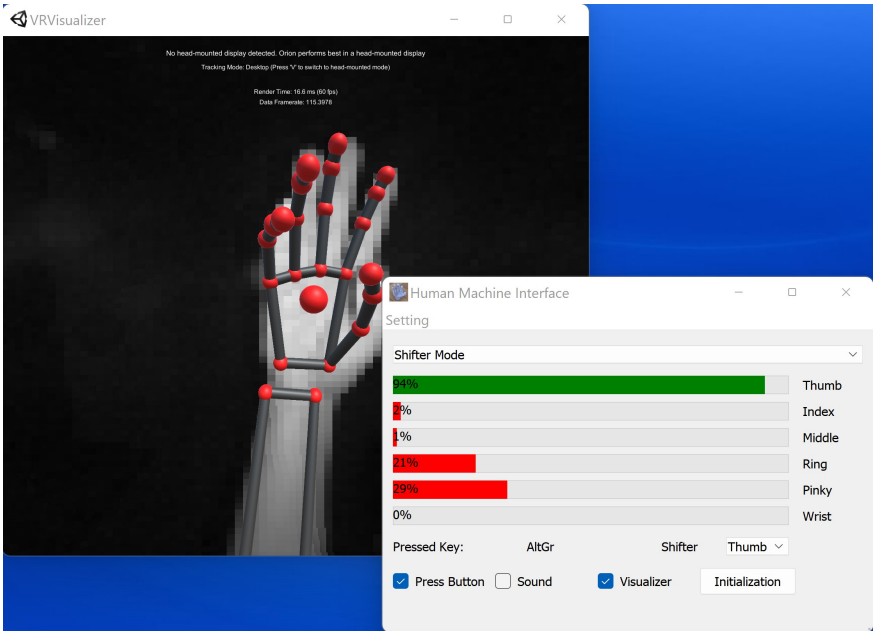
ATC – Assistive Technologies Challenge

Development of human-machine interface based on detection of fingers movements



Authors:
Odile Andres
Nada Guerraoui
Thomas Peeters
Brahim Rejeb

June 10, 2022

EPFL HACKAHEALTH

Contents

A big thanks to the whole hackahealth team, the teachers and especially our challenger who gave us excellent feedback throughout the project which made it a success. We also would like to thank the ultra Leap team for answering to all our questions about LeapMotion. It was a great technical and human experience, so thank you very much and we hope that this device will fulfil its mission by helping the challenger to regain his mobility.

Figure 1: Final solution

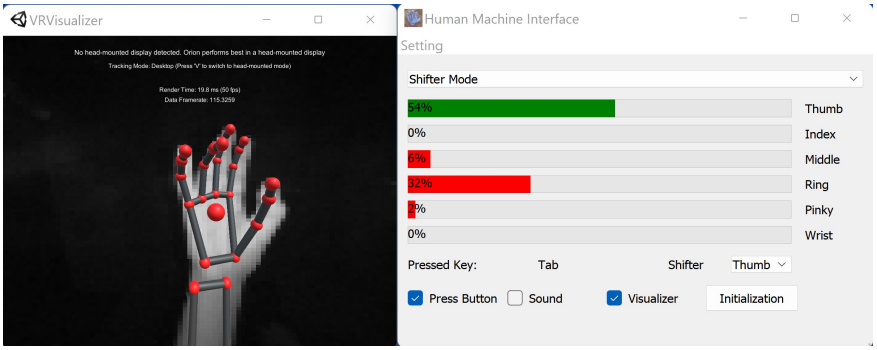


Figure 2: Final solution

HACKAHEALTH

1 Introduction

This project was part of the Assistive Technologies Challenge organized by EPFL for the Spring Semester 2022. The goal of this device is to press keys on a keyboard depending on specific movements of the left hand of our user. The challenger had a stroke and has now limited control of the movements of the left part of his body. Using this device will allow the challenger to use his computer with his two hands and to make some combinations of keys that were before difficult with only one hand such as:'Shift' + 'Letter' to put them in capital or 'AltGr' + 'other key' to get other symbols such as @.

However, the device also has an important medical aspect because it aims to help the challenger to regain some mobility. As he explained to us, after the stroke he was not able to walk and when he started walking again, he had to concentrate entirely on his left foot first and was not able to do anything else at the same time. By doing this exercise every day, the walking movement has become more intuitive and it is now easier for him. As he uses his computer every day, doing specific movements with his left hand can also make these movements more intuitive as he goes along. For this device to be used often, it must be robust, easy to understand, and has a user-friendly interface.

This documentation will go first over the general principle of the device. Then, the back-end and the front-end part of the software will be described. A description of the actual performance and some propositions for further step improvements will be given at the end.

2 Solution

2.1 General principle

The product is a software that detects the residual movement of the fingers of the left hand to select keys on a keyboard using Leap Motion. The controller is attached to a positioning mechanism with six degrees of freedom. The solution has an intuitive interface, a robust back-end, and an easy-to-deploy setting enabling the user to easily control the keyboard. Please refer to the user manual to correctly deploy the product.

2.2 Components and Assembly

The components needed for the solution are listed here :

Component	Image
Leap Motion	
Manfrotto 196AB-2 Single Arm 2 Section	
Manfrotto 035 Super Clamp	
3D printed support for Leap Motion	
Socket Head Cap Screw, 1/4”-20	
Hex socket screw with flat point BN 24	
Cable extender for USB 2.0 or USB3.X	-

An assembly guide is provided in the user manual.

3 Back-end

3.1 General Principle

3.1.1 Leap Motion

To track the hand, we use a Leap Motion Controller. This device is composed of two cameras and some infrared LED. By illuminating the close space near the cameras with infrared light the Leap Motion is able to capture a user's hands and fingers. To use the Leap Motion on your computer, the UltraLeap's hand tracking software must be installed. The latest version is **Gemini** but it can for the moment only be used for C language. Therefore we decided to use the previous version **Orion 4.1.0** that can be used with python. The main improvement from Orion to Gemini is a better detection when there is a two-hand interaction. Since in our case our device will only be used for one-hand detection, using Orion or Gemini software gives similar results. Orion's software is able to discern 27 distinct hand elements (3D coordinates), including bones and joints. We can use those elements to compute different angle and distance measurements that will allow us to detect specific movements.

Figure 3: 3D points tracked with Leap Motion Controller

3.1.2 Movements

We have set up different movements according to the mobility of the user's fingers, which can also follow his progression. So, we have two types of movements: simple and advanced.

Simple movements: The different simple movements implemented are the bending of each finger and the extension of the wrist and are detected respectively if the value of those measurements reaches a threshold :

1. Distance between the tip of the thumb and the palm
2. Angle between the metacarpal and the distal bone of the index
3. Angle between the metacarpal and the distal bone of the middle finger
4. Angle between the metacarpal and the distal bone of the ring finger
5. Angle between the metacarpal and the distal bone of the pinky finger
6. Distance from the center of the palm to the origin (Leap Motion)

For this simple movement, we have also two different modes:

Shifter mode: We set one finger (one movement) as a shifter to switch between the keys. Then, we can execute the chosen key with any one of the other fingers. We thought of this because the user has greater mobility for the thumb compared to the others, we also let the user choose on the interface which finger he wants to use as a shifter. This will allow the user to be able to exercise all his fingers. This can also be useful if he loses mobility in his thumb and regains it in another finger.

All fingers used mode For this mode, each finger has its own key but the movement implemented is the same as before.

Advanced movements: The different advanced movements implemented are the bending of the thumb, the pinching of the 4 other fingers, and the thumb and the hand closed and are detected respectively if the value of those measurements reaches a threshold :

1. Distance between the tip of the thumb and the palm
2. Distance between the tip of the thumb and the tip of the index
3. Distance between the tip of the thumb and the tip of the middle finger
4. Distance between the tip of the thumb and the tip of the ring
5. Distance between the tip of the thumb and the tip of the pinky
6. Distance of the four fingers (pinky, ring, middle and index) to the center of the palm.

The goal is that the user can switch to the advanced movement according to his progression.

3.1.3 Performance

Performance window Since the user wants to regain mobility of his left hand. It is important to give some feedback on his mobility. For that, we measure for each fingertip how much it has moved during the execution of the program. To do this, we record the coordinate of each fingertip at the beginning of the program (resting position) and we measure at each frame the distance of each fingertip between its current position and the resting position. A pop-up window appears at the end of the program to show the user's

performance.

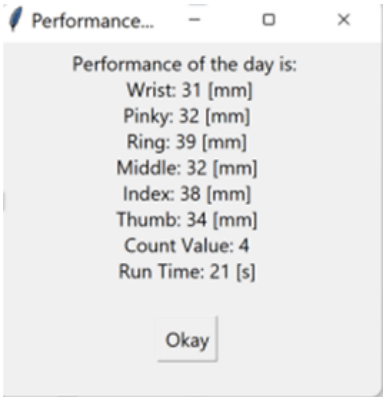


Figure 4: Performance window

Nevertheless, if we let the program run for a long time, the Leap Motion will accumulate some errors in the 3D coordinate and the performance can sometimes give wrong results (if another hand appears on the frame or if you move too much the hand). So, it is up to the challenger to estimate if the performance measured seems reasonable.
Those values are also saved on a performance.txt file and are in a specific format so that we can read them easily as .csv file :

	A	B	C	D	E	F	G	H	I
1	Date	Wrist [mm]	Pinky [mm]	Ring [mm]	Middle [mm]	Index [mm]	Thumb [mm]	Count Value	Run Time [s]
2	20220601-230432	28.29	89.98	94.99	89.09	94.15	119.23	2.0	17
3	20220601-230609	22.84	70.32	90.78	94.11	67.58	47.86	1.0	6
4	20220601-230735	66.14	87.21	107.77	120.44	118.28	110.31	3.0	16
5	20220601-230823	113.25	67.32	108.48	117.38	97.81	85.21	13.0	68
6	20220601-231030	525.12	73.92	84.82	90.47	89.94	65.49	5.0	71
7	20220601-231549	27.4	86.74	97.75	112.02	98.63	65.82	27.0	177
8	20220601-231951	114.93	80.81	133.4	170.92	168.73	134.86	8.0	66
9	20220601-233512	2.58	29.32	48.93	57.84	62.54	49.73	5.0	29
10	20220601-233627	38.07	59.93	75.99	85.24	71.25	52.19	6.0	32
11	20220601-233931	0.0	23.22	62.98	75.76	68.81	56.04	7.0	52
12	20220602-013929	0.0	60.77	77.36	78.72	65.2	52.02	0.0	12
13	20220602-014034	1.98	34.68	40.72	28.65	48.31	45.22	0.0	9
14	20220602-114711	148.19	716.06	170.72	173.45	164.66	127.95	4.0	50

Figure 5: Performance window

3.2 Robustness of the back-end

3.2.1 Precision of the movement measurement

If the user wants to use the program every day, it is important to make the back end as robust as possible to avoid having a key pressed when not wanted or a key not pressed when we want to press one. For that, many analyses have been made to make the detection of the different movements as precise as possible. At each run, all the movement measurements are saved in a .txt file that we can analyze later on Matlab. Here is an example :

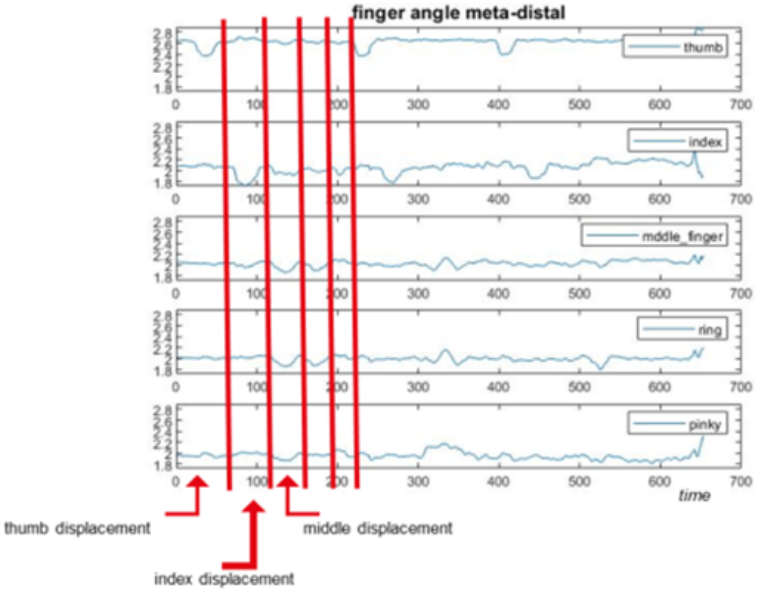


Figure 6: Analysis of the measurement

In figure 6, we bend slightly downward each finger (simple movement) one at a time and we measured for each frame the angle between the metacarpal bone and the distal bone of each finger (in radian, see figure 3). To have reliable detection, it is important that when we bend one finger, we observe a large value gap for the measurement of this specific finger and no change of value for the other fingers. If it is the case, it will be possible to set a threshold so that when it is reached, a key will be pressed. In the figure above we can see that we obtain good results for the thumb and the index because the angle value of those fingers decreases when they are bent and there is no change of value for the other fingers. Nevertheless, for the three last fingers (middle, ring, and pinky), we observe that when one is moved, there is a change of value for the three fingers which means that a movement for the three fingers will be detected. By doing a similar plot for each type of measurement we selected the best measurements for each movement (those explained in 3.1.2).

Some measurements have been made at the challenger office to check if we will detect correctly the different movements:

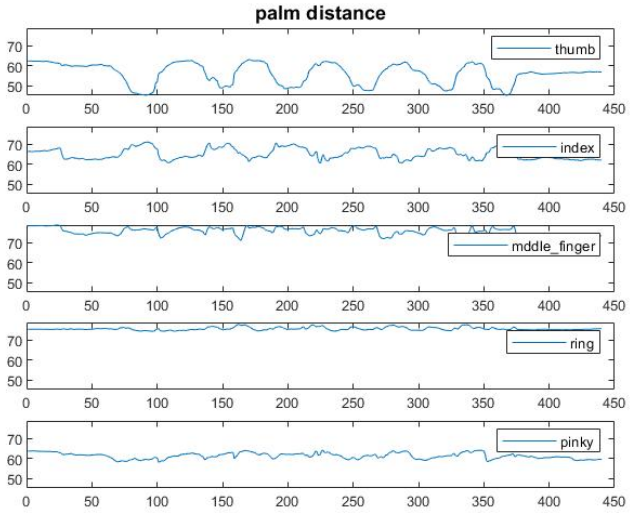


Figure 7: analysis of the thumb movement of the challenger, y axis in [mm] and x axis is the number of frame

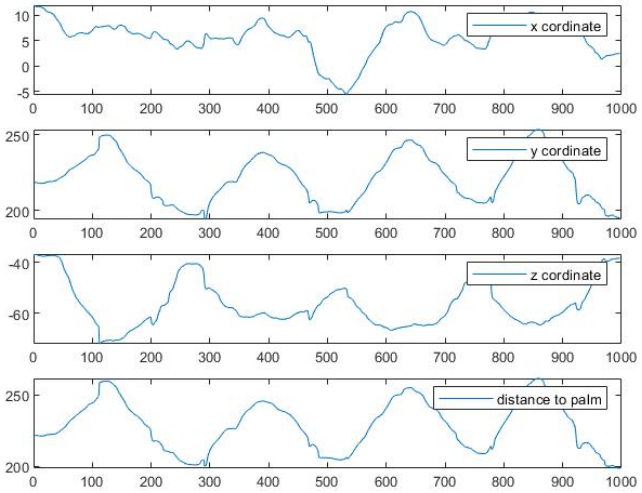


Figure 8: analysis of the wrist movement of the challenger, y axis in [mm] and x axis is the number of frame

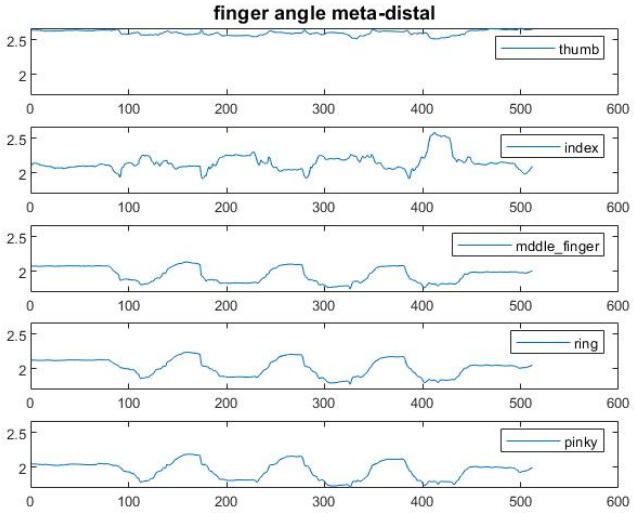


Figure 9: analysis of the middle finger movement of the challenger, y axis in [rad] and x axis is the number of frame

Here we can see that the thumb and the wrist movements are well detected. Again we observe that it is difficult to differentiate a movement for the three last fingers: middle, ring, and pinky. The challenger cannot yet do large movements with those fingers and the Leap Motion is not precise enough when the displacement of the fingers is too small.

3.2.2 Initialisation

We observe that between the different runs, if we put our hand exactly at the same position, the measurements will not be the same and there will be a bias between the measurements computed at each run. Here is an example where we did two tests in the same condition and where we moved the index finger:

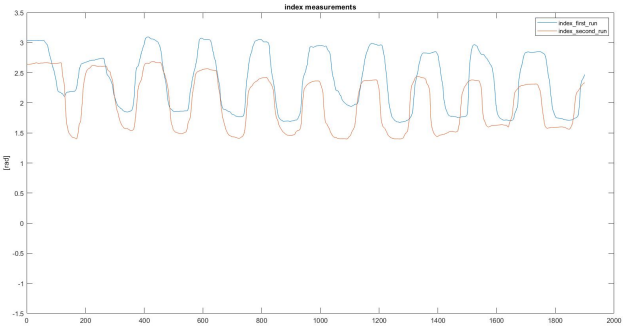


Figure 10: analysis of the measurement

Here we observe a bias between the two measurements that make the selection of a threshold for movement detection difficult to set. To solve this problem, an initialization is made at the beginning of the run where we save the initial measurements (position

at rest) and subtract them from the rest of the measurements made afterward. This normalization allows us to set a threshold that will work at each run:

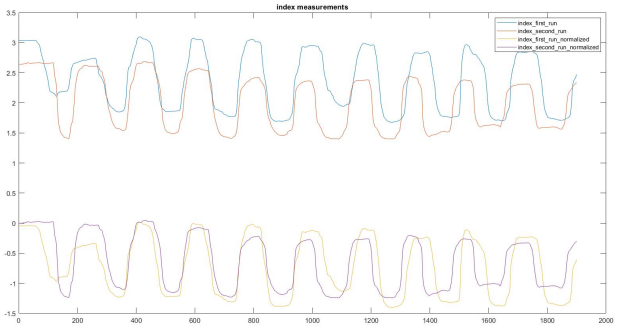


Figure 11: analysis of the measurement

Adding this initialisation improve a lot the final performance of the device. Nevertheless if the hand move too much when the application is running, it is important to redo this initialisation to improve the precision. This is possible by pressing the *initialisation button* on the interface.

3.2.3 Actual Robustness :

The device works well and is able to detect correctly the different movements. Nevertheless, there is still some limitation: First, as seen before, it is difficult to differentiate the movements of the middle, ring, and pinky finger if we bend them a little. Also when we do the extension of the wrist, the bending of one of the fingers can also be detected. The last error is the link with the leap motion and sometimes a hand is detected where there is none.

3.3 Code explanation : Back end

3.3.1 LeapMotion_detection.py and hand2keyPressedRehab.py

The back end part is mainly composed of two main files: **LeapMotion_detection.py** and **hand2keyPressedRehab.py**

The LeapMotion_detection.py file is used to extract the specific measurements of the Leap Motion that are useful. To extract the data from the Leap Motion we used their API that is fully described in this link : Leap Motion API. 4 datas are given to the hand2keyPressedRehab.py at each frame:

- palm : the coordinate of the center of the palm. This data is given so that in case the hand move too much, the initialisation is done once again
- distance : The 6 measurements computed to detect the 6 simple movements: bending of the five fingers and the wrist extension

- advance distance : The 6 measurements computed to detect the 6 advance movements.
- distance_performance : The distance for the five fingers and the wrist between their positions at rest and their actual position. Those values are used to compute the performance of the day.

The hand2KeyPressedRehab.py file is the main file that will receive the data coming from LeapMotion_detection.py and do the different computations for the different movements listed before to press a key if those movements are detected. This file makes also the communication with the mainwindow.py file (file for the user interface) so that it knows which key should be pressed, the values of the different thresholds, and get all the parameters set by the user.

This file will also record the performance to save them in a .txt file and make a pop-up message to give the best performance of the day of the user.

3.3.2 config.ini

In config.ini, different parameters are saved such as the values of the threshold, the last keys selected and different booleans to activate some functions such as *press_key* that when is set to true, the key is pressed when a movement is made or *show_visualizer* that show the visualizer from leap motion so that we can see how well our hand is detected. This config.ini file is updated at each end of the run.

4 Front-end

4.1 General Description

The graphical interface is designed using *QT Designer*, and generated using the following command: `pyuic5 -x file.ui -o file.py`
The resulting code is then linked with the back-end.
The graphical interface is composed of two different windows, coded with pyqt5. The principal one, *mainwindow*, is the first interface that will open when running the program. The second one can be opened by following *Setting/Access setting* .

4.1.1 Mainwindow

The mainwindow is composed of different elements:

- A menu to choose which kind of mode the user want to use: Shifter Mode, All Fingers Mode, Advanced Mode
- Evolution bars that show whether or not the key is pressed. When the threshold is reached, the bar turns green.
- Checkboxes, one to set the sound on, one to open the LeapMotion Visualizer and one to allow the application to activate keys on the keyboard

- An Initialization button, to redefine the resting position
- The rest of the window depends on the chosen mode :
  - Shifter Mode : a menu to choose the shifter finger and a label to display the current pressable key
  - The other modes : a menu for each finger to choose its function

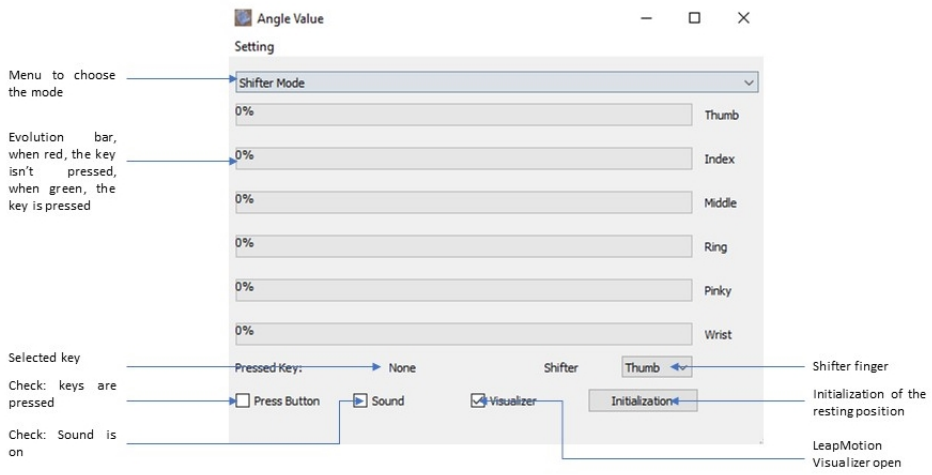


Figure 12: Mainwindow for shifter mode

4.1.2 Setting window

The setting window allows the user to change the threshold for each movement while running the application.

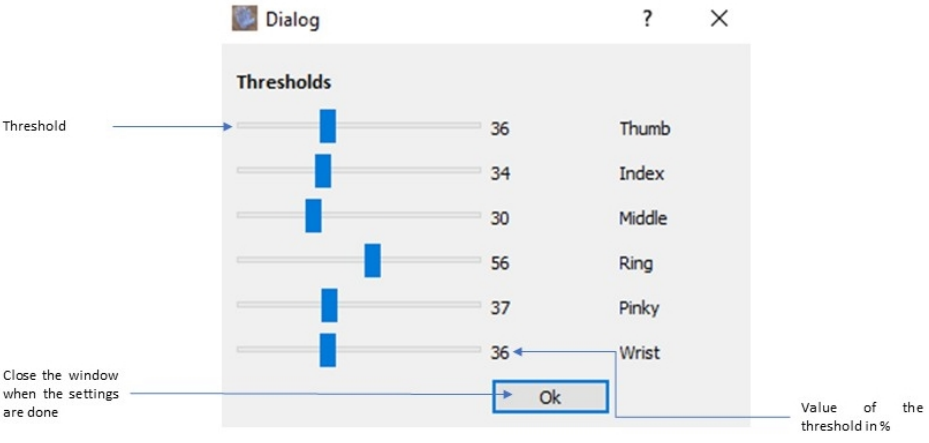


Figure 13: Dialog window for shifter and all fingers modes

5 Further Step

5.1 Improvement

Discussing with the challenger we agree that many improvements can still been made, here are some examples :

- **LeapMotion Software** : Leap Motion are continuously working on improving their softwares and services. If a new software version is available on python it should be important to use it to be more robust.
- **Implement it on a Microcontroller :** Being able to make the application work on a microcontroller would make the product more flexible and usable on all platforms (windows, MacOs, Linux..). For the moment it can only be used on Windows.
- **Using Machine Learning techniques :** Many ML models can be used for hand tracking and for gesture detection. The challenge here for using those models is first the dataset, it is difficult to collect a large dataset for training since our challenger can be tired and can not make to many movement during the day. Moreover, the hand movement capabilities of our challenger can rapidly evolve. It is therefore possible that the movements given to our ML model to train it no longer correspond to the challenger's current movements. This is why we have chosen a more classical detection method based on angle and distance measurements and a threshold that can be adjusted according to the challenger's current abilities. Nevertheless, it exists some ML model for hand tracking such as MediaPipe Hands. The advantage of this is that it only need a raw image as input and can therefore be used with any type of cameras and not only a Leap Motion.
- **Make the device wireless :** Since we fix the Leap Motion below the hand and close to the desk, we need a large cable to connect the Leap Motion to the computer. Making the device wireless could make the utilisation of the device more pleasant to use for the challenger. Here is a possible approach for this task which was discussed with Leap Motion support team : Please check `https://www.virtualhere.com/`

5.2 Medical objective of the device (Rehabilitation, advice from therapist)

The main objective of this device is to re-educate the user's hand and allow him to move it intuitively again. After a discussion with the user's therapist, we concluded that the goal of rehabilitation does not stop at the hand, but that it is necessary to move the whole arm to rehabilitate it (the device currently only works on hand movements). So the main improvement here is to think about how to include whole arm movements for future versions to have full arm rehabilitation. We also found that the position used for the Leap Motion was not recommended by the therapist (hand hanging over the Leap Motion) (see figure 14) and to avoid this she provided us with wrist support as can be seen in the figure 15 that allowed him to be upright and avoid the undesirable position except that unfortunately with this support the Leap Motion did not detect the user's

hand anymore. That's why we tried to add wrist movement which can help to include all the hand movements and also make the wrist work. For these reasons, it is perhaps important to think of something that allows the use of this support and the whole arm movements for the most optimal rehabilitation of the arm. We also conclude that talking to a therapist early in the development of the device is also crucial and can help find the most optimal solution for medical purposes.

Figure 14: Undesired position for the wrist

Figure 15: Wrist support recommended by the user's therapist

6 Conclusion

The final product is a functional device that can be used by anyone. It was a real challenge to make it as robust as possible and to have a clear, easy, and pleasant interface for the user.

After various discussions with our challenger, this device is only a start, there is still a lot of room for improvement as mentioned in the previous section and any new ideas for this device are also welcome! It has been a great experience and we encourage anyone who has the opportunity to do so to give it a try. Working on this project has been interesting both for its technical and human aspects. We were able to exchange a lot with our challenger who gave us relevant feedback on the project but also a lot of advice for our future life as engineers. Moreover, working in a team on a concrete project from design to final product to help someone was really motivating and meaningful.

If you have any questions on the device it is possible to contact us on our GitHub account that you can find on the GitHub of the project : link.