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 in 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 [1] 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 "exemplar". After this phase is obtained the database containing all the exemplars. 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 [7]. It combines the data from the 3 axes (x, y and z axis) of an accelerometer, a gyroscope and a linear accelerometer integrated in a Samsung Galaxy Gear smart watch. 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 [14]. 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 uWave gesture library was used to build a prototype of the system.

An interesting approach based on global alignment kernels is presented in [12]. It is proposed a user-independent approach capable to recognize 8 distinct gestures, using a 3-axis accelerometer integrated into a smart watch. 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 smart phone that the user wears at his or her neck. The target of this system are people with visual impairments.

In this paper, starting from the approach proposed by uWave we build a user-independent system the 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.

SYSTEM OVERVIEW

Figure 1 depicts the communication architecture of the system showing the flow of the data through the system.

For experiments we have used a Garmin Forerunner 935 smart watch 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 Bluetooth Low Energy. To program the smart watch, 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,

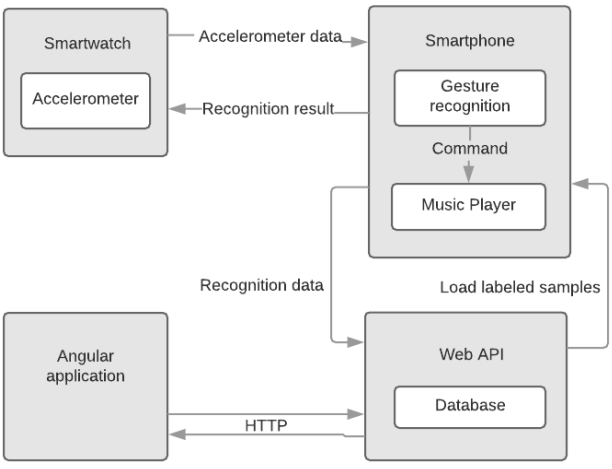

Figure 1: Communication architecture of the system

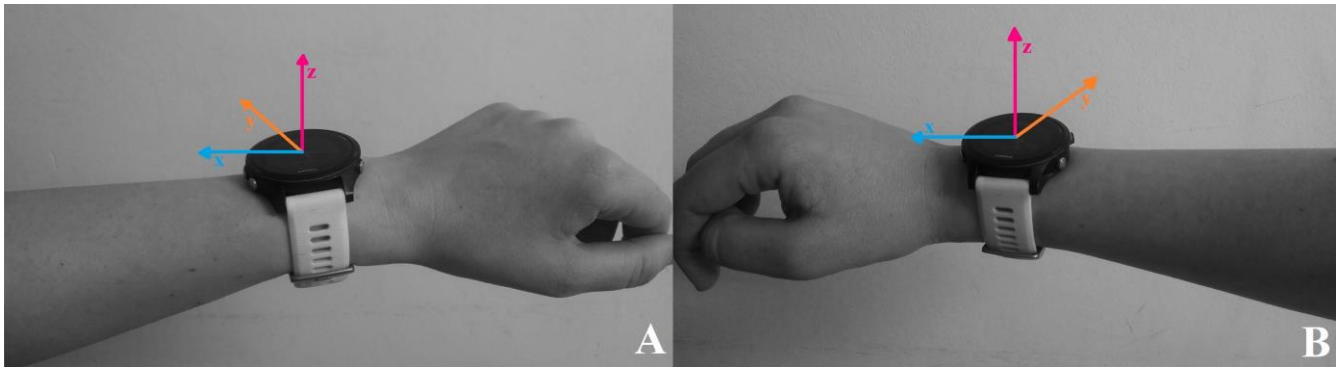

Figure 2: A: smart watch worn on the left hand and the accelerometer axes direction B: smart watch worn on the right hand and the accelerometer axes direction

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.

The Connect IQ SDK imposes many constraints on the applications that can be developed for the smart watches. 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.

In Figure 2 it can be seen the accelerometer axis directions when the smart watch is worn on the left and on the right hand. 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. 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.

The template's database that is used in the recognition process is different for each hand. The user needs to select the hand on which he or she is wearing the smart watch in the application that is running on the smart phone. The templates' database used by the recognition algorithm depends on the selected hand.

To use the system the user presses the button indicating that he or she wants to perform a gesture, the smartwatch will save the accelerometer data for the next 3 seconds and will send it to the smart phone. The used smart phone is BlackBerry Priv that runs Android OS. Both templates' databases for right and left hand will be loaded only once, when the application on the smart phone 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.

For analyzing the system's performance and results we have added the possibility that before doing a gesture, the user can manually choose the gesture that he or she will perform, the context under which the gesture will be made (sitting, walking or running) and these values, together with the recognition result and the execution time for using 5, 10 or 15 templates per gesture, will be saved in the database through the same Web API that was used to load all the templates. For the added functionality a register and log in mechanism were implemented in the application running on the smart phone. Each recognition data that is sent to the server also contains the unique identifier of the user.

GESTURE RECOGNITION APPROACH

Figure 4 depicts the general overview of our proposed approach for gesture detection. Firstly, the templates' database was built. The dictionary with the proposed gestures for controlling the music player can be seen in Figure 3. Having the database One Nearest Neighbors

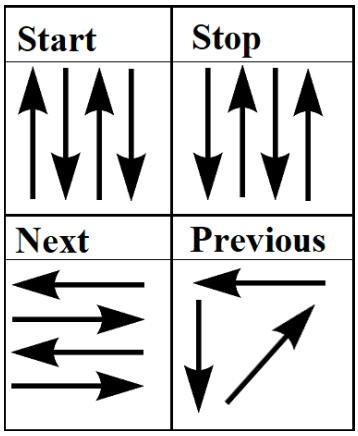

Figure 3: The dictionary of 4 gestures

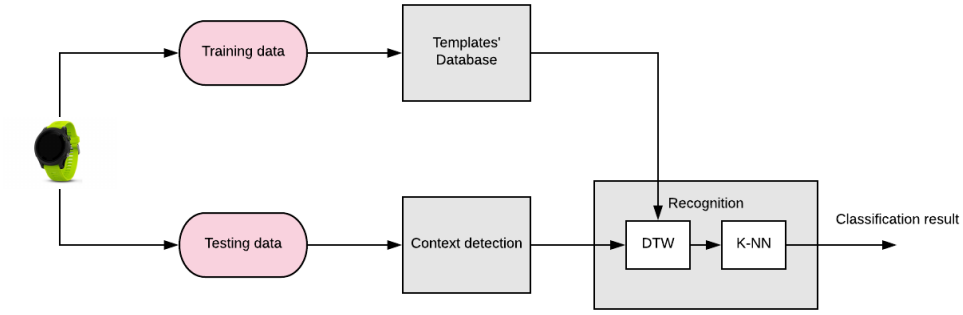

Figure 4: General overview of the recognition system

classifier uses Dynamic Time Warping algorithm to compute the distance between the templates and the new input from the accelerometer and assigns to the input 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 [10]. The system computes the distance between two points as the absolute value of the difference between the values for that points. DTW result for two series G_i and G_j 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.

K 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 [3]. For our implementation, the distance will be measured using DTW and the value of K will be 1.

As it can be seen in Figure 4, one of the system's components is the **Context detection** unit. This component implements a very simple algorithm to determine whether the gesture was performed while sitting or running. The unit was necessary because after plotting the same gesture while running, walking and sitting important differences were observed for the values recorded for z axis.

In the Figure 5, it was plotted the accelerometer data obtained by performing the Previous gesture, its shape can be seen in Figure 3. With a red rectangle we have highlighted the values recorded during the arm movements needed to perform the gesture, for the rest of the time until 3 seconds have passed from the moment when the button was pressed, the user is required to keep the forearm parallel to the ground.

In the Figure 5A, the gesture was performed while the user was sitting. It can be observed that during the time 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).

In the Figure 5B, when the gesture was performed while running for the same period where the user kept his forearm parallel to the ground the values on all three axes are not close to a certain value anymore.

Considerable variations of the acceleration can be seen on 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 and acceleration will appear 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 is no longer close to 0.

In the Figure 5C, the gesture is performed while the user was walking. The same observations as 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 two categories of templates: recorded while the user was sitting and while he was running. 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. The recognition would be very slow if the templates from both categories are taken into consideration 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 **Context detection** unit to decide if the gesture is performed while the user was sitting or running computes 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

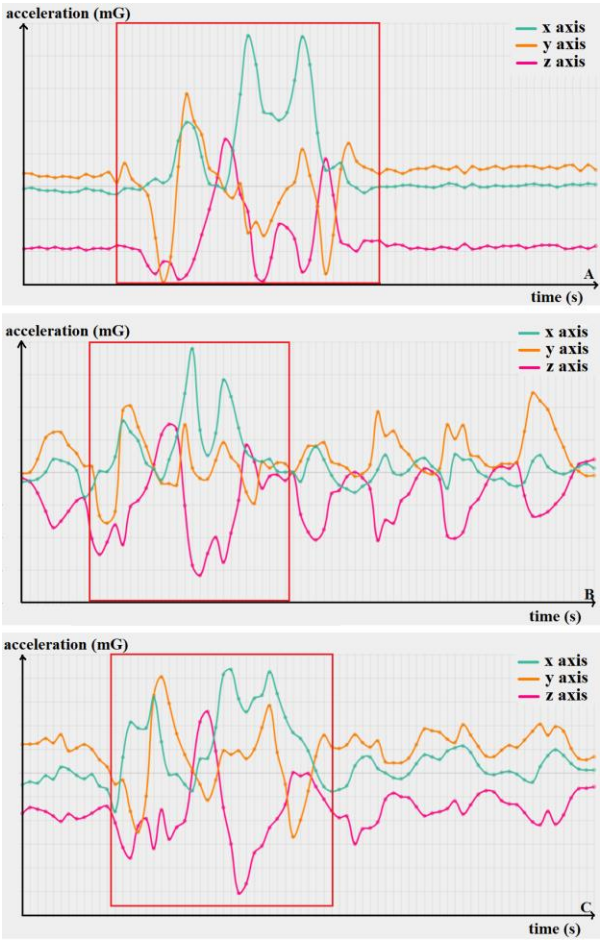

Figure 5 Previous gesture performed while A: sitting, B: running, C: walking

increases the number of local maxima. 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. The filter will replace 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 in a way that the resulted curves would not be too smooth. 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.

For efficiency reasons, the low pass filter and the maxima computation will be done together, the algorithm will compute the mean of point i and will test if point i-2 is a local maxima. 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.

When the user performs a new gesture the **Context detection** unit will determine if the gesture was performed running or sitting. Based on this result and taking into consideration the hand on which the smart watch is worn it is determined the set of labeled templates to be used by 1-NN to classify the input gesture.

EVALUATION

To evaluate the performance of the recognition method, we acquired data from 10 users. From each user it was gathered the following gesture set: 4 gestures (Start, Stop, Next and Previous) x 5 repetitions x 2 arms (left / right) x 3 contexts (sitting / running /walking) = 120 gestures.

The 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 smart watch 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 regularly basis and 6 only occasionally. None of the users had used a gesture recognition system before.

For comparison, we 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 (other application was in foreground, or the screen was locked) or the application was in 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

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

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

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

Table 2: Accuracy of recognition the gestures performed while running

to use our system in nearly real time, so is very important to make a trade-off between the accuracy and the recognition time.

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 did 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 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.

CONCLUSION

We have proposed a simple gesture recognition system that employs a single 3-axis accelerometer integrated on a smart watch 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 Dynamic Time Warping (DTW) algorithm. Not all the templates from the database will be used for every new input. The algorithm will use only the templates for the hand indicated by the user and it will choose between the templates recorded while running or sitting depending on an algorithm that we have implemented.

The system can 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.

ACKNOWLEDGMENTS

We would like to thank Garmin Romania for offering technical support and test devices.

REFERENCES

1. Ahmad Akl, Chen Feng, and Shahrokh Valaee. A novel accelerometer-based gesture recognition system. IEEE Transactions on Signal Processing, 59(12):6197–6205, 2011.
2. Amit Das, Ivan Tashev, and Shoaib Mohammed. Ultrasound based gesture recognition. 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2017. p. 406-410.
3. 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.
4. 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.
5. 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.
6. 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.
7. 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.

8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. 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.
15. 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.