2017 IEEE 41st Annual Computer Software and Applications Conference

Wrist Player: a smartwatch gesture controller for smart TVs

Mateus M. Luna, Thyago P. Carvalho, Fabrizzio Alphonsus A. M. N. Soares,
Hugo A. D. Nascimento and Ronaldo M. Costa
Universidade Federal de Goiás - Instituto de Informática - Goiânia, Goiás, Brazil, 74690–900
Email: mateus.m.luna@gmail.com,{thyagoperes, fabrizzio, ronaldocosta, hadn}@inf.ufg.br

***Abstract*—Emerging technology on mobile and wearable market, smartwatches have embedded movement sensors whose potential is yet to be fully explored. This paper proposes an interaction method with smart TVs via gestures performed by person's wrist using a smartwatch. Detailed architecture and implementation for a complete prototype, named Wrist Player, is presented. A user study is also conducted, in order to evaluate the prototype performance and the user's interest on the proposal. Results show that the method works very well, with participants reporting having a good experience with the prototype. We present our insights on the concept, challenges faced in our research and ideas for future studies.**

***Keywords*— Gesture Recognition; Human Computer Interaction; Movement Sensors; smart TV; smartwatch; Internet of Things.**

I. Introduction

Since its creation, in the 1930s, television took a part in people's daily time and got many new features to become what we name today as smart TV. Currently, they can be seen as media centers connected to the Internet and to other smart devices. In the past, one had to interact with TV directly by dials, sliders and push buttons to perform simple operations such as changing channels or setting up volume. Some years after, remote control was created in order to perform the same operations, but keeping the viewers comfortably on their couches.

Nowadays, smart TVs have embedded new features such as voice control and camera gesture recognition in order to provide more interactions, but both features show strengths and weaknesses. Voice Recognition can free the users from holding a physical control device. However, using it in a noisy environment, simultaneously with with other communication "channels", can be very challenging and frustrating. Gesture recognition by depth cameras can do a great job mapping gestures performed by the viewer's body, but still faces limitations regarding distance, positioning and specially ambient lighting.

Recently, devices such as smartwatches have become popular, providing to users touch screens, Wifi and Bluetooth® communication and being able to understand voice and gesture commands. Just like what happened with the smartphone evolution, smartwatches are literal "wrist computers", with a processing potential comparable to many of our desktops. For being so well attached to our body, they allow gesture detection with a good precision for several operations, such as waking up the screen with a simple twist or visually moving items on a list by using a common drag-and-drop interaction.

In this paper, we present Wrist Player, our envision on integrating smart consumer devices, making use of communication, computing and potential interaction between those devices. Particularly, this work focus on how to interact with smart TVs via wrist gestures, taking advantage of accelerometers, available in almost every smartwatch. The interaction concept can be summarized in Fig.1.

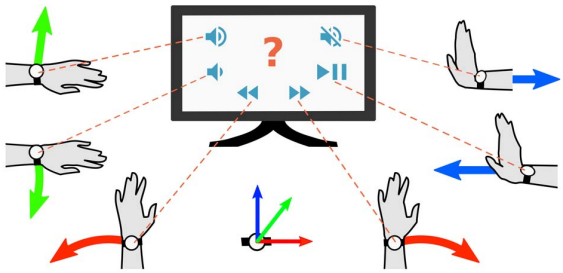


Fig. 1: Tracking wrist gestures with a smartwatch in order to classify them as regular gestures (push, pull, and move up, down, left or right), and to map them to actions on a smart TV, such as toggling play/pause, changing the volume and performing playback control.

The paper is organized as follows: in next section, the related work is presented. Then, in Section III, we introduce the Wrist Player interaction method and its prototype in details. A user study of the system is described in Section IV. Finally, we draw our conclusion and suggest ideas for future work in Section VI.

II. Related Work

Back in 2001, Rekimoto [1] proposed a concept of wearable technology called GestureWrist. His prototype consisted of a 2-axis accelerometer on the top of a watch, and capacitor sensors on a wristband. The author stands out the need of having wearable devices less intrusives and more natural to the user. Few years later, this aspect started receiving more attention, and have now been subject of several studies.

A. Accelerometer Control

Kela *et al.* [2] studied gestures recognition using Hidden Markov Model with a small portable device to interact to smart environments. Similarly, Liu *et al.* [3] and Kühnel *et al.* [4] conducted studies with smartphones and Nintendo Wii remote using Dynamic Time Warping (DTW) for gesture recognition. In their conclusions, it is highlighted that although an initial gesture set can be useful, users prefer defining their own gestures.

B. Smartwatches

A prototype for controlling a music player application via touches and swipes in a wristband, as well as a glove, was built by Speir *et al.* [5]. Results show that users have interest on this kind of interaction, specially swipe gestures on wristband, instead of touch ones.

McGuckin *et al.* [6] analysed the interest of 16 subjects on using gesture control in a message application, both with smartwaches and smartphones. The users approved mostly the basic gestures, specially on smartwatches, and proposed

0730-3157/17 $31.00 © 2017 IEEE
DOI 10.1109/COMPSAC.2017.266

a combination of tapping and gesture control. Many of the interactions studied there are becoming standard in Android Wear devices, as well as the system itself has gestures recognitions in its most recent versions.

Mace *et al.* [7] focused on the adoption of smartwatches for gesture classification and compared implementations of the Naïve Bayes Classifier and DTW. Mace also suggested some improvements such as Plane Adjustment and the reduction of noise caused by gravity acceleration.

Kerber *et al.* [8] used DTW to compare a set of gestures with routine movements on smartwatches for obtaining an appropriate "gesture delimiter", that would not conflict with daily actions performed by user's wrist. Their research enforces that having this delimiter would avoid the need of two hands for interactions (Opposite-Side Interaction - OSI). After significant data collection and analysis, it was defined that a wrist rotate or double rotate is suitable enough for using as gesture delimiter. Previously, Ruiz and Li [9] had already proposed a double flip gesture as delimiter following similar experiments, but for smartphones.

Another work where trigger gestures are used appears in Porzi *et al.* [10], to interact with a smartphone camera activating identification of objects for visually impaired people. In their work it was used a Global Alignment Kernel (GAK) and a Support Vector Machine (SVM) classifier to detect the proposed set of gestures. Later, in [11], that work was extended with a transfer learning approach, allowing users to provide personalized gestures.

Xu *et al.* [12] used accelerometer and gyroscope from a special device attached to a wrist band, to prove that a classification of finger, hands and arm gestures is feasible on smartwatches. They analysed the magnitude of energy to classify move source, and then applied machine learning for classification. A training set was obtained from one person only, although good results were presented, in a set of 37 test gestures, including experiments on finger writing.

C. Applications

Sen [13] detected eating events based on smartwatch sensors to activate a camera in the device for a diet control system. Few tests were realized and a series of challenges to this approach were presented, but the author achieves good results on classifying different types of eating modes and frequency.

Ferrari *et al.* [14] proposed a soft authentication system using Bluetooth®, based on the detection of handshakes between pair parts using smartwatches. Machine Learning Techniques for classification were compared, but few tests were presented in terms of the overall proposal. In an different perspective, Sarkisyan *et al.* [15] analysed the feasibility of smartwatch sensor data collection to predict PIN numbers typed on a smartphone. Even having several optimizations to estimate the values, the prediction accuracy varied from 41% to 91% on different data sets.

Waris and Reynolds [16] used Finite State Machines (FSM) optimized by cultural algorithm techniques to propose a gesture interaction system with automotive smartdevices, such as Android Auto, also using an Android Wear. Although it was proved that cultural algorithm improves FSM structure over training data and that this algorithm was less resource consuming than strategies such as DTW, there was no much validation of the proposed gestures on the context of driving. We highlight that, in Waris and Reynolds' work, a twist gesture is also proposed as a trigger to start data collection.

D. Smart TV

Vatavu *et al.* [17] offered insights concerning the use of a Leap Motion to execute tasks on a smart TV. One important observation made is the user preference for gesture commands instead of the TV remote control (82% mean preference, in a set of 21 actions, proposed and validated by 18 volunteers). Although this study presents many user's insights on gestures, most of them use hand and finger gestures, not all feasible on smartwatches.

Keshav *et al.* [18] presented a system planned for integration between smartwatches and smart TVs through gestures. In their paper, it is pointed out the need of a sensor data processing in a *smartphone* that then sends commands to a smart TV, where a browser application interprets a set of gestures. Although, this is a very interesting work, neither it has a user study nor presents how gestures are delimited over time. In addition, their application for smart TVs employs gestures for actions not related to the standard media playback, and employs a specific branded protocol, which make its prototype limited for the TV hardware being used.

III. Wrist Player

Wrist Player is our prototype to provide user interaction with smart TVs through wrist gestures tracked by smartwatches. Thus, six gestures are proposed, as presented in Fig. 1, and their mapped actions are clarified in TABLE I.

TABLE I: Wrist Gestures and Corresponding Actions

Gesture	Action
To move wrist up	Increase volume
To move wrist down	Decrease volume
To move wrist left	Rewind playback
To Move wrist right	Forward playback
To push wrist	Toggle Play/Pause
To pull wrist	Toggle Mute/Unmute

This set of gestures was defined via a brainstorm, with ten volunteers, basing on works of Kela [2] and Kuhnel [4].

In The following sections, we present the used materials, the architecture of the prototype and the details of the proposed interaction method.

A. Materials

For our prototype development, Samsung Gear Live and Moto 360 1^{st} generation smartwatches were used. The Moto 360 ran Android Wear version 1.4.0 and had embedded an InvenSense MPU-6051 chip of Six Axis, endowed of accelerometer and gyroscope. The Gear Live ran Android Wear 1.5.0 and had an InvenSense MP92M chip of 9 axis, with accelerometer, gyroscope and magnetometer.

A LG NEXUS 5 smartphone, running Android 6.0.2 was used and connected to the smartwatches via Bluetooth®. Android Studio 2.0.0 and the API 23 were employed for the development.

A smart TV module was simulated using a regular TV connected to a Chromecast 1^{st} generation via HDMI and USB ports, for video/audio and power source, respectively.

The Chromecast-smartphone communication happened via wifi network.

We highlight that, up to Android Wear version 1.5.0, the smartwatch is limited to communicate only with a paired smartphone, even when using Wifi connection. Google promised to provide full communication stack via Wifi in the Android Wear 2.0. However, when this paper was written, the new API was available only for few devices. We also split the prototype between the smartwatch and the smartphone in order to distribute processing load and to avoid a fast depletion of the smartwatch battery.

B. System Architecture

The prototype system for Wrist Player is composed by a Styled Media Receiver, a Media Controller App and a Wearable App, as seen in Fig. 2.

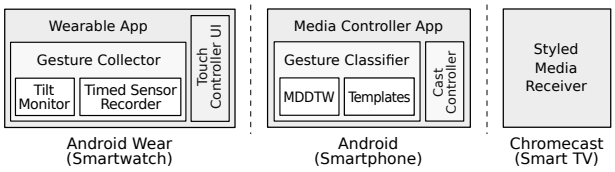


Fig. 2: Wrist Player: System Architecture.

The Styled Media Receiver is an application developed by Google using HTML, Javascript and CSS, that is installed on Chromecast on demand, when a streaming is requested by a connected smartphone.

The Media Controller App is a Cast Controller, with a Gesture Classifier module, integrated to the smartwatch. It is a fork from Google Sample Media Controller, which provides basic resources for Chromecast controlling. Once the user establishes a Chromecast session, the Media Controller App starts the Wearable App, presenting its home screen (Fig. 3a) on the smartwatch and giving gesture control ability to the user.

The Wearable App has a Gesture Collector Module and a Touch Controller UI. The Gesture Collector Module can operate in Gesture Controller mode or in Template Recorder mode. In Gesture Controller (Fig. 3a), the module sends gestures to the Gesture Classifier module to be translated as action commands in the smart TV. In Template Recorder (Fig. 3b), it sends gestures to the same module, but for being stored as templates. The Touch Controller UI offers buttons related to the available actions, as an alternative interface, in case the user does not want to perform gestures (Fig. 3c). The three screen options are available and accessed via swiping.

(a) (b) (c)

Fig. 3: Screen options: (a) Gesture Controller mode[1], (b) Template Recorder mode, (c) Touch Controller UI.

The app modules are detailed in the following subsections.

C. Gesture Collector Module

The Gesture Collector module is the kernel of the Wearable App. It is responsible for collecting gestures to be used as control actions or recorded as templates.

In this approach, every wrist gesture is expressed as a time series of accelerometer data in the three axis X, Y, and Z. However, collecting such data continuously leads to many problems. First, people are constantly performing gestures with their hands, for instance when talking, walking, and so forth. Thus, processing a stream of data continuously can imply in undesired gesture recognition. Second, synchronizing the beginning of a collected time series with an actual gesture can be challenging and result in false positive and false negative gesture recognitions. Finally, it implies in unnecessary load processing, with excessive energy draining, and fast battery depletion.

Considering such limitations, we adopt a strategy of executing the collecting routine only during an empirically determined time window of 1.2 seconds, after a "trigger" is fired. Such trigger could be the click of a button, but that would break the logic of having a full gesture-only experience.

In this work, a wrist tilt gesture is adopted as a trigger, based on [8] and [9], and it mimics a similar gesture available in Android Wear since version 1.4. We highlight that the tilt gesture in Android only supports the scrolling of lists and notification cards. It is expected, for future versions of Android Wear, that the developers can program their own actions. For now, we implemented our own tilt monitor, in order to run the Wrist Player prototype.

1) Tilt monitor: A tilt gesture consists of flicking the wrist outward, performing a rotation greater than 90 degrees, and then turning it back to its initial position. The lower arm must be parallel to the ground, and preferably pointing towards the user field of view.

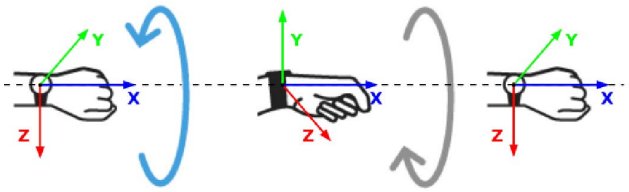


Fig. 4: Tilt gesture sample.

The procedure is demonstrated in Fig. 4. The X axis is aligned to the forearm and the hand (0°), the Y axis is aligned to the watchband (90°) and the Z axis is perpendicular to the watch screen surface (0°). X axis is assumed as revolution axis.

The Watch orientation is obtained based on the accelerometer data (gravity acceleration). Its X, Y and Z angles are calculated using Equation 1 [19]:

$$\theta_a = \arccos\left(G_a \times \left(\sqrt{G_x^2 + G_y^2 + G_z^2}\right)^{-1}\right) \qquad (1)$$

where θ is the watch angle on axis $a = \{x, y, z\}$ and G is the gravity acceleration on its respective axis.

[1]Hands illustration adapted from Android Wear documentation.

In order to detect a tilt gesture, two states are assumed: first, when the watch face is up ($X = 0°$, $Y = 90°$, and $Z = 0°$); second, when the watch face is turned around the X axis, *counterclockwise*, for left hand or *clockwise* for right hand ($X = 0°$, $Y = 180°$, and $Z = 90°$). All axis are analysed together to enforce the user to put the arm in a predetermined position, ignoring Y or Z rotations, so avoiding false tilt detection.

A tilt is detected when a transition occurs from the first to the second state, followed by another transition from the second to the first state. To avoid stability problems, a tolerance of ten degrees is defined as a limit for state transitions.

2) Timed Sensor Recorder: Once a tilt is detected, the smartwatch vibrates and shows a screen image (Fig. 5a) as a feedback to the user. Then it starts a Timed Sensor Recorder. This procedure gathers accelerometer data for a predetermined period, which value was empirically defined in order to record a gesture. During that time window, the trigger detection is suspended. As recommended by Google in the Android Wear documentation, for user interaction, the sensor rate used was the SENSOR_DELAY_UI, which corresponds to ≈ 15 reads/sec.

(a) (b) (c)

Fig. 5: Screens samples: (a) Gesture collecting ongoing, (b) Idle detected feedback and (c) Recognized gesture feedback.

After the timer is lapsed, for avoiding useless processing, the collected data is analysed to identify whether it corresponds to a gesture or to an idle state. An idle state happens when the standard deviation of the time-window data is bellow a certain threshold. For this aim, the accelerometer data is evaluated for each axis (X, Y or Z) individually. In our experiments with different users, better results were empirically achieved with a threshold value of 1.5 for Samsung Gear Live and 2.5 for Moto 360.

Finally, if the collected data corresponds to a gesture, then it is sent to the smartphone to be processed by the Gesture Classifier module. Otherwise, a feedback image (Fig. 5b) is presented on the watch screen for 3 seconds, notifying an idle state to user, and no data is sent to the smartphone.

D. Gesture Classifier Module

The Gesture Classifier module is a smartphone service that receives the collected data from the smartwatch, classifies it with a Multi-Dimensional Dynamic Time Warping (MD-DTW) algorithm as one of the recorded template gestures and performs an action. Thus, when a gesture is recognized, a control operation is sent to the smart TV (or the Chromecast), and a feedback screen image (as in Fig. 5c) is presented for 3 seconds on the smartwatch. Completing this routine, the module returns to awaiting a new gesture state from the Gesture Collector.

The MD-DTW is a variation of the DTW, proposed by Holt *et al.* [20]. It aligns two temporal multidimensional sequences with different sizes. In our proposal, we compare the sensor data sequence with templates recorded before the test sessions.

Firstly, the algorithm calculates a distance matrix (Eq. 2) between each of the multidimensional temporal series.

$$d_{i,j} = \sqrt{\sum_{k=x}^{|K|}(U_{i,k} - T_{j,k}))^2} \tag{2}$$

where U and T are time series of peformed user gesture and the template gesture, respectively. K is the set of axis (X, Y, and Z), and i and j are instant positions in the time series, with length N and M, respectively.

Secondly, the algorithm builds an accumulated distance matrix as shown in Equation 3.

$$D_{i,j} = d_{i,j} + \min\{D_{i-1,j}, D_{i,j-1}, D_{i-1,j-1}\} \tag{3}$$

where $D_{i,j}$ is the accumulated distance value in line i and column j, given by the distance value $d_{i,j}$, explained on Eq. 2, added to the smallest value of the three neighbor positions, preceding the referred position.

Finally, we get the accumulated distance stored in the last matrix element, $D_{N,M}$. These steps are performed for all stored gestures, and the least accumulated distance is used for classifying the gesture. We highlight that it is not necessary to hold the path for the alignment, since we only use the minimum accumulated distance value.

IV. User Study

In order to evaluate our prototype in a scenario of TV controlled by smartwatches via gestures, experiments were performed. Experiment variables, such as gesture accelerometer data, time stampings, classifier setups, resulting control actions and so forth, were collected and stored on a database for analysis.

A. Apparatus

For the user study, a Samsung Gear Live smartwatch, running Android Wear 1.5.0, was used together with a LG NEXUS 5 smartphone, running Android 6.0.2. A Chromecast 1st generation was connected via HDMI and USB ports to a Samsung 40" smart TV. The Gear Live was connected to the NEXUS 5 via Bluetooth, and the NEXUS was connected to the Chromecast via a dedicated Wifi hotspot with an Internet link. The smart TV was set up on a wall at 1.2 meters of height from the ground and at 1.5 meters away from a couch.

B. Participants

The experiments were conducted with 15 participants, with age between 21 and 48 years old and average of 22 years. They were undergraduate students, mostly from Information Systems and Computer Science courses. One was from Psychology and another from Graphic Design. All of them declared to have none or very basic knowledge on smartwatches and little experience with gesture recognition. Only 3 participants opted to wear the smartwatch on the right arm.

C. Template Recording, Testing Scenario and Evaluation Metrics

Initially, the participants were presented, one at a time, to a scenario with a video displaying on a TV controlled by a smartwatch. The behavior and the logic of the prototype were explained, followed by a demonstration in which proper posture, trigger action and gesture options were clarified. Hence, whenever the participants were comfortable with all concepts presented, they were invited to a template recording session.

1) Template Recording : The template recording session involved two parts.

Firstly, the participants were asked to perform the trigger gesture for as long as they need to get used to it. Also, they could perform any gesture freely, trying to mimic those demonstrated previously.

Then, the participants were asked to record their own gestures for each option of the gesture set. Although they were suggested to use a visual description of the gestures as a reference, they had the freedom to record the actions in the way they felt more comfortable, as long as they were aware of the timeout for gesture performing. The template recording was performed twice, changing the posture (one with the participant sitting down, and the other with he or she standing up).

During this stage of the experiment, the watch screen was kept in a template recording mode (Fig. 3b), consequently not having any effects on the video. As described in Section III-C2, the recorded gestures were stored in the smartphone for being used as a parameter by the Gesture Classifier module. After all templates were recorded, the testing session started.

2) Testing Scenario: In a test routine, a participant could interact to the video playback with the sequence of gestures that he or she prefers. The only constraint was to have each gesture performed at least once, completing a test set. When a gesture was correctly classified, the corresponding command request was sent to the Chromecast, causing a response action on the TV.

Testing routines were planned to be performed four times for each posture (sitting down or standing up). The participants were instructed to act the most natural as possible, and to keep their focus on the device that was being controlled, instead of on the smartwatch. The gesture classification rate was considered a test independent variable, while posture and gesture types were assumed as dependent variables.

At the end of each test session, a questionnaire was applied to verify the participant opinions, thoughts and suggestions on the proposed interaction model and about the prototype.

3) Evaluation Metrics: For better visualizing the obtained data, confusion matrices were built. A confusion matrix shows the relation between an actual class, which, in our case, is the expected gestures performed, and a predicted class, here the answer returned by the Gesture Classifier module. It is useful to count the hit rate of each gesture (true positives), and also to identify for which gestures the classifier got mistaken (false positives).

V. Results

In summary, the testing sessions were performed by 15 participants, executing each gesture 4 times sitting down and 4 times standing up, 60 samples per gesture each, providing a total of 720 gestures. However, before each testing session, one set for each posture was recorded as user template for the classifier validation. All testing gestures were stored, properly labeled, organized in confusion matrices as presented in TABLE II, and analyzed.

TABLE II: Confusion Matrix

(a) Sitting down

Expected Gesture \ Predicted Gesture	UP	DOWN	LEFT	RIGHT	PUSH	PULL
UP	**57**	1	1	1	0	0
DOWN	0	**58**	0	1	0	1
LEFT	1	2	**52**	3	1	1
RIGHT	2	4	1	**46**	2	5
PUSH	1	4	1	5	**48**	1
PULL	2	3	6	3	7	**39**
Hits	**95%**	**97%**	**87%**	**77%**	**80%**	**65%**

(b) Standing up

Expected Gesture \ Predicted Gesture	UP	DOWN	LEFT	RIGHT	PUSH	PULL
UP	**58**	0	1	1	0	0
DOWN	0	**56**	0	2	1	1
LEFT	0	1	**53**	1	1	4
RIGHT	0	1	0	**55**	4	0
PUSH	0	0	0	2	**57**	1
PULL	3	3	2	2	1	**49**
Hits	**97%**	**93%**	**88%**	**92%**	**95%**	**82%**

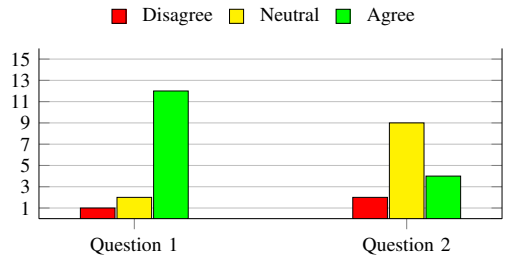
As shown in the table, hits were very high for most of the gestures in both postures, although, for the sitting down posture, they were a little lower. The pull gesture was the least effective one, specially in the sitting down posture. We were aware that this would be challenging, because this gesture is not ergonomic enough, mainly in the sitting down posture given the obstacle caused by the backrest and the armrest of the sofa. However, the pull gesture was kept in the gesture set in order to evaluate arm movements in all possible ways.

The Up and down gestures showed better accuracy, possibly as a consequence of causing a clear change on the accelerometer data when opposing to the Y axis. Some participants preferred performing the up gesture simply raising their wrist up, while others performed more elaborated gestures, twisting their wrist to have the hand palm towards the ceiling. However, this behavior did not interfere on the hit rate.

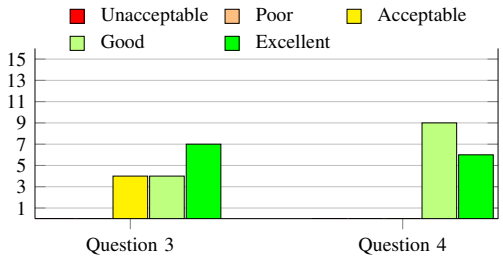
Fig. 6a) presents a summary of questions 1 and 2 from the questionnaire answers. Question 1 is about the participant experience during the experiments and about 12 of 15 participants said they enjoyed it. Question 2 covered if they would use our approach to control a TV. About 9 of 15 participants said they could not decide yet, but some of them stated that they would consider the idea if they could have control over more devices.

Questions 3 and 4 from the questionnaire were about the experience with the prototype trigger and with the proposed gestures (Fig. 6b). In question 3, 7 and 4 participants qualified the trigger detection as excellent and good, respectively. We should note that none qualified it as poor or unacceptable, but 4 participants qualified it as acceptable and this reveals that we have to fix some issues about the hand and arm positions of the trigger, in order to provide a better user experience. For question 4, 9 and 6 participants qualified the proposed gesture set as good and excellent, respectively.

Finally, we requested more detailed explanations to the participants in the questionnaire. Some of them argued that



(a) User experience during experiments (Question 1) and possibility of technology adoption (Question 2).



(b) Trigger (Question 3) and proposed gestures (Question 4).

Fig. 6: Answers for user experience questionnaires.

they would like to watch TV in a more relaxed position than the ones defined in the test, such as laying down in a sofa. Also, most of the participants mentioned enjoying the feeling of having the "power in their hands". We observed that this feeling was intensified by the visual and audio feedback on the TV, and also the smartwatch vibration.

VI. Conclusion

We proposed a fully functional prototype for smart TV gesture control via smartwatches, entitled Wrist Player. We analysed a classification algorithm performance on two different postures, where participant executed six different gestures with repetitions. The average hit rate was very high for both postures, even using only one recorded templates set before each test session. We also evaluated the user experience, showing that although our prototype needs some improvements, it gives a very good experience and has potential to complement the traditional TV remote control.

VII. Future Work

We intend to improve the classification technique using other approaches such as the k-Nearest Neighbors method, and more training sets. Adding the gyroscope sensor data can also be useful and improve gesture and trigger detection. In the current project, we avoided the gyroscope due to some hardware instabilities. For example, in the Moto 360 this sensor was completely disabled after some system updates. We also plan to expand our approach to the Internet of Things, and allow users to control other devices such as air-conditioners and windows.

VIII. Acknowledgment

We would like to thank for collaboration from PixelLab members and volunteers who were very kind and supportive on elaboration of this paper.

References

[1] Rekimoto, Jun, "Gesturewrist and gesturepad: Unobtrusive wearable interaction devices," in *Wearable Computers, 2001. Proceedings. Fifth International Symposium on*. IEEE, 2001, pp. 21–27.

[2] Kela, Juha, Korpipää, Panu, Mäntyjärvi, Jani, Kallio, Sanna, Savino, Giuseppe, Jozzo, Luca, and Di Marca, Sergio, "Accelerometer-based gesture control for a design environment," *Personal and Ubiquitous Computing*, vol. 10, no. 5, pp. 285–299, 2006.

[3] Liu, Jiayang, Zhong, Lin, Wickramasuriya, Jehan, and Vasudevan, Venu, "uwave: Accelerometer-based personalized gesture recognition and its applications," *Pervasive and Mobile Computing*, vol. 5, no. 6, pp. 657–675, 2009.

[4] Kühnel, Christine, Westermann, Tilo, Hemmert, Fabian, Kratz, Sven, Müller, Alexander, and Möller, Sebastian, "I'm home: Defining and evaluating a gesture set for smart-home control," *International Journal of Human-Computer Studies*, vol. 69, no. 11, pp. 693–704, 2011.

[5] J. Speir, R. R. Ansara, C. Killby, E. Walpole, and A. Girouard, "Wearable remote control of a mobile device: Comparing one- and two-handed interaction," in *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices & Services*, ser. MobileHCI '14. New York, NY, USA: ACM, 2014, pp. 489–494.

[6] S. McGuckin, S. Chowdhury, and L. Mackenzie, "Tap 'n' shake: Gesture-based smartwatch-smartphone communications system," in *Proceedings of the 28th Australian Conference on Computer-Human Interaction*, ser. OzCHI '16. New York, NY, USA: ACM, 2016, pp. 442–446.

[7] Mace, David, Gao, Wei, and Coskun, Ayse K, "Improving accuracy and practicality of accelerometer-based hand gesture recognition," *on Interacting with Smart Objects*, p. 45, 2013.

[8] F. Kerber, P. Schardt, and M. Löchtefeld, "Wristrotate: a personalized motion gesture delimiter for wrist-worn devices," in *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*. ACM, 2015, pp. 218–222.

[9] J. Ruiz and Y. Li, "Doubleflip: A motion gesture delimiter for mobile interaction," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '11. New York, NY, USA: ACM, 2011, pp. 2717–2720.

[10] L. Porzi, S. Messelodi, C. M. Modena, and E. 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*, ser. IMMPD '13. New York, NY, USA: ACM, 2013, pp. 19–24.

[11] G. Costante, L. Porzi, O. Lanz, P. Valigi, and E. Ricci, "Personalizing a smartwatch-based gesture interface with transfer learning," in *2014 22nd European Signal Processing Conference (EUSIPCO)*, Sept 2014, pp. 2530–2534.

[12] C. Xu, P. H. Pathak, and P. Mohapatra, "Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch," in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, ser. HotMobile '15. New York, NY, USA: ACM, 2015, pp. 9–14.

[13] S. Sen, V. Subbaraju, A. Misra, R. K. Balan, and Y. Lee, "The case for smartwatch-based diet monitoring," in *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*. IEEE, 2015, pp. 585–590.

[14] A. Ferrari, D. Puccinelli, and S. Giordano, "Gesture-based soft authentication," in *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct 2015, pp. 771–777.

[15] A. Sarkisyan, R. Debbiny, and A. Nahapetian, "Wristsnoop: Smartphone pins prediction using smartwatch motion sensors," in *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*, Nov 2015, pp. 1–6.

[16] F. Waris and R. G. Reynolds, "Using cultural algorithms to improve wearable device gesture recognition performance," in *Computational Intelligence, 2015 IEEE Symposium Series on*. IEEE, 2015, pp. 625–633.

[17] R.-D. Vatavu and I.-A. Zaiti, "Leap gestures for tv: Insights from an elicitation study," in *Proceedings of the 2014 ACM International Conference on Interactive Experiences for TV and Online Video*, ser. TVX '14. New York, NY, USA: ACM, 2014, pp. 131–138.

[18] Seetharamu, Vinod Keshav, Bose, Joy, Sunkara, Sowmya, and Tigga, Nitesh, "Tv remote control via wearable smart watch device," in *2014 Annual IEEE India Conference (INDICON)*. IEEE, 2014, pp. 1–6.

[19] M. Pedley, "Tilt sensing using a three-axis accelerometer, freescale semiconductor application note,(2013), n," *AN3461, rev*, vol. 5.

[20] ten Holt, Gineke A, Reinders, Marcel JT, and Hendriks, EA, "Multi-dimensional dynamic time warping for gesture recognition," in *Thirteenth annual conference of the Advanced School for Computing and Imaging*, vol. 300, 2007.