Margherita Antona
Constantine Stephanidis (Eds.)

Universal Access in Human-Computer Interaction

Users and Context Diversity

LNCS 9739

10th International Conference, UAHCI 2016
Held as Part of HCI International 2016
Toronto, ON, Canada, July 17–22, 2016, Proceedings, Part III

3 Part III

Springer

GyGSLA: A Portable Glove System for Learning Sign Language Alphabet

Luís Sousa[1], João M.F. Rodrigues[1(✉)], Jânio Monteiro[2], Pedro J.S. Cardoso[1], and Roberto Lam[1]

[1] LARSyS and ISE, University of the Algarve, 8005-139 Faro, Portugal
luiscarlosrsousa@outlook.com, {jrodrig,pcardoso,rlam}@ualg.pt
[2] INEC-ID (Lisbon) and ISE,
University of the Algarve, 8005-139 Faro, Portugal
jmmontei@ualg.pt

Abstract. The communication between people with normal hearing with those having hearing or speech impairment is difficult. Learning a new alphabet is not always easy, especially when it is a sign language alphabet, which requires both hand skills and practice. This paper presents the GyGSLA system, standing as a completely portable setup created to help inexperienced people in the process of learning a new sign language alphabet. To achieve it, a computer/mobile game-interface and an hardware device, a wearable glove, were developed. When interacting with the computer or mobile device, using the wearable glove, the user is asked to represent alphabet letters and digits, by replicating the hand and fingers positions shown in a screen. The glove then sends the hand and fingers positions to the computer/mobile device using a wireless interface, which interprets the letter or digit that is being done by the user, and gives it a corresponding score. The system was tested with three completely inexperience sign language subjects, achieving a 76 % average recognition ratio for the Portuguese sign language alphabet.

Keywords: HCI · Gesture recognition · Sign Language · Assistive technologies

1 Introduction

Sign Language (SL) is a communication medium for the deaf and mute people, and Natural User Interface (NUI) is a term used for Human-Computer Interaction (HCI) where the interface is invisible or becomes invisible after successive user-immersion levels. Typically relies in nature or human natural elements.

Sign language uses manual communication and body language to convey meaning, which can involve simultaneously combining hand shapes, orientation and movement of the hands, arms or body, and facial expressions to fluidly express a speaker's thoughts.

In terms of NUI, currently there are several sensors with the ability of tracking and recognize body gestures, such as Kinect [6], Leap Motion [8] and Structure Sensor [11]. All these sensors have a great importance to the industry of

M. Antona and C. Stephanidis (Eds.): UAHCI 2016, Part III, LNCS 9739, pp. 159–170, 2016.
DOI: 10.1007/978-3-319-40238-3_16

gaming and user-machine interaction tools. These sensors, when supplemented with the appropriate software, have the ability to detect the body structure and/or the user's hand, and accurately replicate that structure on a 3D mesh, allowing gestures detection. Nevertheless, all these sensors are based on color cameras (RGB) and/or depth (infrared) and therefore have problems of space limitations, e.g., the user has to be located near the device and in the area where these cameras are pointing, otherwise, they will not work properly. In addition to spatial limitations, in most cases, these devices do not work well when they are near an infrared source (e.g., on sunlight) or in a room with fluorescent lamps.

In a system where the major interest is to develop a NUI to teach sign language alphabet to inexperienced persons, it is important that those persons can move openly in any environment with a system which should be free from "environmental" errors. A good solution to this problem is to develop a wearable glove, where the users can freely practice the signs, integrated with an application that can work in a standard personal computer or in any mobile device.

Wearable gloves are not a novelty, as examples from big commercial companies exist, see e.g., [3]. In 2012, Benbasat and Paradiso [1] described an inertial gesture recognition framework composed of three parts. The first, is a compact, wireless 6-axis inertial measurement unit to fully capture three-dimensional (3D) motion. The second part comprises a gesture recognition algorithm, that analyzes the data and categorizes it on an axis-by-axis basis, as simple motions with magnitude and duration. The third part allows an application designer to combine recognized gestures, both concurrently and consecutively, to create specific composite gestures that can then be set to trigger output routines. Mehdi and Khan [7], also in 2012, presented a sensor glove to capture signs of American Sign Language performed by a user and translates them into sentences of the English language. In that work, artificial Neural Networks (NN) are used to recognize the sensor values coming from the sensor glove. See [9] for another example of the use of NN to recognize American Sign Language words. In 2014, Praveen et al. presented an approach for interpreting the sign language using a portable smart glove [10]. Kim et al. [5], in 2015, presented a sign language recognition system using a data glove composed of 3-axis accelerometers, magnetometers, and gyroscopes. The information obtained by the data glove is transmitted to an host application, implemented on a MS Window program, running on a personal computer (PC). Next, the data is converted into angle data, and the angle information is displayed on the host application and verified by outputting 3D models to the display.

In this paper a completely portable glove system is presented, to teach a sign language alphabet. The main contribution of the paper stands in the developed NUI: a portable system integrating a wearable glove with a game application (used to learn the alphabet). The system can be used anywhere with a mobile device, or in standard personal computers.

2 Sign Language Alphabet Learning System

This section describes the implementation of the GyGSLA system, created to help people in the process of learning a sign language alphabet. To do it, (a) a computer game-interface and (b) a hardware device (wearable glove), called GyroGlove (GyG), were implemented. When interacting with the computer or mobile device interface, the user is asked to represent alphabet letters by replicating the hand and fingers positions, which are obtained using the GyroGlove module. Then, the GyroGlove uses a wireless interface to send the acquired information to the computational device, which interprets the gesture that is being done by the user and gives it a corresponding score. In the following, we start by describing the implementation of the GyroGlove, followed by the game interface.

2.1 GyroGlove

The GyroGlove is based on Inertial Measurement Units (IMU) sensors (see e.g. [1,4]) to detect the user's 3D rotation of the hand and fingers positions. An IMU sensor is an electronic device capable of measuring various types of inertial forces. Depending on the composition of the device, it can be formed by several independent sensors such as gyroscopes, accelerometers, magnetometers and, less commonly, altitude sensors (atmospheric pressure). Each IMU present in the GyroGlove contains an accelerometer and a gyroscope, each one of three axis, thus becoming an IMU sensor with 6 degrees of freedom (DoF).

Accelerometers can measure acceleration (in g-force) from one to three axes. Those that support three axes are the ones that have more functionality. Although accelerometers are fairly accurate when acquiring data from devices stable during long periods of time, they are unstable in short time data acquisitions. In other words, when an accelerometer is placed in a device that moves or shakes significantly, it is very difficult to accurately measure all the acceleration data. The gyroscope is a sensor capable of measuring the angular velocity (measured in degrees per second, °/s), being used to obtain the moving direction of an object. The data acquisition range of the gyroscope can be selected taking into consideration that there is a trade-off between range and accuracy. If a low range is chosen, the device can be quite accurate, but cannot exceed its maximum angular velocity. On the other hand, if the maximum value is too high, the accuracy is reduced. Thus, those values must be adjusted according with the desired application. There are also gyroscopes able to measure the angular velocity, from one to three axes. While the gyroscopes are very accurate in measuring angular velocity, they suffer from drift problems measuring low constant angular velocities, even when immobilized. Unlike the accelerometer, the gyroscopes are very accurate in short periods of time, and inaccurate otherwise. In such a way, accelerometer and gyroscopes should and can be combined.

The GyroGlove has several IMU MPU-6000 sensors (Magnetic Pickup Unit) [4] and a central controller module whose function is to program and configure all the IMU sensors and to serve as interface between the glove and the computational device. Each sensor has an accelerometer and a gyroscope, both

3-axis, with configurable ranges varying from $\pm 2g$ up to $\pm 16g$ and $\pm 250°/s$ up to $\pm 2000°/s$, respectively. As the sensors are placed in the user's hand, high values of g-forces or high angular velocities are not expected. Thus to keep the values as much accurate as possible, it was decided to use a range of $\pm 2g$ for the accelerometers and $\pm 500°/s$ for the gyroscopes.

To minimize the number of sensors without constraining the capability of capturing the rotations of the whole hand, so as to be able to extract the 3D rotation of each finger and also the hand itself, the sensors are strategically placed on the glove. The hand/finger bones that were considered more important for the gesture representation were the distal, intermediate and proximal phalanges (see Fig. 1 left). In this sense, the use of 11 sensors on the locations presented in Fig. 1 right was decided, enabling the extraction, without limitation, of all the rotational data of the fingers and hand. Because the distal bones are very short, it is very uncommon to fold this finger part without moving his adjacent one. The exception is the thumb, which is the only finger that does not have this characteristic. Thus, in the thumb it was decided to put two sensors, one in the distal phalange and another in the proximal phalange. All other fingers have also two sensors, in the intermediate and proximal phalanges. The last sensor (sensor 11) is placed on top of the hand, next to the main module, and is used to extract the overall orientation of the hand. This latter sensor is the basis for the correlation of all other sensors, as explained next.

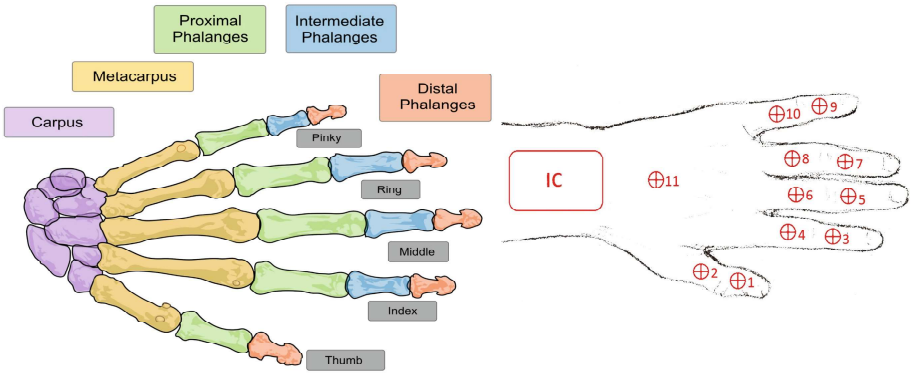


Fig. 1. On the left, hand bones anatomic names (adapted from https://en.wikipedia.org/wiki/Hand). Location and number identification of the sensors $(1, 2, \ldots, 11)$ and main module (IC) on the right .

In summary, the hardware system consists of 3 major modules: (a) a main controller, (b) a receiver connected to the PC or mobile device, and (c) eleven MPU-6000 sensors. Each IMU sensor has a size of 17×23 mm (millimeters).

The main controller, (a), acts as the intermediary interface between the sensors and the application on the computer, being responsible for programming/configuring all sensors and for sending the data via Bluetooth to the

receiver on the computational device. The main controller consists of: (i) a microcontroller ATMega Atmel 328p with a 8 MHz oscillator; (ii) a USB-UART converter (FT232RL) that converts the Micro USB port data to the microcontroller and vice versa; (iii) a LiPo battery charger (MCP73831); (iv) a voltage supervisor circuit (BD523G); (v) a LED emitting a warning to the user if the battery is low; (vi) a voltage regulator (TPS13733), that regulates 5 V from the USB or battery voltage to the main 3.3 V of the circuit; (vii) a charge distributor (LTC4413) that provides an automatic way of selecting the power source; and (viii) a Bluetooth module (HC-06). The battery is used in the circuit when it is the single power source. The USB becomes the main power source when connected, regardless of how many sources are available (USB and/or battery).

The receiver, (b), is a Bluetooth module similar to that used in the controller, but with the ability to be used as a master device, i.e., it has the initiative to bind to other Bluetooth modules. Similar to the controller circuit, the circuit of the receiver has: (i) a USB-UART converter (FT232RL), converting the USB micro port data to the Bluetooth, and a (ii) Bluetooth module. Mobile devices (tablets or smartphones) only need to be equipped with a standard Bluetooth for the system to work.

Figure 2 top shows: on the left the prototypes of the printed circuit boards (PCBs), and on the right the prototype of the GyG module assembled on the glove. To achieve the required performance, the transmission of data to the computational devices has to be done as fast as possible. For this reason, as represented in Fig. 2 bottom, the microcontroller and Bluetooth modules are programmed to use a 115200 bps (bits per second) transmission rate, which is the maximum possible rate allowed by the devices. The rate limitation results from the fact that the 8 MHz oscillator on the micro-controller does not allow higher speeds. For the same reason, the interface between the microcontroller and the sensors communicates at a top speed of 2 MHz SPI (Serial Peripheral Interface). The maximum frequency of the ATMega328P on the SPI interface is 1/4 of the oscillator frequency. These transmission speeds allow the system to update all data on the computer at a frequency of 33.3 Hz (validated by practical speed tests).

Finally, (c), each of the 11 MPU sensors placed in the glove (see Fig. 1 right), integrates a DMP (Digital Motion Processor) [4] that is used to process complex algorithms of 6-axis motion fusion. These algorithms are proprietary, registered by InvenSense [4], and the mode of operation is not of public knowledge. One solution to eliminate or reduce the problems of inaccuracies of the sensors is to use a filter to join the data from the accelerometer and gyroscope (existing in the same silicon die). This filter combines in a single expression, a low pass filter for gyroscope and a high pass filter for the accelerometer. But, by doing this, the angle around the z-axis of the accelerometer cannot be calculated as was done with the y- and x-axis, because accelerometer computations rely on gravity pointing in the z-axis (this is a phenomenon known as Gimbal Lock, which is a problem/limitation that occurs when working with Euler angles).

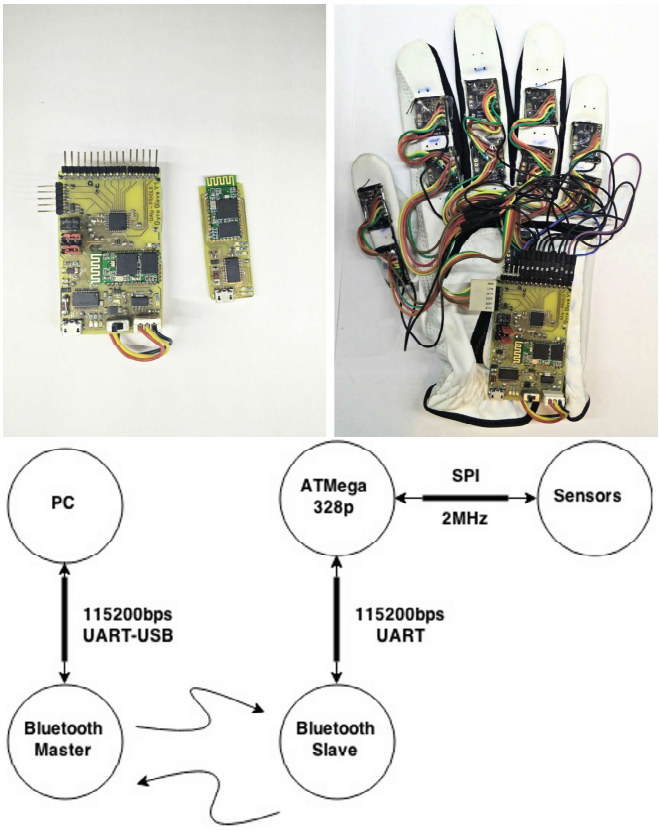


Fig. 2. On the top, the PCB prototype on the left, and the assembled glove on the right. On the bottom, the communications scheme between the GyG and the PC.

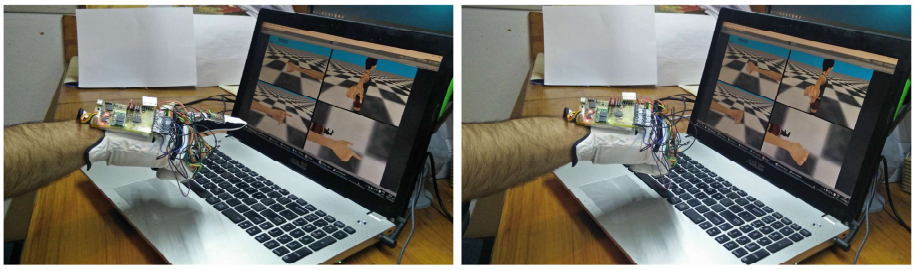
This implies that the angle with the z-axis can only be calculated using the gyroscope, therefore, this angle will suffer from a small drift along time.

As a result, the DMP system has the highest importance. This proprietary system uses quaternions, that do not suffer from the Gimbal Lock limitation. The quaternion is an alternative way to represent an angle on a 3D space: $Q = q_w + q_x x + q_y y + q_z z$, where q_x, q_y and q_z are the values of the position direction vector and q_w is the rotation about this axis, formed by the direction vector. These are necessary for the calibration and communication with the application, as shown next.

2.2 Application

The application has two main modules: (a) calibration and (b) game-learning application for the sign language alphabet.

For calibrating purposes, (a), an application was made using the Unity 3D [12] software, being its development out of the scope of this paper. One of the main reasons to use Unity is the easy deployment to different mobile platforms (e.g., Android and iOS). The application replicates the user's hand position, using a 3D model of a human hand, as shown on Fig. 3 top row, where two different situations are shown: hand with index finger pointing and a with tilted position.



1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte
$	X_m	X_l	Y_m	Y_l	Z_m	Z_l	W_m	W_l	i	CR	LF
	X		Y		Z		W				

Fig. 3. Top row shows examples of the user's hand and the replication on a human hand 3D model. In the bottom, the structure of the transmitted packet between the GyroGlove and the computational device.

For the calibration, and the transmission of the angles between the GyG and the PC/mobile, the DMP system can itself calculate the exact rotation of each sensor but needs an initial set up configuration, for which there is no control. This auto-configuration aims to reduce and possibly eliminate the drift problem caused by the gyroscope, as explained earlier. After the system starts, all the sensors enter into an auto-configuration procedure which lasts between 10 and 20 seconds. When the procedure finishes, all sensors will be stable. During this time the extracted rotational data is not usable, suffering from extreme drift. To correct the offset problem it is necessary to know, a priori, the state of the rotation of all sensors, i.e., to position all sensors in a known orientation.

Being Q a quaternion with an unknown direction, setting $K = Q^{-1} \times Q$ gives a quaternion with no rotation, stabilized. In the application running on the computer/mobile device, objects that replicate the users' hand orientation require initialization. In a process guided by the calibration application, the user has to place his hand with a certain orientation, before powering up the system. Being i the sensor number and t_1 the time at which the initial configuration is finished, all rotations of the sensors in time t_1 are stored in the quaternion U_{it_1}. Now, if $K_{it} = U_{it_1}^{-1} \times S_{it}$, being S_{it} the data from the sensors after the initial setup configuration (i.e., after t_1), then K_{it} contains the final rotations which is

equal to the user's hand rotations. If the user's hand is not placed with all the sensors stabilized, then K_{it} must be obtained from $K_{it} = (U_{it_1}^{-1} \times D_i)^{-1} \times S_{it}$, where D_i is the quaternion with the offset rotation of each sensor, available before powering up the system.

To initialize the system, the user must keep his hand as straight as possible on a table or horizontal plane during the setup. Then, $K_{it} = U_{it_1}^{-1} \times S_{it}$ is applied, avoiding the offset rotations of the sensors, with the exception of the ones positioned on the thumb. In fact, as can be easily seen from Fig. 2 top right, when the hand is positioned horizontally on a table, all sensors are straight (without rotation), with the exception of the two sensors in the thumb. In this particular case, the estimation of the associated rotation of the two thumb sensors is necessary and applied to each of them, $K_{it} = \left(U_{it_1}^{-1} \times D_i\right)^{-1} \times S_{it}$. We opted to apply an initial rotation of $-45°$ (obtained empirically) to both x- and y-axis.

Having now a way to compute the angles, the communication between the glove and the computer/mobile device is made using a character oriented packet with the format shown in Fig. 3 bottom row. Each sensor transmits a packet with their associated quaternion data. The packet starts with a \$ character, followed by the quaternion data, X, Y, Z and W, formed by 16-bit sets, i.e., two sets of 8 bits (X_m, X_l), where m and l represent the most and less significant elements, respectively, followed by the index number of the respective sensor i, (as identified in Fig. 1 right). The end of the packet is done by CR (Carriage Return) and LF (Line Feed) characters.

After developing the GyroGlove and implementing the transmission data between GyG and the computer/mobile device, the game-application, (b), was implemented (again using the Unity 3D) to help the learning of a sign language with GyG.

As the GyroGlove only allows (for now) the detection of a "static" position of the rotation of all fingers and hand, there is no way to learn sign language words, phrases or sentences. On other words, the system only accepts sign language alphabet, where each letter and digit possesses a direct relationship with the hand position and fingers. Since the GyGSLA system was developed in Portugal, the alphabet and digit used in the initial testing (see Fig. 4 top row) was the Portuguese Sign Language (PSL)[2]. Other alphabets can easily be integrated in future developments.

To recognize each of the hand's positions shown in Fig. 4, it is necessary to compare all the rotations of the sensors with all the rotations associated with a particular letter of the alphabet. Several classification methods could be used (e.g., NN as in [7]). However, since the goal was to be able to run the application in any mobile device, a "simpler" and less CPU/GPU demanding method was chosen.

Instead of comparing the quaternion of each sensor, a comparison was made between the quaternion of a pre-defined set of adjacent sensors, as well as the relation between the quaternion of the hand sensor (number 11 on the glove) and the vertical axis. This last one is used to determine the overall direction of the hand (face up, down, left etc.). There are 11 distinct relations (R_j). To obtain

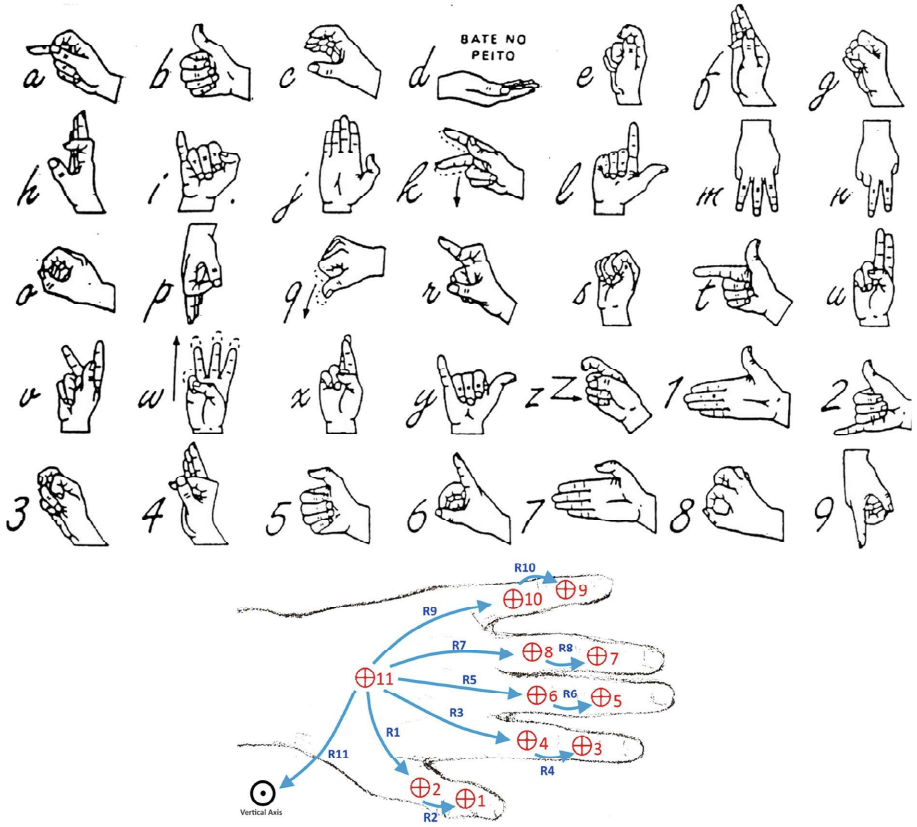


Fig. 4. On the top, the Portuguese sign language alphabet (adapted from [2]). On the bottom, the association (R_k) between adjacent sensors and between sensor 11 and the vertical axis.

these relations, the angle of each rotation is calculated between all sets of related sensors. As illustrated by Fig. 4 bottom, each sensor is associated with one or more sensors. For example, sensor number 8 (ring proximal finger) is related to sensor 7 and 11, identified as relation R_8 and R_7, respectively.

As a quaternion defines a rotation and not a vector in space, the rotation of each quaternion to a known vector must be applied, in our case, $\overrightarrow{V}_f = (0, 0, 1)$. From $\overrightarrow{V}_i = K_i \times \overrightarrow{V}_f$ results a vector with the rotation of K_i (raw quaternion data obtained from the sensors).

After obtaining the orientation of each sensor in the 3D space, $\overrightarrow{V}_i$, the absolute angle between the orientation vectors is calculated. These angles are computed between each related sensors pair (shown in Fig. 4 bottom), applying formula: $A_j = \left|(180/\pi) \times \arccos\left(\overrightarrow{V}_n \cdot \overrightarrow{V}_m\right) / \left(\left\|\overrightarrow{V}_n\right\| \left\|\overrightarrow{V}_m\right\|\right)\right|$. Angles A_j range from 0° to 180° and represent the angle formed between each set of related

sensors (n and m, with $n, m \in \{1, 2, \ldots, 11\}$) and also between the main hand sensor (sensor number 11) and the vertical vector, $\overrightarrow{V}_{up} = (0, 1, 0)$.

Before deploying the application, the reference positions for each letter/digit are obtained and recorded. These reference positions are later used to evaluate the user's hand position when he tries to perform some letter/digit of the sign language alphabet. As previously, this is done by calculating the angles, A_j, between each sensor and its reference sensor, for each letter and digit of the alphabet, $\{a, ..., z, 1, ..., 9\}$. The reference values include the minimum and maximum angles, such that: $Smin_{j,l} = \min\{A_j[m] : t - \Delta_s \leq m \leq t\}$ and $Smax_{j,l} = \max\{A_j[m] : t - \Delta_s \leq m \leq t\}$, where $l \in \{a, ..., z, 1, ..., 9\}$, $A_j[m]$ is the angle at instant m and Δ_s comes from the fact that the hand's position needs to hold for a certain time (Δ_s seconds) to be associated with a letter, and posteriorly validated.

Whenever a new hand data is available, the application computes the A_j angles and checks if they are comprised between $Smin_{j,l}$ and $Smax_{j,l}$ for each letter and during a minimum period of time, $\Delta_s = 1s$ (set empirically). In other words, if during Δ_s seconds the inequalities $Smin_{j,l} - \Delta_m \leq A_j \leq Smax_{j,l} + \Delta_m$ - where Δ_m is a margin value in order to allow adjusting the detection sensitivity (set to 20) - and $l \in \{a, ..., z, 1, ..., 9\}$, are verified then a positive gesture detection is considered.

Figure 5 shows some examples of the game-application developed for the learning of sign language alphabet. The interface consists of 4 parts: (1) the letter and hand's position of the corresponding image, (2) a visual input of the hand position replicated in a 3D human model, (3) a scoring system, and (4) a level bar to indicate how close the user is standing next to the goal. On the left it shows the system working in a portable computer, and on the right on a Android mobile device.

3 Tests and Results

When the users interact with the application, random letters are shown. The user then has to replicate the image shown in the right side of the screen (PC case). When he/she is well succeeded, the system scores the associated result, according with the time spent to do the associated letter and the correctness of the hand's position, as well as the application game-level.

The implemented application was tested by three inexperience sign language evaluators. During these tests, the following parameters were quantified for each alphabet letter: (i) the success ratio, or percentage of letters that were correctly recognized; (ii) the failure ratio; and (iii) the average time in seconds taken by the evaluators to replicate the letter that was shown. Each evaluator was identified by T1, T2 and T3 and each one of them respectively tested all letters for 5, 8 and 10 repetitions.

The results of the system regarding all the letters and digits, achieved an average success ratio of 76 %, with an average time to detect each letter/digit of approximately 3.6 s. Although this does not seem very generous, 35 % of the

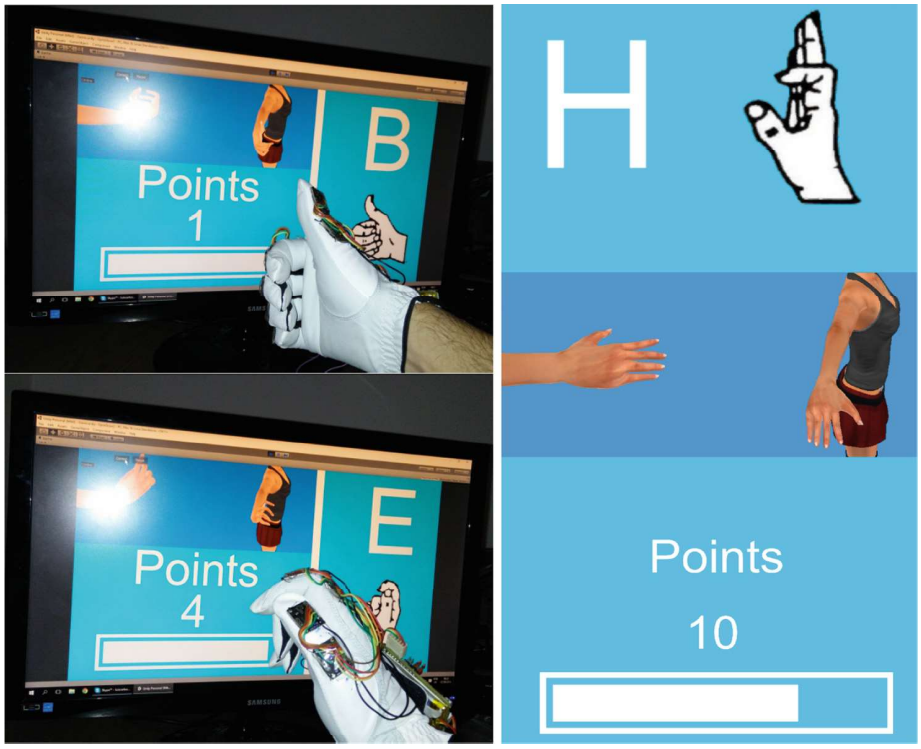


Fig. 5. Application for learning stimulation of alphabet sign language. The user is encouraged to replicate the hand position. Left, the system working in a portable computer, on the right in a Android mobile device.

letters/digits had a success ratio over 90 % (many of those achieved 100 %), and only 19 % had a success rate below 70 %. The tests indicated that some letters are particularly difficult to be detected by the system, such as the R, Q, U and V (with a resulting average of success ratio around 46 % and a detection time of approximately 6 s). As a note, most of the failure ratio was due to a system inaccuracy created by the estimation of the initial rotation of the two sensors placed on the thumb.

4 Conclusions

In this paper, a portable NUI system that supports the learning of a sign language alphabet was presented. The system combines a computer/mobile device interface-game and an hardware device - wearable glove. When interacting with the computer or mobile device, the user is asked to represent sign language alphabet letters by replicating the hand and fingers' positions, which are obtained using the wearable glove. The glove then sends that information to the computer/mobile device using a wireless interface, which interprets the letter that is

being done by the user and gives it a correspondent score. The glove uses IMUs with accelerometer and a gyroscope, each one of 3 axis, thus becoming an IMU sensor with 6 DoF.

For now, the system presents satisfactory success results, and it was presented in a personal computer, currently being converted and tested in mobile devices. This is expected to be a smooth transition, since the software was already implemented having this in mind. In terms of future work, both the NUI and the accuracy of the system will be improved, expanding the system to also detect and train words.

Acknowledgements. This work was partially supported by the Portuguese Foundation for Science and Technology (FCT) project PEst-OE/EEI/LA0009/2013.

References

1. Benbasat, A.Y., Paradiso, J.A.: An inertial measurement framework for gesture recognition and applications. In: Wachsmuth, I., Sowa, T. (eds.) GW 2001. LNCS (LNAI), vol. 2298, pp. 9–20. Springer, Heidelberg (2002)
2. Vivendo em Silencio: Dactiologia portuguesa. https://vivendoemsilencio.files.wordpress.com/2010/07/dactl-lgp.jpg (2015). Accessed 15 June 2015
3. Fujitsu: Fujitsu glove-style wearable device. http://vandrico.com/wearables/device/fujitsu-glove-style-wearable-device (2015). Accessed 14 Dez 2015
4. Invensense: InvenSense. http://www.invensense.com/ (2015). Accessed 15 June 2015
5. Kim, K.-W., Lee, M.-S., Soon, B.-R., Ryu, M.-H., Kim, J.-N.: Recognition of sign language with an inertial sensor-based data glove. Technol. Health Care **24**(s1), S223–S230 (2015)
6. Kinect.: Kinect for Windows. http://goo.gl/fGZT8X (2014). Accessed 10 Nov 2014
7. Mehdi, S.A., Khan, Y.N.: Sign language recognition using sensor gloves. In: Proceedings of the 9th International Conference on Neural Information Processing, vol. 5, pp. 2204–2206. IEEE (2002)
8. Motion, L.: Leap motion. https://www.leapmotion.com/ (2014). Accessed 10 Nov 2014
9. Oz, C., Leu, M.C.: American sign language word recognition with a sensory glove using artificial neural networks. Eng. Appl. Artif. Intell. **24**(7), 1204–1213 (2011)
10. Praveen, N., Naveen Karanth, M.S., Megha.: Sign language interpreter using a smart glove. In: International Conference on Advances in Electronics, Computers and Communications, pp. 1–5. IEEE (2014)
11. Structure: Structure sensor. http://structure.io/ (2014). Accessed 10 Nov 2014
12. Unity: Unity 3D. https://unity3d.com/pt (2014). Accessed 10 Nov 2014