UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA MAGISTRALE **BIOMEDICAL ENGINEERING**

STUDY AND DEVELOPMENT OF HAND MONITORING ALGORITHMS FOR AWAKE NEUROSURGERY

RELATORE:

CHIAR.MO PROF. **MONTERIÙ ANDREA**

TESI DI LAUREA DI:

TROCONIS LUIGI GABRIEL

CORRELATORI:

DOTT. **FELICETTI RICCARDO**

DOTT. **IARLORI SABRINA**

A.A. **2021** / **2022**

Table Of Contents

List of Figures

List of Tables

For those who

gave me the wings

for the future

Mum and Dad

Chapter 1

Introduction

Awake surgery represents a particular type of neurosurgery during which the patient remains awake during the whole operation. Some applications of awake surgery are, for instance, resection of brain vascular malformations, tumoral removal, and to treat lesions in brain language areas. Awake surgery's main advantage over unconscious sedation is the precise localization method of brain functions, in such a way as to minimize the possible postoperative risk of permanent functional deficit. During surgery, the neurosurgeon excites the brain area, with the direct electrical stimulation, to localize accurately the functional brain area, which must be avoided by the surgical incision. The sedated patient, who remains conscious during the whole surgery, can do some tasks which are requested by the neuropsychiatrist. In the pre-surgery phase, that are going to be performed by the patient are chosen. The criteria for defining which tasks are to be proposed during the surgery depend on the brain localization of the target to remove, as well as the patient's pre-surgery capabilities (i.e., level of education). A possible objective is to verify the occurrence of movement disorders (e.g., freezing of the movement during the task), dysexecutive syndrome (e.g., increased delay in reasoning and answering), etc.

1.1 Motivation and objective

In the current surgical practice, the task assessment of the patient during the surgery is visually performed by the neuropsychologist, who considers if the patient's capabilities are preserved, and eventually drives the surgeon to reach the target by minimizing the damaging cut. Such evaluations are mostly subjective; therefore, the neuropsychologist must note any variations during the test and make sure the patient is not in discomfort and does not feel pain during the entire time. Due to the subjective neuropsychiatrist’s evaluation of the tasks performed by the patient during awake surgery, the purpose of this thesis is to define a novel approach to support the neuropsychiatrist’s decisions with a tool that provides objective and reproducible evaluations. In particular, the research topic

of this thesis is to support the neuropsychiatrist with an automatic method that is based on the assessment of the extracted signals from a stereo infrared camera, i.e., the Leap Motion Controller (LMC) in order to quantitatively evaluate the hand and fingers movement during selected exercises performance. Then, the signals are processed in order to define characteristic features to specify if the task has been performed well or not.

The most important advantage provided by such a solution is a better objectivity in brain mapping, easing the task of the neuropsychologist. Enabling for a precise and fast assessment, the patients' capabilities are better preserved, improving their post-surgical quality of life and autonomy. Moreover, the duration of the operation can be reduced, with lower dosage and risks associated to anesthesia.

Finally, a faster operation could improve the operating room efficiency.

1.2 State of the art

In the literature, tablet use during awake surgery has been suggested as a potential method to ensure as much support as feasible for the neuropsychologist throughout the operations [2], but no data have been stored by the authors and no criteria for decision support are studied. A portable and efficient system for presenting and simplifying challenging language and cognitive tasks during the surgery has been tested in [3], and it can also capture standardized patient response data. The platform is used to give language and cognitive activities to patients undergoing awake surgery. However, no features on hands gesture or motion in general are examined: the authors mainly focus on verbal intelligence, and no automatic assessment is shown. The LMC is employed in the literature for other purposes, such as in stroke rehabilitation [4] or to quantify bradykinesia in parkinsonian patients under deep brain stimulation [5].

1.3 Solution

This thesis proposes a support device to help the neuropsychiatrist's decision during the surgery for the evaluation of motor tasks, in particular for fine motor skills associated to the hand motion. The proposed study is characterized by the design and implementation of algorithms: to reconstruct the hand coordinates recorded by the LMC, to extract signals directly from the hand and fingers, and to define features that characterize whether the tasks proposed by the neuropsychologist are done correctly by the patient.

1.4 Thesis outline

The remainder of the thesis is organized as follows.

In Chapter 2, the leap motion controller is introduced. Its specifications, its working principles (hardware, software), and its previous applications are detailed.

In Chapter 3, materials and methods used in this thesis are presented.

In Chapter 4, the main results are organized, while the discussion of such results is proposed in Chapter 5.

Conclusions and future works are discussed in Chapter 6.

The bibliography and the appendices conclude this thesis.

Chapter 2

2 Vision system for hand tracking

2.1 Leap Motion Controller

The Leap Motion Controller (LMC) (Figure 2.1) is a hand tracking system that is based on a stereo infrared camera. It has a comfortable interaction zone, i.e., up to a maximum of 80 cm distance, a wide field of view, and it can work in several environmental conditions, because it is equipped with infrared LED that generate the necessary amount of light. Hence, it can be employed in several tasks where sensing the hand pose is a key feature. The original purpose of LMC from Ultraleap (The West Wing, Glass Wharf, Bristol, England, BS2 0EL) is to allow users to interact with digital information in a natural way by having their hands and fingers tracked optically. In fact, the LMC is compact, quick, and precise, and it can be used with Windows, macOS, and Linux computers for productivity applications, integrated into commercial-grade hardware or displays, or connected to virtual or augmented reality headsets for AR/VR/XR prototyping, research, and development [6]. Additional equipment and software to add other features to interact with the digitalized world are available in online repositories, e.g., in GitHub. Moreover, there are programs designed by UltraLeap company for this type of sensor which are used to play games, and many more.

Figure 2.1: Leap Motion Controller Sensor

2.2 Mechanism

2.2.1 Hardware

The LMC hardware is quite simple. It is characterized by two infrared cameras and multiple LEDs. The subject hands which are using the LMC device are illuminated by the LEDs with infrared light that is invisible to the human eye. The main infrared frequency is 850nm. Each camera is a 640x240 matrix, and the cameras are spaced 40mm apart. While the image sensors send the acquired data to the computer via USB to track his/her hands. The operation range is narrowed to the near-infrared spectrum for both light and cameras in LMC devices, which allows them to be robust in a range of environments and lighting conditions [7]. Figure 2.2 shows the internal components of LMC. The dimensions of LMC are small (80x30x11.3 mm). More practically, the LMC can be powered via a conventional 5v 2.0 USB port, which is also used to communicate the data to a connected device in real time, such as a laptop. The cameras operate at 120Hz, providing a smooth reconstruction of the hand.

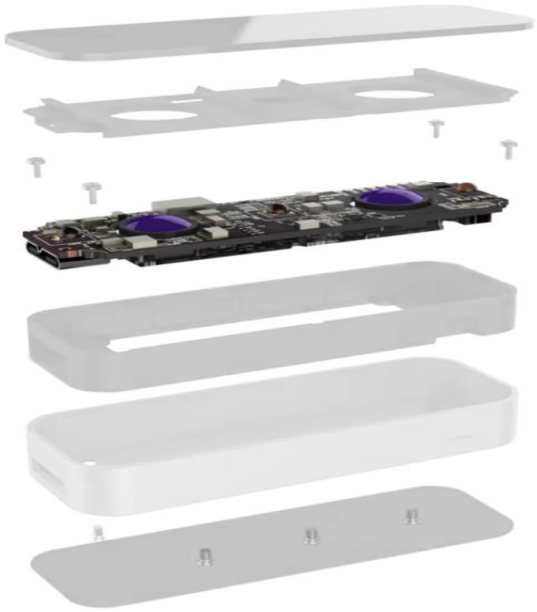


Figure 2.2: Leap Motion Internal Components, in colors, is the electronic component (infrared cameras, LEDs, and other parts), instead of grey the casing parts.

The specifications of the hardware indicate a wide hand tracking interaction zone, due to the wide-angle lenses. The interaction zone of the LMC device has an extension from 10 cm to 60 cm or more, with a 140°x120° typical field of view. Figure 2.3 and Figure 2.4 show the interaction zone described by the LMC device. Its framerate is high, approximately 120 Hz. Therefore, the interaction zone of LMC has a form of an inverted pyramid, the intersection of the binocular cameras' fields of view defines this pyramid shape. The dimension of the field of view depends on the propagation of the LEDs' light waves through space since the propagation becomes as harder as the distance increases. If the hand is too far hand detection is unreliable due to interferences.

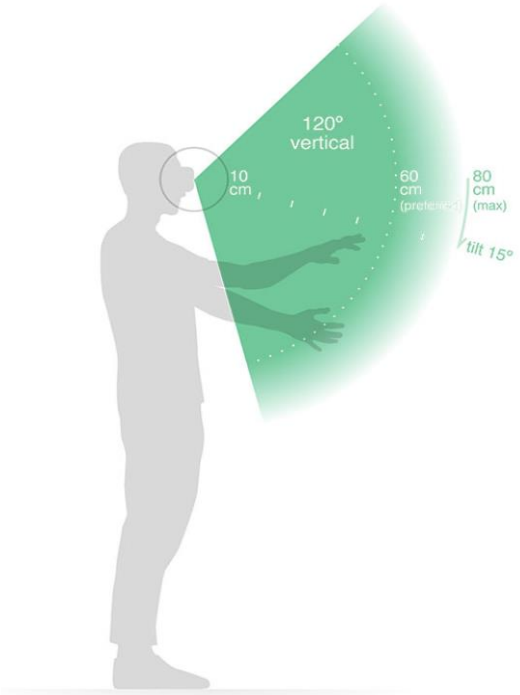


Figure 2.4: Interaction zone of Leap Motion Controller, lateral view

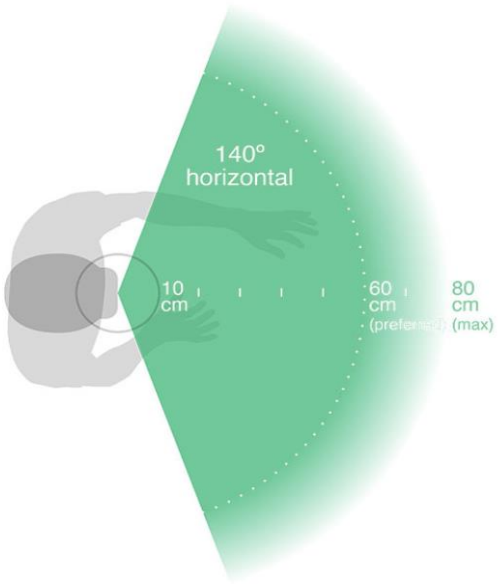


Figure 2.3: Interaction zone of Leap Motion Controller, above view

2.2.2 Software

When the image data is acquired from the device, it is streamed to the connected computer, and some computations are necessary. The algorithms are applied directly to the raw sensor data because the hand tracking platform does not generate a depth map

[8]. For the computing, it is necessary to download the Leap Motion Service, whose software has the aim to process the images coming from the LMC. After counterbalancing the background objects (such as heads) and the lighting around, the images are analyzed to build a 3D representation. Subsequently, the tracking layer matches the extracted data to track the information, for instance, the fingers, and palms movements. Then, the algorithms interpret the 3D representations and infer the positions of occluded objects.

2.3 Leap Motion Application Review

One of the first studies on the LMC was developed by Weichert *et., all* [10]. Their research is based about the analysis of the accuracy of this device, which until their study was not determined. The aim of their study was focused on the evaluation of the accuracy and repeatability of the LMC data. Using an appropriate evaluation, with the exploiting of an industrial robot, they obtained a deviation between a desired 3D position and the average measured position below 0.2 mm for static setups, instead for dynamic ones the value increases to 1.2 mm. But according to their conclusion, known these deviation values can be improved the development of applications of LMC in the interaction human-computer. Moreover, there was a study in 2016, from which has been making the validation for the LMC [11]. The study was based on the evaluation of the kinematic data output from LMC, and to validate this output was used the gold standard procedure, marketed motion capture technology. Their conclusion expresses that the LMC is capable of providing data for wrist flexion/extension with clinically meaningful, and it is not able to give data, with clinically meaningful, for forearm pronation/supination. And recommend continuing to validate the LMC as updated versions of their software are developed.

Furthermore, the LMC has a great field of applicability, i.e., from videogames [4] to rehabilitation procedures. Some studies of different application of this device are regarding the assessment of Parkinson's disease motor symptoms of bradykinesia and tremor [9, 12, 13, 14]. There is two intraoperative applications of the LMC during deep brain stimulation, to assess the motor symptoms during this surgery [9, 12].

For rehabilitation use, there are also different studies, as to facilitate the palm and finger rehabilitation [15], or to assess hand rehabilitation, with the monitoring exercises, for post-stroke patients [4, 16]

One of the main researching fields, where is used LMC, is gesture recognition due to its accuracy [17, 18, 19, 20, 21, 22, 23]. Other applications are the reconstruction of the finger motion [24], and the recognition of sign language [25, 26, 27].

Moreover, there are two interesting studies in which the LMC is used, one has a finalized aim towards the operating room, and the other one to assess the surgical dexterity among users with different experience levels. The first study concluded with the development and enhanced contactless interfaces with gesture recognition to improve the operating room control systems [28], and the second one, instead, allow trainees to evaluate their performance based on a reference model [29].

Chapter 3

3 Materials and Methods

For the aim of this research, the LMC and the programming environment MATLAB® are employed. The previously introduced LMC was used to acquire the data, which is then managed to extract useful information. Instead, the algorithms are developed in MATLAB®. As the LMC is an external device, it is necessary to use an interface algorithm between LMC and MATLAB® (called *Matleap*) [30].

3.1 Methodology

Using the interface *Matleap* to process the data in MATLAB®, it is necessary to understand the types and the utility of the outputs of LMC. On the official site of Leap Motion Controller (Ultraleap), there is a developer section, which details in depth the documentation of the LMC. Once the LMC outputs have been understood, several scripts to process the data and obtain useful features can be developed.

The LMC data are computed for each frame, and they are managed in structures named "*Hands*", moreover, these structures are composed of other structures, which describe some characteristics of the specific acquired hand (for instance, the visible time of the hand). Table 1 shows an example of the "*Hands*" structure. As we can see from this table, there are many interesting parameters to be used as features, but they are not enough for the purpose of this thesis, so during this work additional features have been defined. The last three components in Table 1 are other structures that present additional information on the palm, digits, and arm (for instance, their spatial coordinates, and orientation). Tables 2 and 3 show how the additional information about the palm, and arm are defined. Instead, the additional information about the digits is analogous to Table 2. The description of each parameter is acquired from the Ultraleap developer's official site [31].

Table 1. Hand's structure parameters and their description

Parameter	Description
ID	Unique ID for hand tracked across frames. If the tracking of a physical hand is lost, a new ID is assigned when tracking is reacquired.
Type	Identifies the chirality of this hand.
Confidence	How confident we are with a given hand pose. The parameter is still undefined in the current version. .
Visible_time	The total amount of time this hand has been tracked, in microseconds.
Pinch Distance	The distance between index finger and thumb.
Grab Angle	The average angle of fingers to palm.
Pinch Strength	The normalized estimate of the pinch pose.
Grab Strength	The normalized estimate of the grab-hand poses.
Palm	Additional information associated with the palm.
Digits	Additional information associated with the fingers.
Arm	The arm to which this hand is attached.

Table 2. Additional information associated with the Arm and each Finger

Parameter	Description
Previous Joint	The base of the bone, closer to the heart (The bones origin).
Next Joint	The end of the bone, further to the heart.
Width	The average width of the flesh around the bone in millimeters.
Rotation	Rotation in world space from the forward direction.

Table 3. Additional information associated with the Palm

Parameter	Description
Position	The center position of the palm in millimeters from the Ultraleap tracking camera device origin.
Stabilized Position	The time-filtered and stabilized position of the palm. Smoothing and stabilization are performed in order to make this value more suitable for interaction with 2D content.
Velocity	The rate of change of the palm position in millimeters per second.
Normal	The normal vector to the palm. If the hand is flat, this vector will point downward, or "out" of the front surface of your palm.
Width	The estimated width of the palm when the hand is in a flat position.
Direction	The unit direction vector aims from the palm position toward the fingers.
Orientation	The quaternion represents the palm's orientation corresponding to the basis

3.1.1 Hand Reconstruction

Once the meaning of the LMC outputs is clear, the studied hand has been visualized in a 3D representation in MATLAB®. To reconstruct the hand following the positions of the arm, palm and digits were calculated, taking in mind the anatomical representation of the hand. Figure 3.1 represents the palm reconstruction, in which the points in space of the digits are linked with lines to approximate the volume dimension

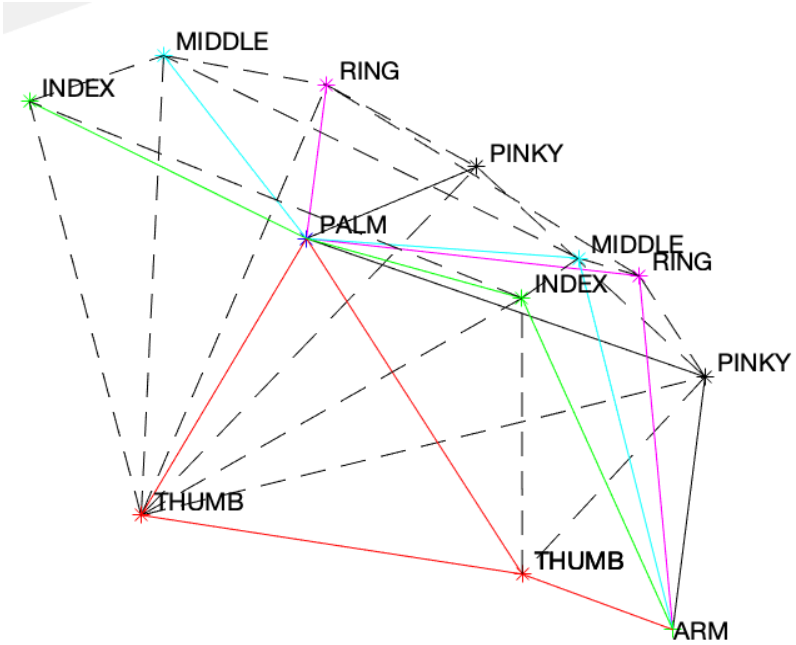


Figure 3.1: left view of hand 3D representation in MATLAB®.

of the hand in MATLAB®. Therefore, having the 3D graph of the palm to obtain the relative hand whole representation misses just the other digits' points, as shown in Figure 3.2, in which are represented Thumb in red, Index in green, Middle in light blue, Ring in purple, and Pinky in black

From Figure 3.2 one can see some feature signals useful for our study, for instance, the distance between each digit joint and the center of the palm. The main property of this type of features is that, independently of the reference system, the distances do not change. So, we focused our feature signals on the extraction of relative distances between hand joints. Moreover, for sake of simplicity, the center of the plot coincides with the center of the palm.

The entire script used to graph Figure 3.1 and Figure 3.2 is present in Appendix 1.

- function_plot (*Hand*_struct, start, frame, hold_on_Y_N)

This function plots the acquired frames one by one, and it has three inputs:

1. *Hand*_struct ⇒ the struct that contains the hand data

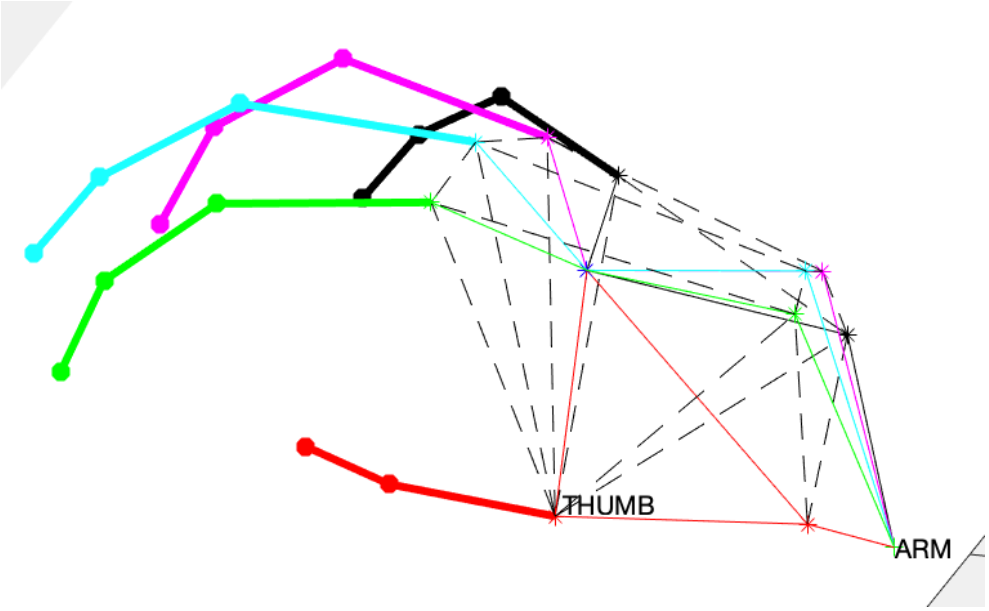


Figure 3.2: Left view of hand 3D representation in MATLAB ®.

2. Start ⇒ the initial frame to start plotting
3. Frame ⇒ The final frame
4. Hold_on_Y_N ⇒The choice of plotting different frames in the same figure or one frame in one figure.

As output, we obtain the desired time graph of the acquired hand data in the specific interval of frames, its entire code is reported in Appendix 2.

3.1.2 Extracted Hand Signals

As already stated, we decided to extract other signals from the already acquired data by LMC. For simplicity, we express these extracted signals as "*Hand*" signals, furthermore, since they are time-dependent signals, a signal-based analysis can be performed for the extraction of the features.

These "*Hand*" signals are divided into two groups:

- Distances

- Angles

For each group some scripts have been written, firstly, to define the signals, and secondly to extract the features from each "*Hand*" signal.

For the Distances group:

we extracted two "*Hand*" signal types. In brief, one is the hand grab, and the other is the distances between the thumb joints and the joints of the other digits. The entire functions used to obtain these types of "Hand Signals" are reported in the Appendix 3.

One function is:

[grab_Thumb, grab_Index, grab_Middle, grab_Ring, grab_Pinky] =

- function_palm_digits (*Hand*_struct, frame)

The above function has two inputs:

1. *Hand*_struct ⇒"*Hand*" struct where the data is stored
2. Frame ⇒ The desired end frame,

and five outputs. Each output has four values, as a column vector, computed with the following methods:

- i-*th* phalanx coordinates – palm coordinates,

but the exactly i-*th* phalanx coordinates depend on the management of the "*Hand*" structure.

Thus, from this function are obtained five different outputs, and for each finger there are four hand signals, as the number of bones joints in the finger except for the thumb which has three of them. Figure 3.3 shows only the distances between the last joints of each finger and the center of palm.

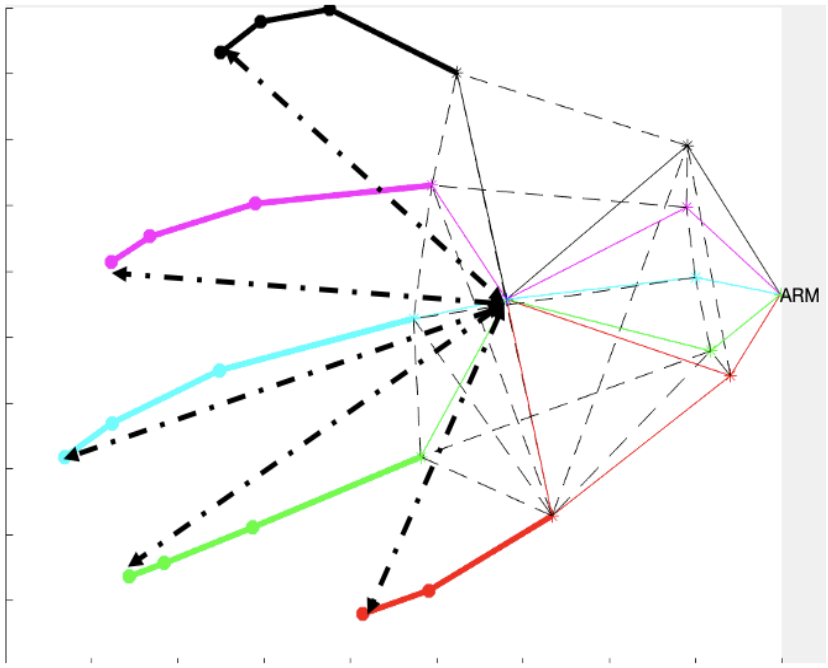


Figure 3.3: Example of the distance computed with function_palm_digits

The other used function is:

[Thumb_Index, Thumb_Middle, Thumb_Ring, Thumb_Pinky] =

- function_thumb_other_digits (*Hand*_struct, frame)

- This function has the same inputs explained in the previous function. Instead, the outputs represent the distance between each thumb joint and each other digit joint. Like the previous function these outputs are vectors, but in this case, they are composed of three Distances

Angles values. For the computation of these values, the floating finger joints and their fingertips have been used. The exact procedure of computing is not described since depends on the management of the "*Hand*" structure data. Figure 3.4 shows an example

of which thumb joints are taken to compute the distances, and which digit joints are taken. Figure 3.4 shows just the Index but the same occurs for the other fingers

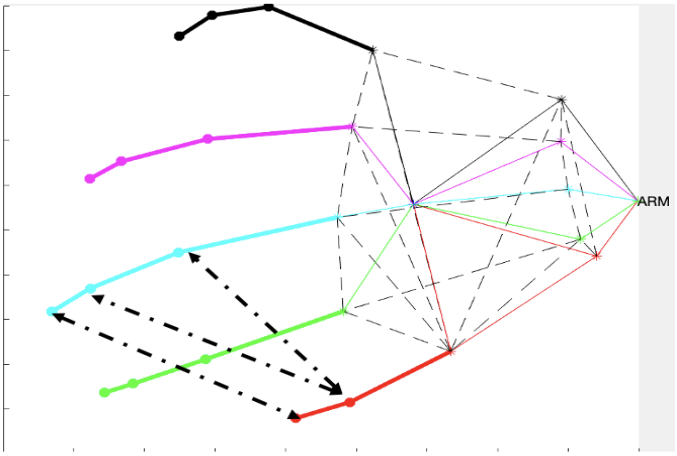


Figure 3.4 Example of the Thumb_Index outputs computed with function_thumb_other_digits

As for the Angles group of features, the traditional "Yaw, Pitch, and Roll" parametrization has been chosen. The orientation of the reference frame is shown in Figure 3.5.

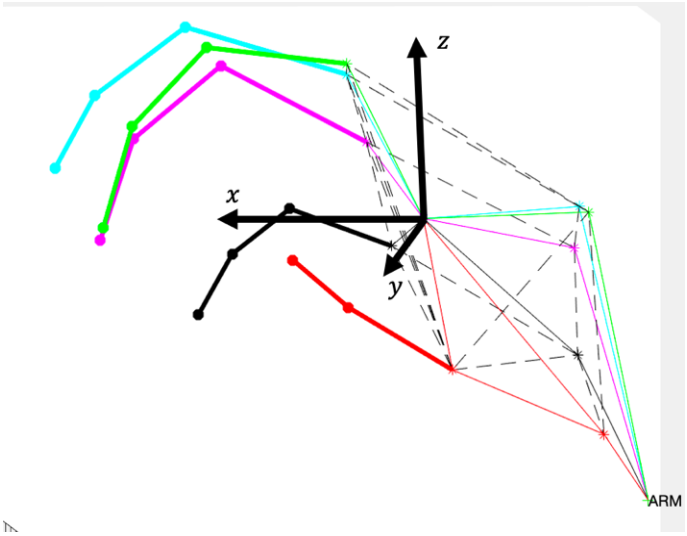


Figure 3.5: Reference Frame applied in the palm reference

Broadly speaking, and referring to the reference frame applied in Figure 3.5, the "Pitch" is related to rotation around the y axis, the "Yaw" around the z axis, and the "roll" around the x axis. From the "*Hand*" structures, the relative *orientation* is computed in form of

quaternions, of the palm, the arm, and each digit component, but this information has not been used, as it is redundant and poorly documented. Thus, the yaw and pitch angles are calculated according to the following equation:

$$\theta = \cos^{-1}\left(\frac{(\vec{a}\cdot\vec{b})}{|\vec{a}||\vec{b}|}\right) \quad (1)$$

where:

- $\theta \Rightarrow$ angle in the joint finger,
- $a \Rightarrow$ previous finger bone,
- $b \Rightarrow$ following finger bone.

Figure 3.6 shows graphically the meaning of the element in the above equation. In other words, the above equation computes the angle described by the dot product between $\vec{a}$ and $\vec{b}$.

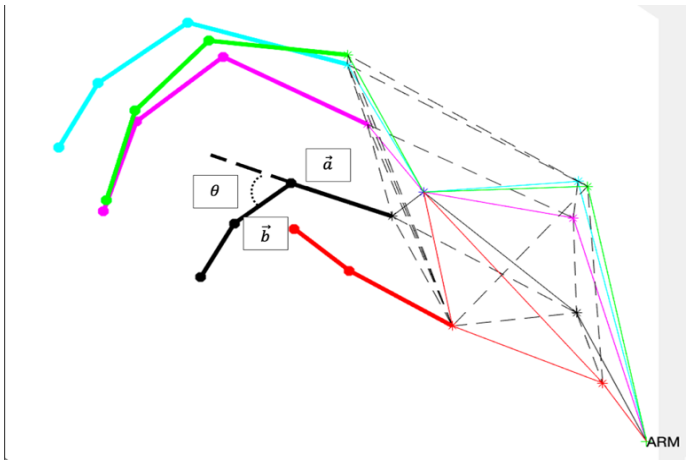


Figure 3.6: Example of taken segments to compute the angle.

From Figure 3.5 it is possible to see that angle describes the relative pitch angle between two consecutive bones of the same finger. This procedure is applied to each couple of consecutive bones in the fingers. The entire code to compute the "Pitch" angles is reported in the Appendix 4, while in the following the inputs and outputs of this function are reported:

- [pitch] = pitch_digits (*Hand*_struct, frame)

the inputs are the same used for the previous functions, i.e., *Hand*_struct and frame, necessary to know where the data is stored (*Hand*_struct) and when (frame). The output, instead, is a structure that contains the angles information of pitch for each finger bone and finger.

Eq. (1) to compute the θ angle was useful to elaborate the ϕ "Yaw" angle between two consecutive fingers. Figure 3.8 shows the segments used to compute the angles, which are not defined as $\vec{a}$ or $\vec{b}$ because the values of the angles do not change for this specific configuration.

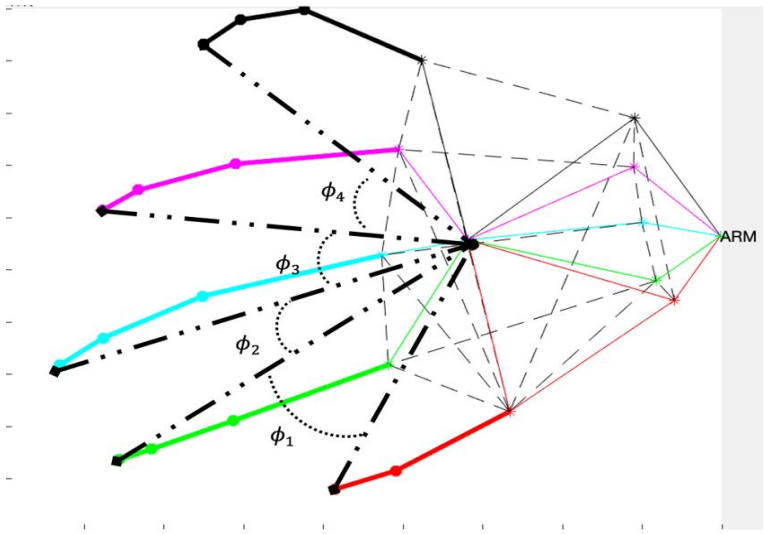


Figure 3.7: Shows how the "Yaw" angles are computed.

The function used to obtain these angles is like the pitch function:

- [yaw] = yaw_digit (*Hand*_struct , frame)

The inputs are the same as the previous ones, only the output is different. Also, the output is a structure in which there are the "Yaw" angles computed in the defined frame. The entire code is reported in the Appendix 5.

Table 4 summarizes the above-described functions, the numbers of their extracted hand signals, and their dimension unit. As shown in Table 4, there are a high number of extracted hand signals. Therefore, it is possible to analyze only one signal for each finger elaborated by each function. However, this signal must be the best of the other ones, and we followed this procedure. In other words, we restrict our attention to the best signal according to its sensitivity, i.e., its total displacement.

Table 4. Functions summary

Function	Number of hand signals	Total
function_palm_digits [mm]	5 signals for each of the 4 joints of the fingers	20
function_thumb_other_digits [mm]	4 signals for each the 3 bones of thumb	12
pitch_digits [deg]	3 signals for each the 3 bones of the fingers	15
yaw_digit [deg]	1 signal for each finger	4

3.1.3 Filtering

To enhance the extraction of features from these hand signals, they must be filtered to remove the sensor noise and any tremble, and to design the filter, the MATLAB function "*butter*" is employed. This function generates a Butterworth filter with the desired specifications in the input. Moreover, the function "*filtfilt*" was also applied to filter each hand signal twice with the designed Butterworth filter, once in the forward and once in the backward direction. This procedure allows to decrease the filter order that specified as an input in the Butterworth function. As a drawback, "*filtfilt*" can be employed only at the end of the signal acquisition. Thus, the specifics of our filter are**:**

- Order: 2

- Cut-off frequency: 6 Hz
- Filter type: Low-pass filter

Also, the sampling frequency of the signals must be defined: it can be set to 120 Hz, as the output frequency of the LMC which is consistent with the documentation. The code is reported in Appendix 6.

3.1.4 Features Extraction and Evaluation

Given the "hand" signals previously computed, the following step is to define which features can be extracted from these signals. During most of the tasks, some repetitive patterns occur. For sake of simplicity only time-depending features have been chosen, in other words, we have taken just features in the time domain.

The chosen time-domain features are:

- Period [s]
- Amplitude [mm or deg]

The Period feature has been thought as the time distance between two repetitions of the same task. Instead, the Amplitude feature is the total displacement, in mm or deg, between the maximum and minimum values inside each period.

Once the features have been defined, their evaluation is an open problem. Inspired by the training performed by the neuropsychologist before the surgery, a possible way is to perform and evaluate some pre-surgery acquisitions (the number depends on how many tasks the patient would do). These acquisitions give us the necessary information to apply thresholds inside a "task control algorithm". According to the task, the "task control algorithm" warns the neuropsychiatrist if the movement is good or not, i.e., how much is the percent difference between the task executed in pre-surgery and during the surgery. A negligible loss means that the area is safe, and the neurosurgeon can cut the previously excited brain zone.

3.1.5 Velocity and Acceleration features

As a possible additional feature, the velocity and acceleration graphs relative to the "hand" signals have been computed. The elaboration of these derivatives has been done using the definition of the rate of change of a discrete function:

$$\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{F}(\mathrm{t}) = \frac{\mathbf{F}(\mathrm{t} + t_s) - \mathbf{F}(\mathrm{t})}{t_s} \tag{2}$$

where:

- **F** is the function that we want to differentiate
- t_s is the sampling period (in this case t_s= 1/120 s)

Applying this equation, we obtain the velocities when **F** represents a distance or an angle, and we obtain the accelerations when **F** is a velocity function of distance or angle.

3.1.6 Maximum and Minimum Computation

In order to define both the period and the amplitude of the task, locating the minima and maxima of each feature is crucial. Such evaluation must be robust with respect to small ripples caused by noise and flicker, which cannot be totally eliminated by filtering. This calculation is applied both for distances and angles signals, starting from the first ones. In MATLAB there are many functions that can compute these points, but to obtain the desired points precisely it is necessary to set some conditions. To avoid errors in searching maximum/minimum points, caused by the non-smoothness of the inputs, we have defined a customized function.

The proposed function, reported in the Appendix 7, is based on the definitions of maximum and minimum points (i.e., when the first derivative is equal to zero). Its definition is the following:

- [Area, derivative] = Computation_area (*Hand*_Signals)

As input is a structure, in which are present all the *hand* Signals that have been previously described. The outputs are the derivatives, which are computed inside this function, and the values of Areas. With the term "Values of the Areas", we mean the computation of the area between the derivative function and the x-axis of the function. Thus, knowing the areas we can define if the points where the derivative is zero are maximum or minimum (whether the area is positive means that the following zero point is a maximum, or if the area is negative means that the following zero point is a minimum).

Now, we know the maximum and minimum points in each signal. Then, is necessary to determine a way to get the period and amplitude features from these points.

3.1.7 Determination of Max and Min Points to Extract the Features

The computation of the Max and Min points is the hardest step because requires to create an algorithm that is capable of separating the features, previously described, and other points which, for sake of simplicity, we called them "particulars". The points of Max and Min used to extract the features must be, properly, the max and min displacement of each finger during the task, and obtained them, we are able to compute the period and amplitude features. Instead, the "particulars" are the points inside the range of the used points to elaborate the features.

Therefore, independently all points are important, some to extract the features (then used to determine the thresholds), and some others to define if inside the task there are warnings, like freezing of the hand, or a slowdown of the movement.

To identify those points, two algorithms are reported in the Appendix 5. The first code, easier than the second one, determines the desired points according to the values of

the areas, in absolute value, i.e., it takes only the points preceded by an absolute area value bigger than the average of the absolute all areas inside the function. Instead, the second code computes the linear interpolation between every two points, which are determined from the output of the *Computation_area* function. The linear interpolation is characterized by the following formula:

$$y = m\,x + q \tag{3}$$

Therefore, we compute the m and q values to find the passing line through every two points. Once got the m and q values, corresponding to the line which link two points at time, we exploit their value to know how many is the changing on amplitude depending on the slope value of the interpolating line. For to get the results we must impost a threshold on the slope, what was between -0.5 and 0.5. Therefore, due to this interval only the slope greater or lower than the threshold is got and processed to compute the max and min inside the signal. The code is reported in Appendix 8.

The Results will show the differences between these two codes as well.

3.2 Pre-Surgery Acquisitions

These types of acquisition are the most important because through the elaboration of them we can determine each patient-specific threshold and avoid errors due to the generality of the algorithm.

The protocol regarding which task the patient must execute depends on the neuropsychiatrist, this could change in every surgery.

3.3 The Chosen Tasks to apply the Codes

In order to verify the reliability of our codes, we have chosen two different tasks: this choice was made just to verify this code in the easiest task (open and close the hand), and in a little bit more complex task, namely the movement of the thumb towards the other fingers one by one.

Chapter 4

4 Results

previously described methodology. In detail, the reported graphs follow the order in of the functions which have been previously discussed. Being studied two different tasks, we decided to show in Section 4.2 the results of the “open and close” task, and in Section 4.3 the results of the thumb towards the other fingers “one by one” task.

The results of the *computation_area* are only regarding the filtered distance “hand” signal, and also to determine the necessary maximum and minimum points to acquire the desired feature. The angles plots are not detailed in deep because there is not a suitable criterion to evaluate them.

Moreover, the two codes written to get the desired points to extract the amplitude and the period features, are shown only for the open and close task, because the results of this code are not satisfying for the second studied task in this research.

4.1 by function_plot

The results of this function have been already shown in the previous Figure 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, and 3.7. Moreover, being just a picture of a single frame, it is not so useful to evaluate the task. Hence, we decided to build an animated sequence in which we can see the motion of the hand and its fingers during the task together with the grab of each finger graph. The code is reported in Appendix 9. Note that he insertion of the results of this upgraded code is not possible because, in other words, it is essentially composed of a sequence of different frames.

This function is independent of which task the patient does. Moreover, Sections 4.2 and 4.3 show the outputs of this function in the specific frames, depending on the task, to mark the importance of the 3D visualization.

4.2 Open and Close task

The below Figure 4.1 shows the two-characteristic steps of these task, namely completely open and close hand.

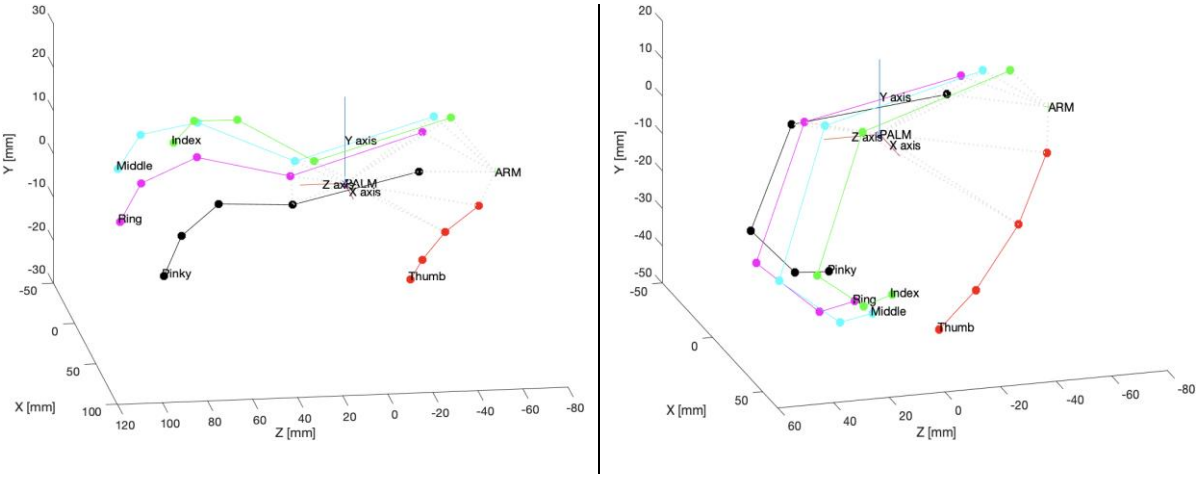


Figure 4.1: left image, hand in completely open; right image, hand complete close.

4.2.1 by function_palm_digits

The below Figures from 4.2 to 4.6 show four different graphs, which each corresponds to a joint of the finger, from the closer to far joint has the following colors: Blue, Red, Yellow, and Purple. Moreover, are shown both the unfiltered and filtered signals.

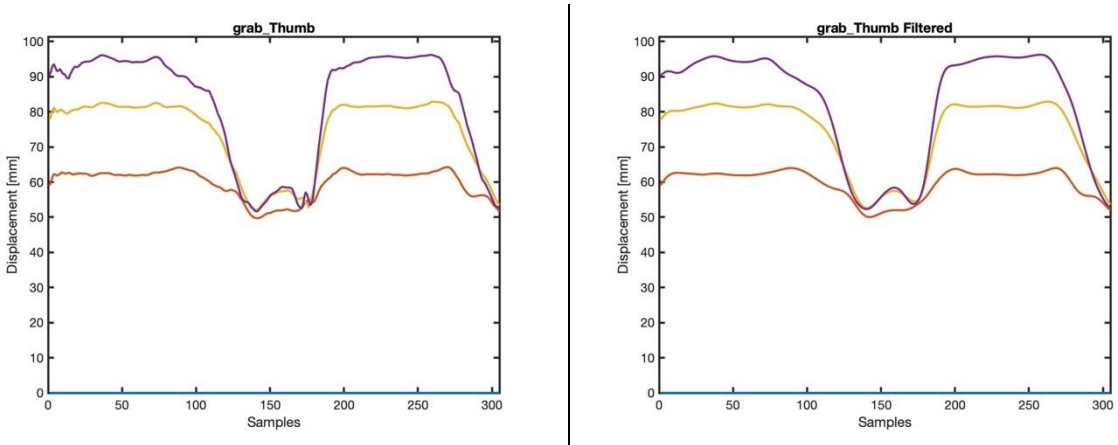
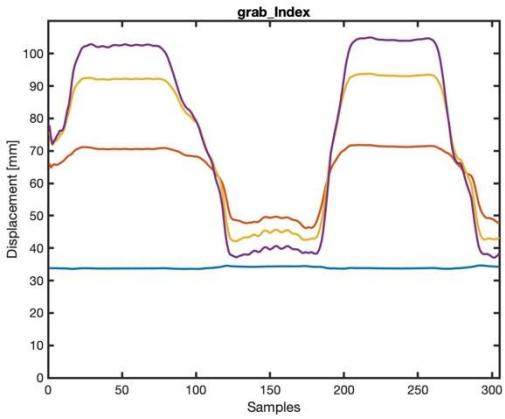


Figure 4.2: First task results of the Grab of Thumb



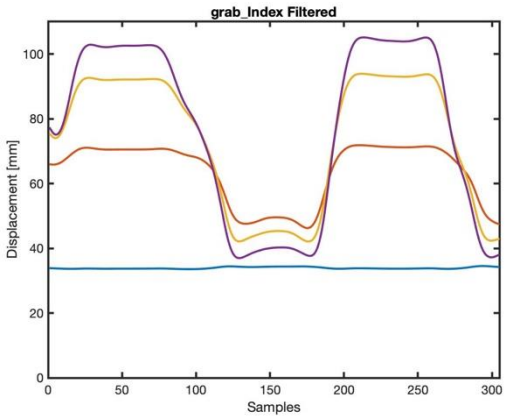
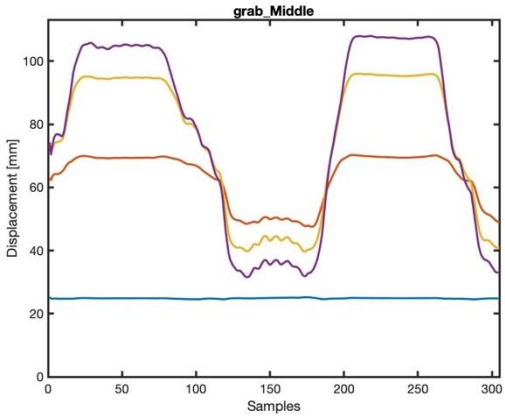


Figure 4.3: First task results of the Grab of Index



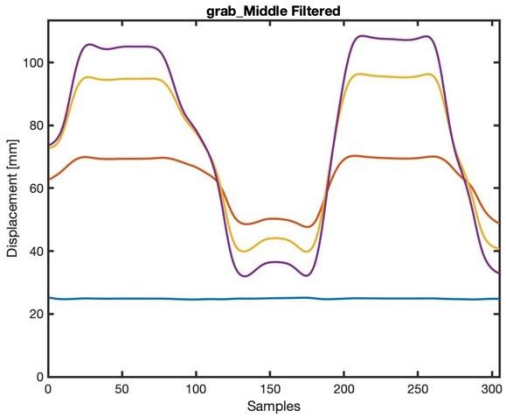
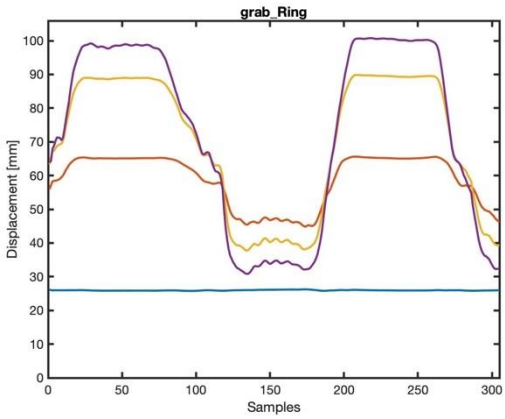


Figure 4.4: First task results of the Grab of Middle



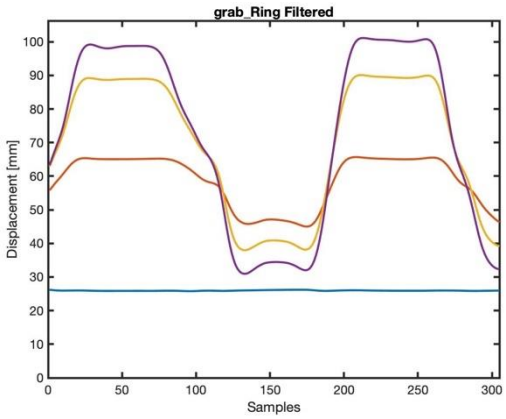


Figure 4.5: First task results of the Grab of Ring

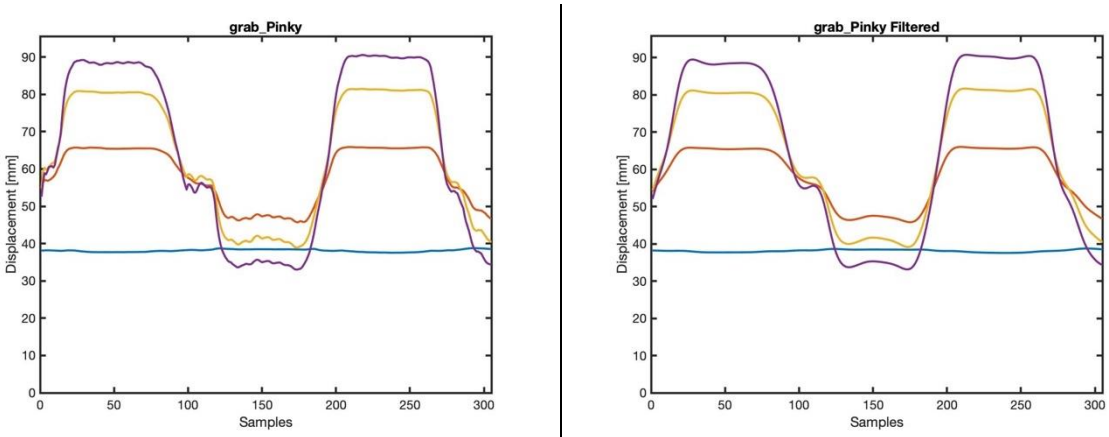


Figure 4.6: First task results of the Grab of Pinky

4.2.2 by function_thumb_other_digits

The below Figures from 4.7 to 4.10 are composed of three different curves, unfiltered and filtered. The meaning of each graph corresponds: in Yellow the distance between the tips of the thumb and the other fingers, in Blue and Orange, instead, are the distances between the thumb joint, immediately after the tip of the thumb, and the other two joints of each finger (as is shown in Figure 3.4).

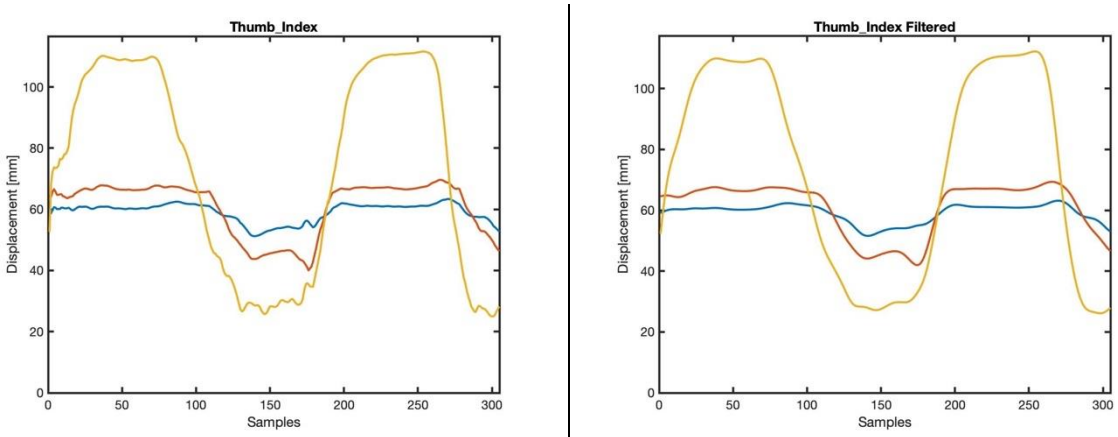


Figure 4.7: First task results of the Thumb towards Index

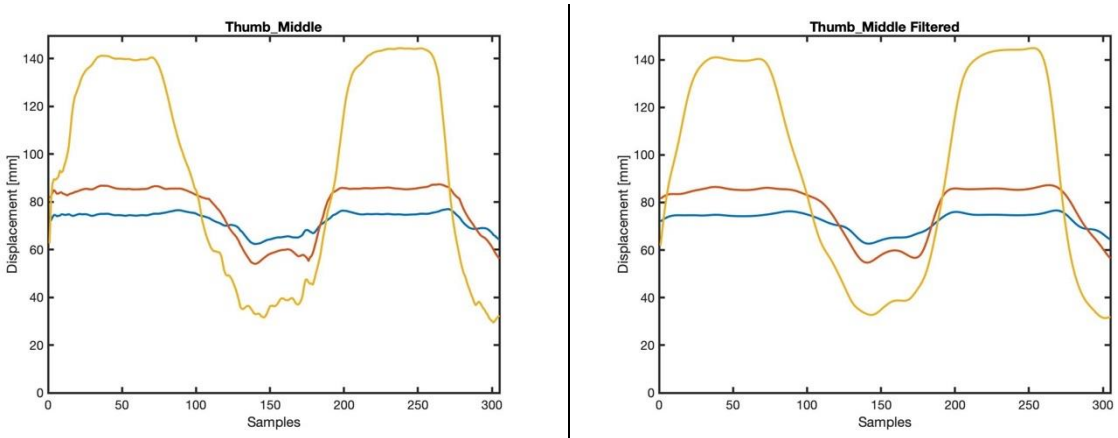


Figure 4.8: First task results of the Thumb towards Middle

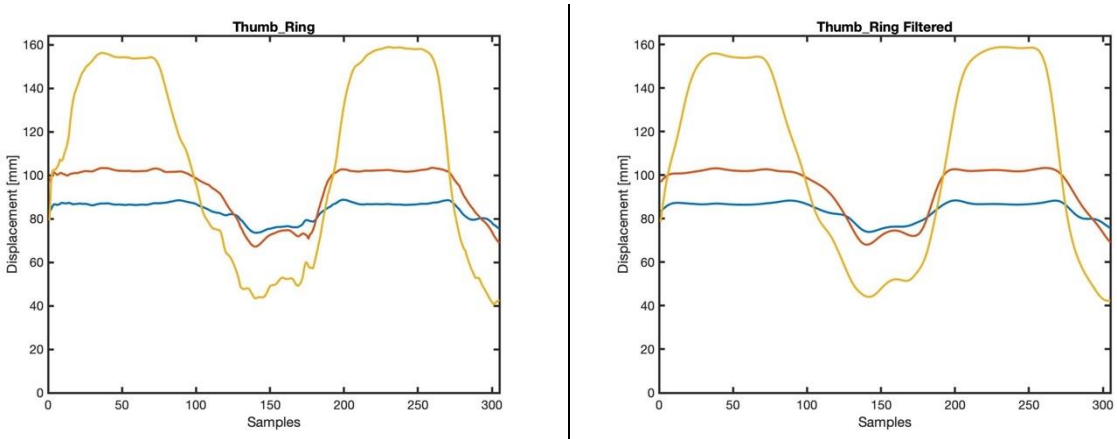


Figure 4.9: First task results of the Thumb towards Ring

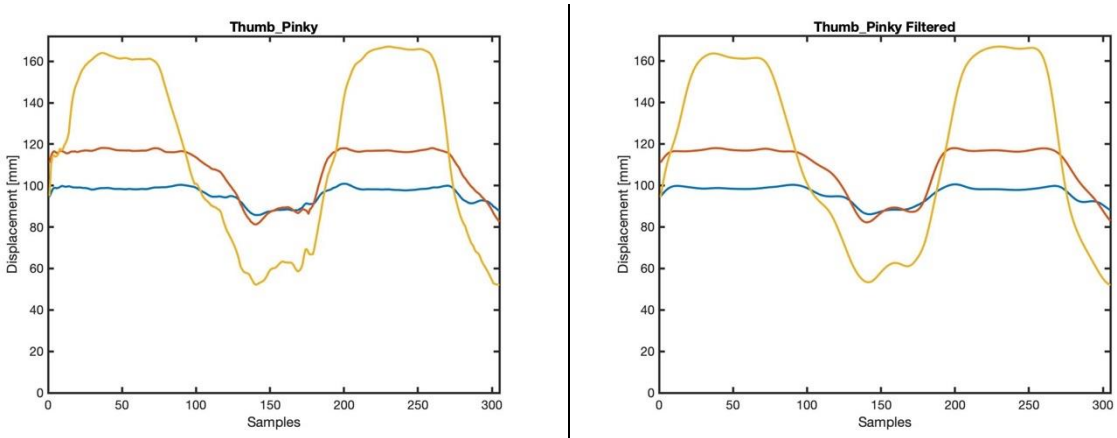


Figure 4.10: First task results of the Thumb towards Pinky

4.2.3 by pitch_digits

From Figure 4.11 to Figure 4.15, show the results of the pitch angles in each finger. The colors represent: in Blue, the angle between the first and second phalanx, in Orange, the angle between the second and third phalanx, and in Yellow, between the third and fourth phalanx. On the left is the unfiltered data and, on the right, the filtered one,

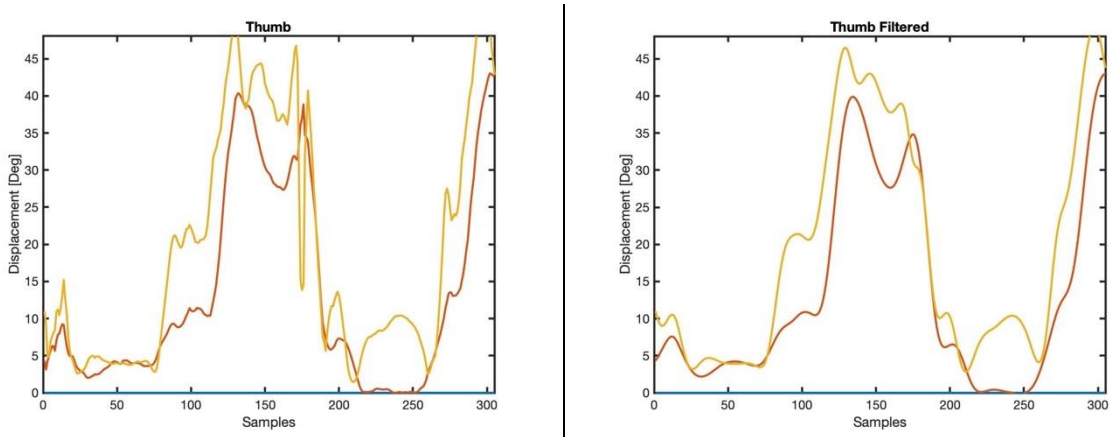


Figure 4.11: The Pitch angles in the Thumb in the first task

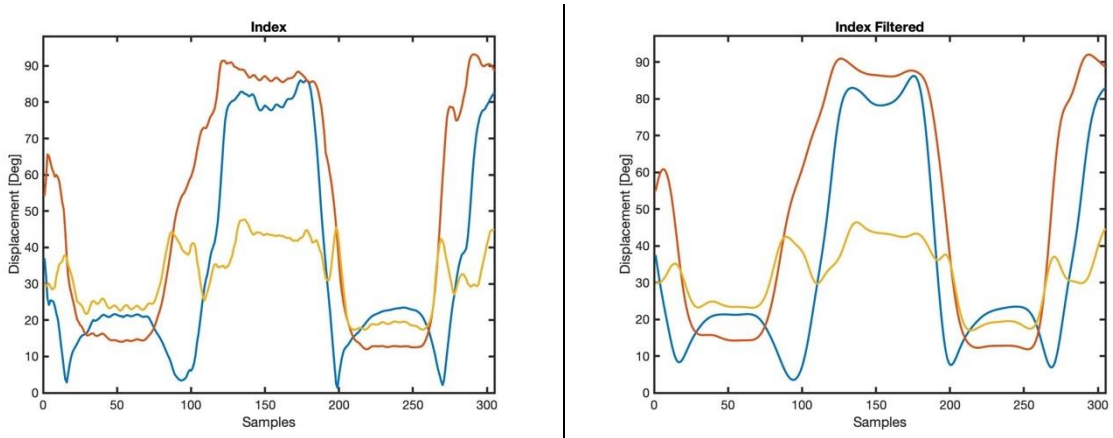
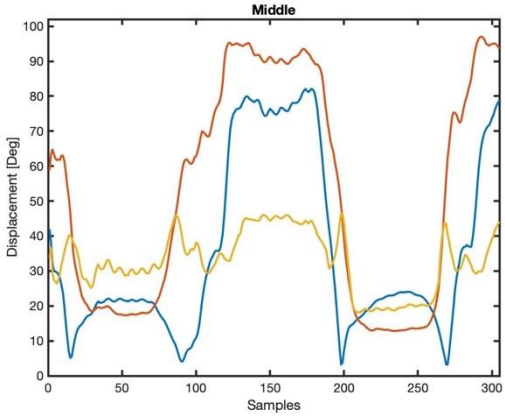


Figure 4.12: The Pitch angles in the Index in the first task



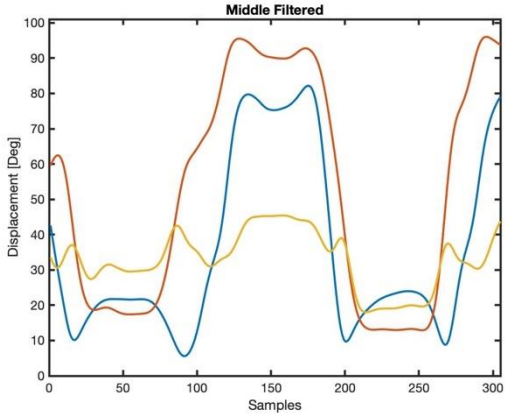
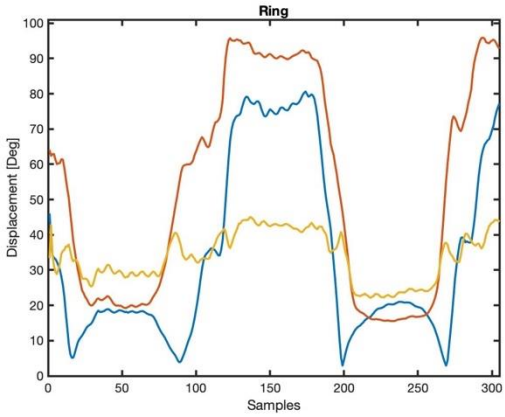


Figure 4.13: The Pitch angles in the Middle in the first task



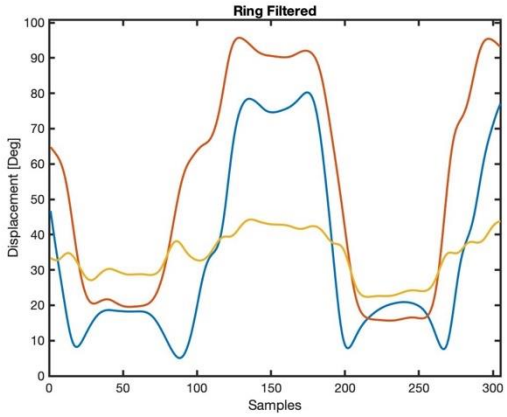
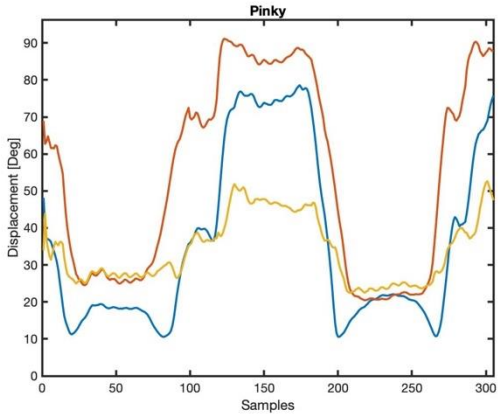


Figure 4.14: The Pitch angles in the Ring in the first task



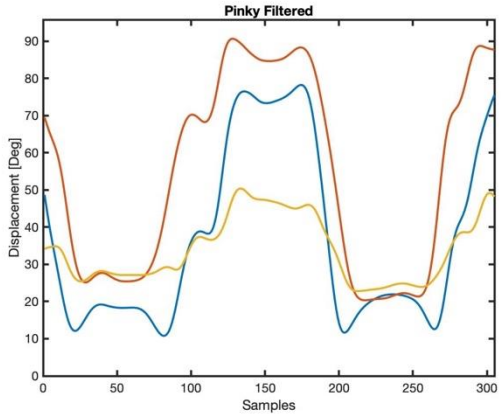


Figure 4.15: The Pitch angles in the Pinky in the first task

4.2.4 by yaw_digits

The resulting graphs from Figure 4.16 to 4.19 are the angle representation between two consecutive fingers. The data on left are unfiltered, instead on right are filtered.

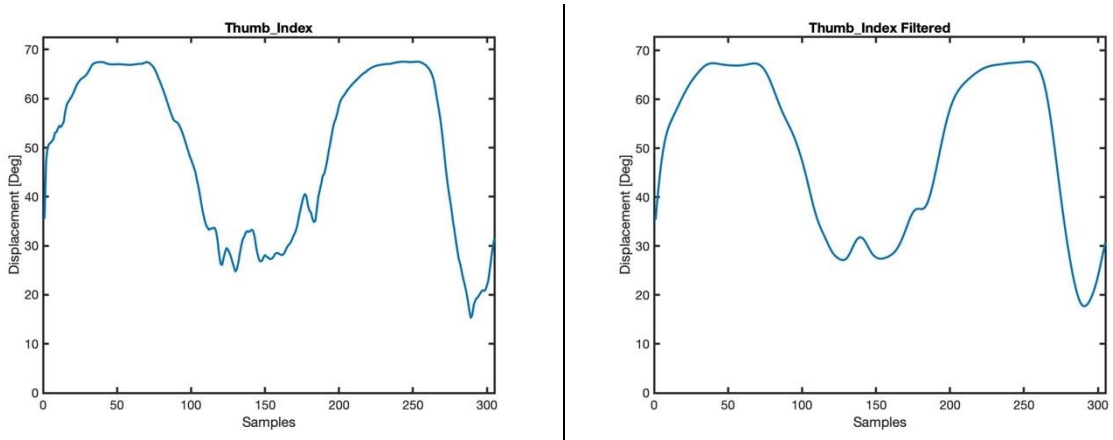


Figure 4.16: The Yaw angles between Thumb and Index in the first task

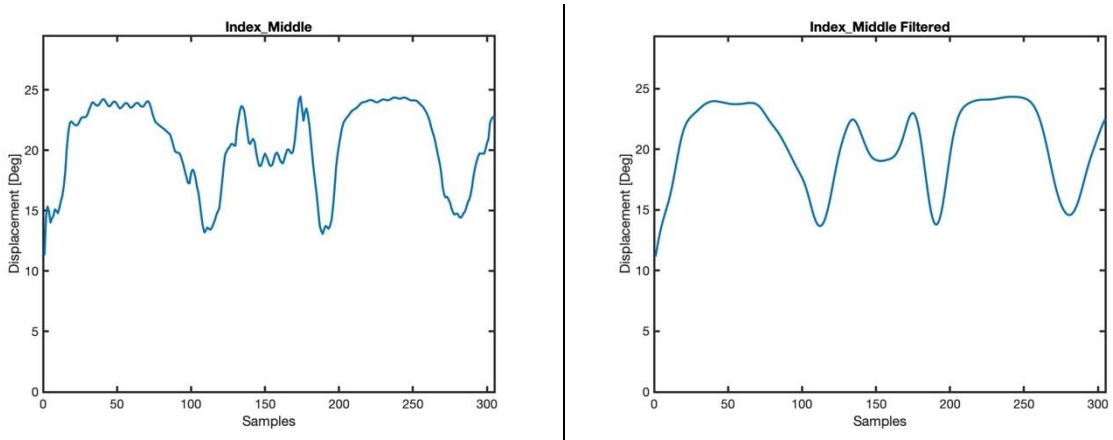


Figure 4.17: The Yaw angles between Index and Middle in the first task

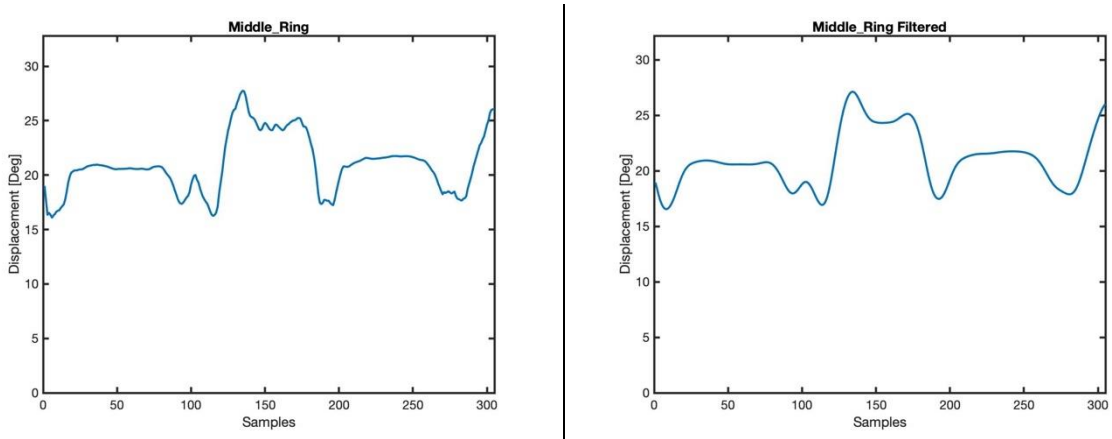


Figure 4.18: The Yaw angles between Middle and Ring in the first task

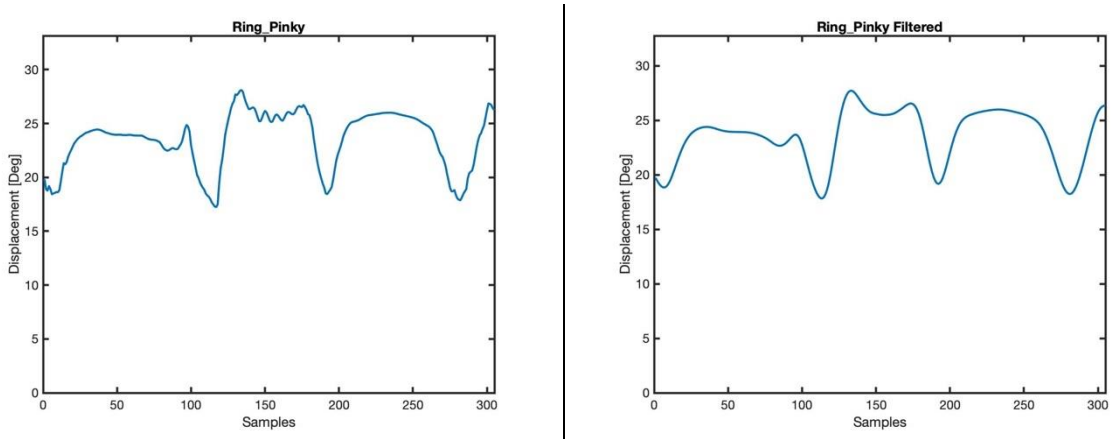


Figure 4.19: The Yaw angles between Ring and Pinky in the first task

4.2.5 by computation_area

The following Figures, from Figure 4.20 to 4.28, are shown the higher displacement graphs (the resulting graphs from Section 4.2.1 and 4.2.2), which have been multiplicated by a constant, because to visualize better those graphs, which explains why the red graphs have an amplitude completely different than their previous ones. In Yellow are shown their first derivative, and in black dots are the corresponding zero-crossing of their first derivative.

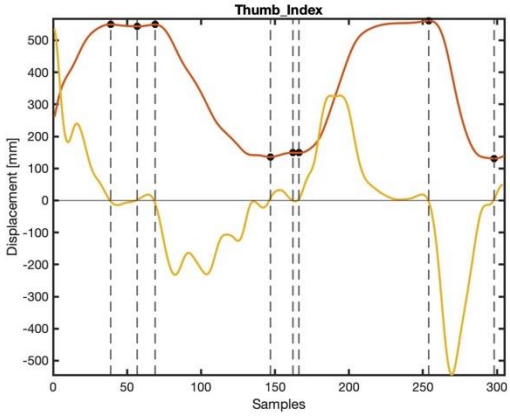


Figure 4.20: Zero-crossing points determination for the Thumb towards Index

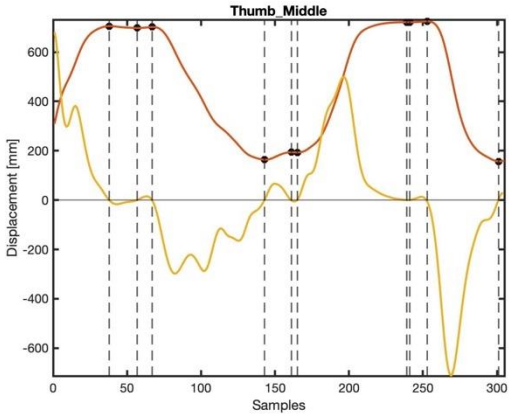


Figure 4.21: Zero-crossing points determination for the Thumb towards Middle

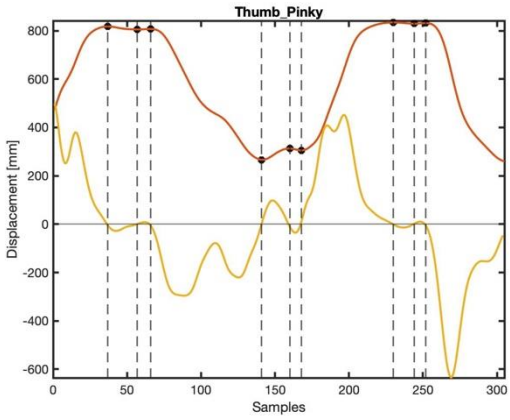


Figure 4.22: Zero-crossing points determination for the Thumb towards Ring

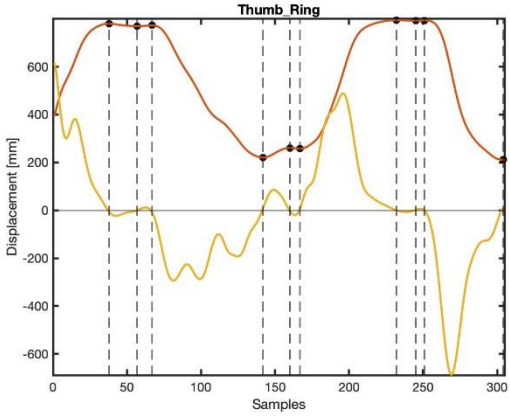


Figure 4.23: Zero-crossing points determination for the Thumb towards Pinky

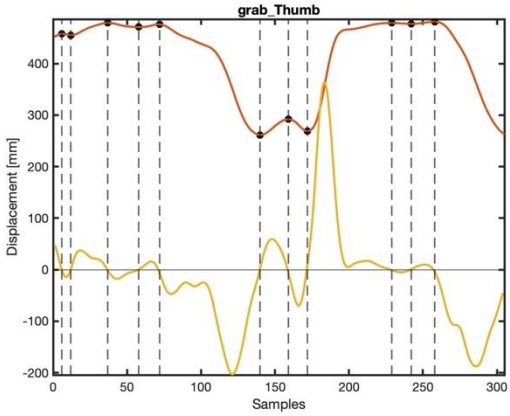


Figure 4.24: Zero-crossing points determination for the Grab of Thumb

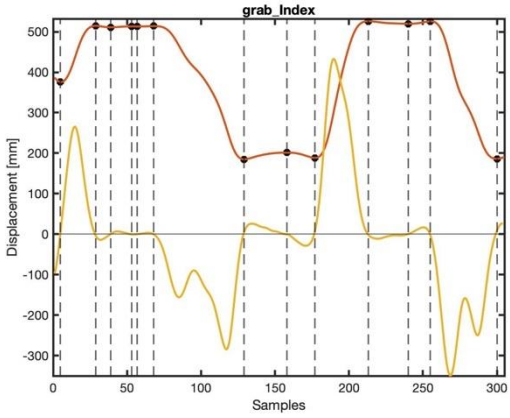


Figure 4.25: Zero-crossing points determination for the Grab of Index

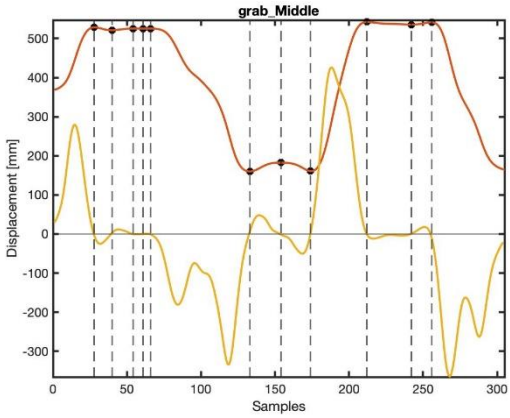


Figure 4.26: Zero-crossing points determination for the Grab of Middle

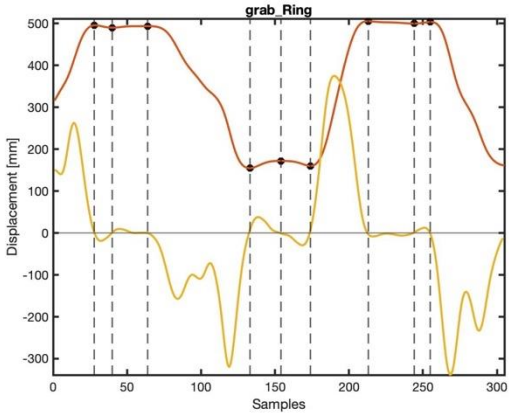


Figure 4.27: Zero-crossing points determination for the Grab of Ring

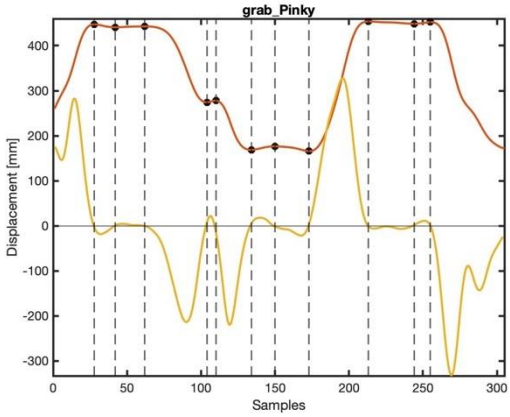


Figure 4.28: Zero-crossing points determination for the Grab of Pinky

4.2.6 by the functions to find the Max and Min points

The Figures from Figure 4.29 to 4.37 show on the left the outputs from the “area” code and on the right the outputs from the “slope” code, both described in Section 3.1.7. the curves inside each graph are the same as specified in the Section 4.2.5. Moreover, for the right graph is expressed in black dots the maximum points, and in red the minimum points, according to the “slope” algorithm.

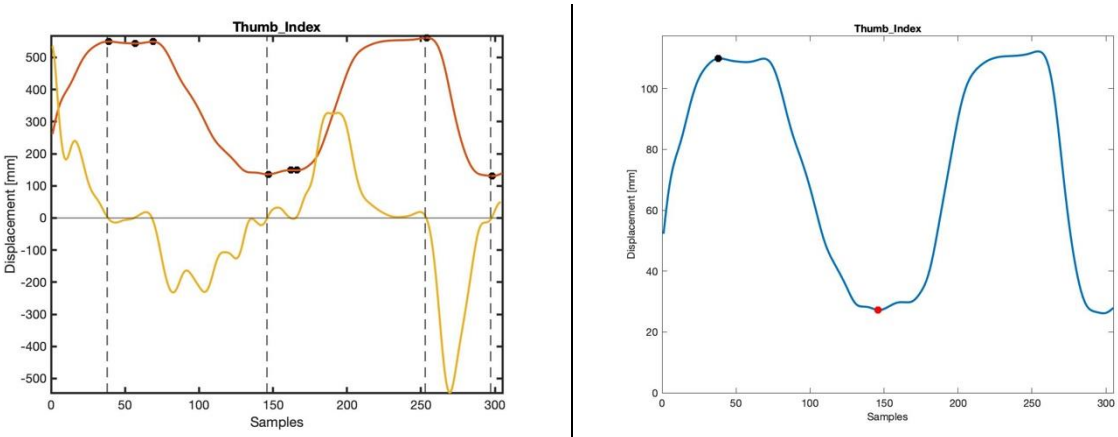


Figure 4.29: Maximum and Minimum points determination in the First task results of Thumb towards Index

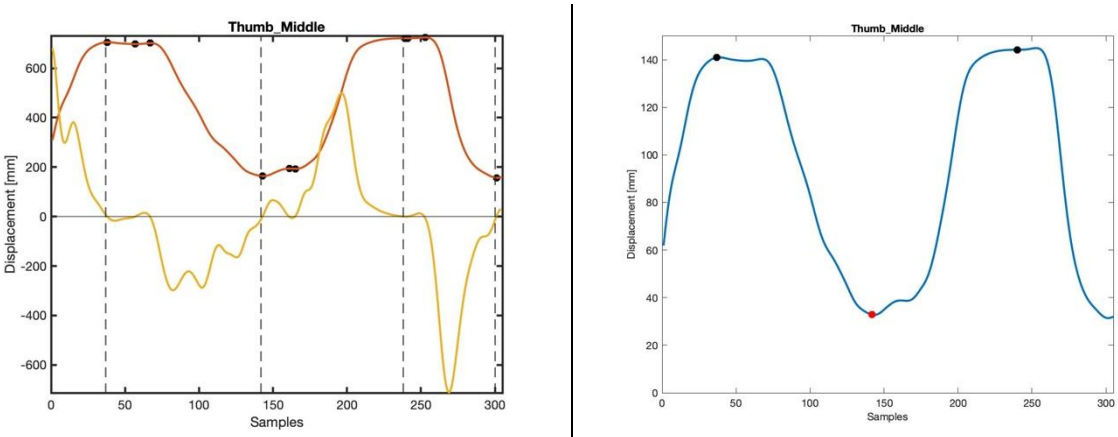


Figure 4.30: Maximum and Minimum points determination in the First task results of Thumb towards Middle

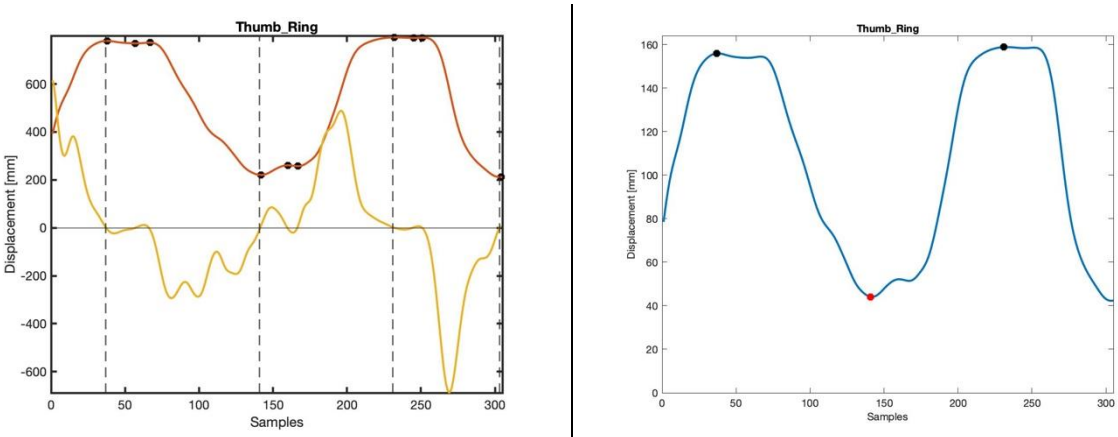


Figure 4.31: Maximum and Minimum points determination in the First task results of Thumb towards Ring

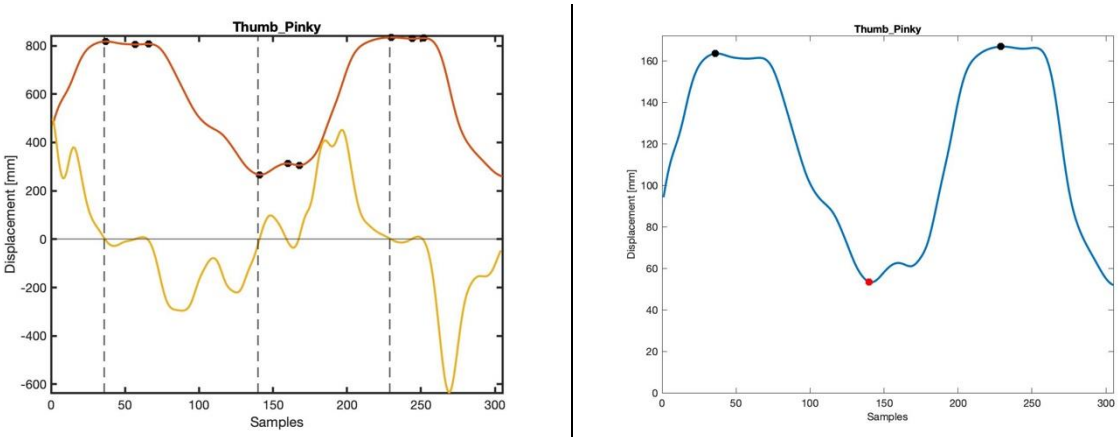


Figure 4.32: Maximum and Minimum points determination in the First task results of Thumb towards Pinky

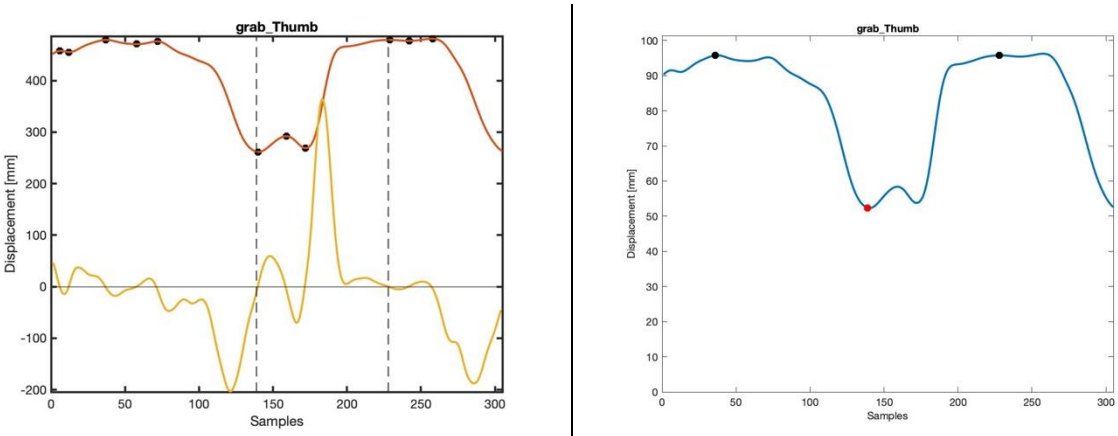


Figure 4.33: Maximum and Minimum points determination in the First task results of The Grab of Thumb

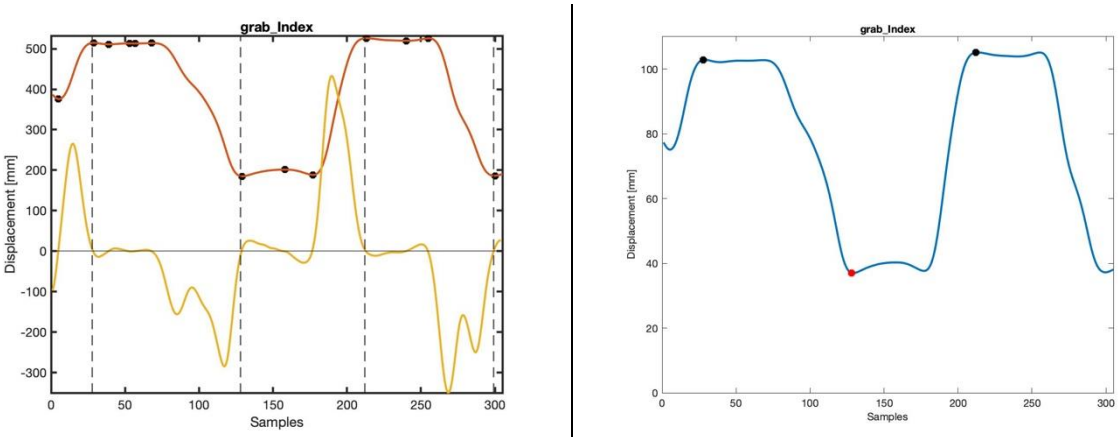


Figure 4.34: Maximum and Minimum points determination in the First task results of The Grab of Index

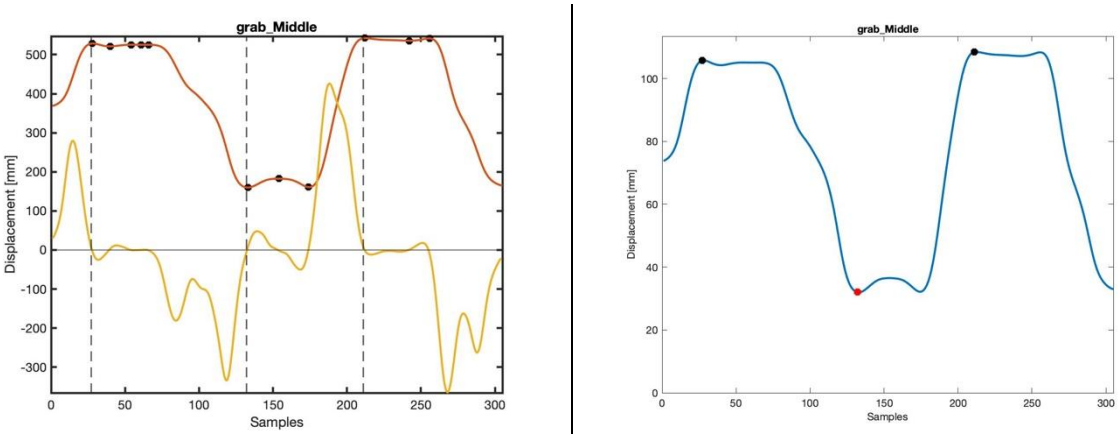


Figure 4.35: Maximum and Minimum points determination in the First task results of The Grab of Middle

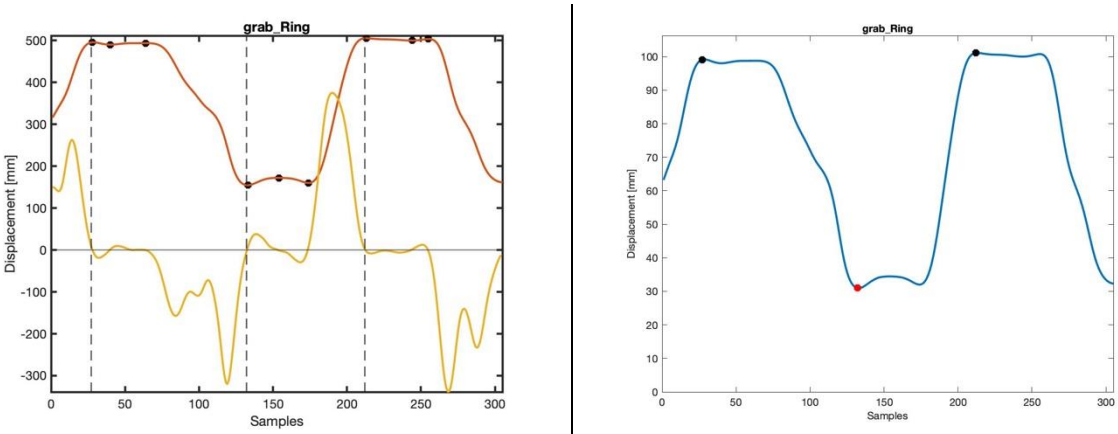


Figure 4.36: Maximum and Minimum points determination in the First task results of The Grab of Ring

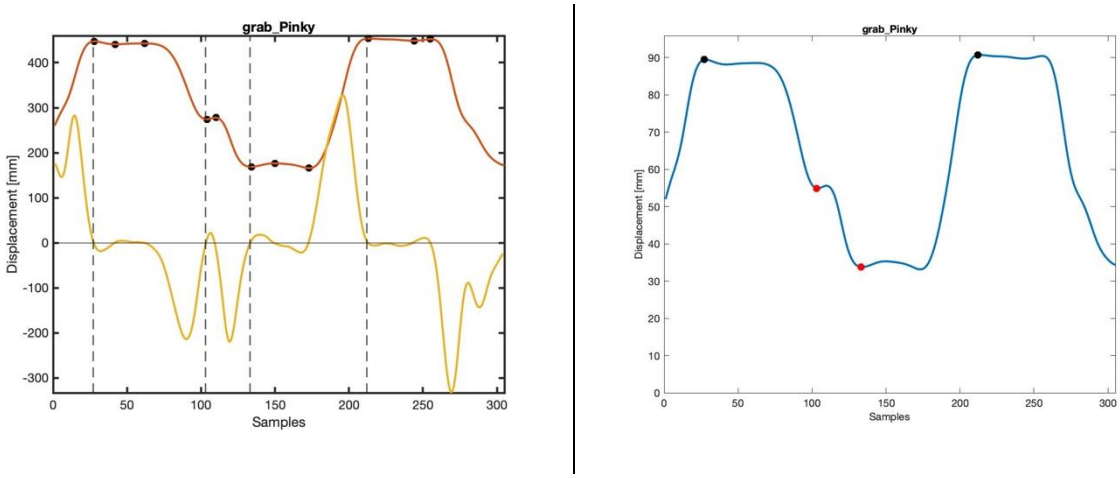


Figure 4.37: Maximum and Minimum points determination in the First task results of The Grab of Pinky

4.3 Thumb towards the other fingers one by one task

The below Figures, from Figure 4.3 to 4.42, show the mainly steps of this task, namely completely open, Thumb touches Pinky, Thumb touches Ring, Thumb touches Middle, Thumb touches Index.

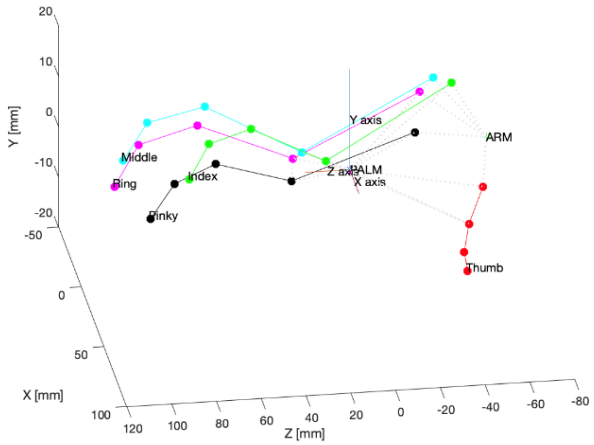


Figure 4.38: Hand completely open

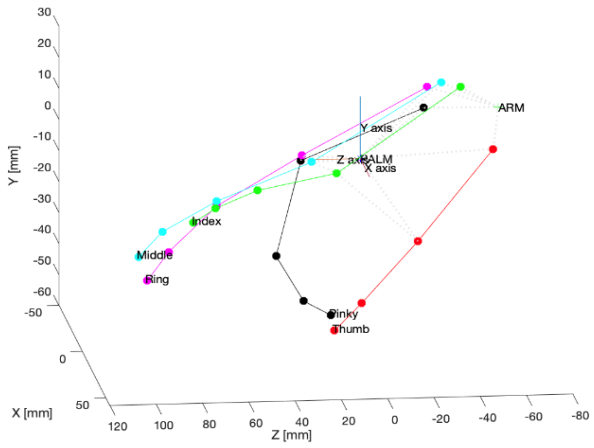


Figure 4.39: Touch between Thumb and Pinky

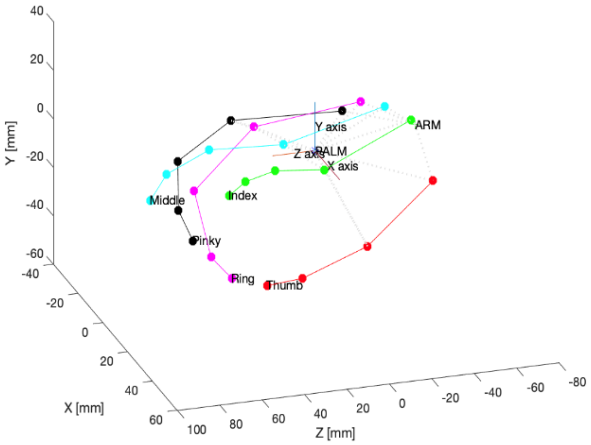


Figure 4.40: Touch between Thumb and Ring

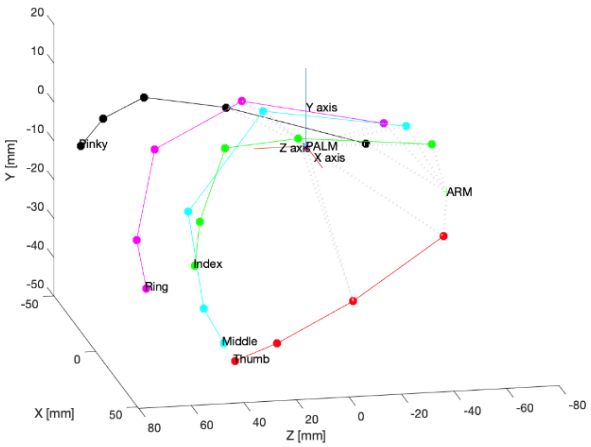


Figure 4.41: Touch between Thumb and Middle

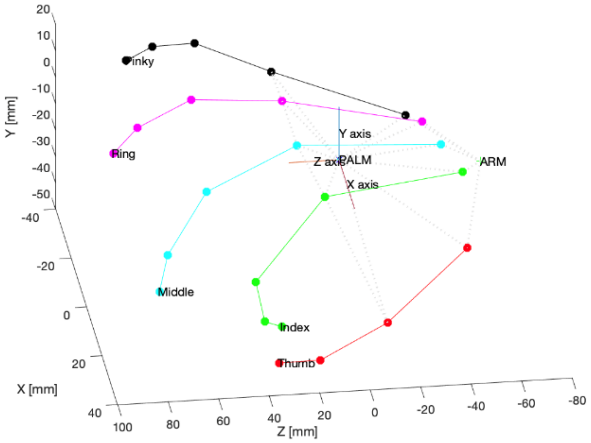


Figure 4.42: Touch between Thumb and Index

4.3.1 by function_palm_digits

The below Figures from 4.43 to 4.47 show four different graphs, which each corresponds to a joint of the finger, from the closer to far joint has the following colors: Blue, Red, Yellow, and Purple. Moreover, are shown both the unfiltered and filtered signals.

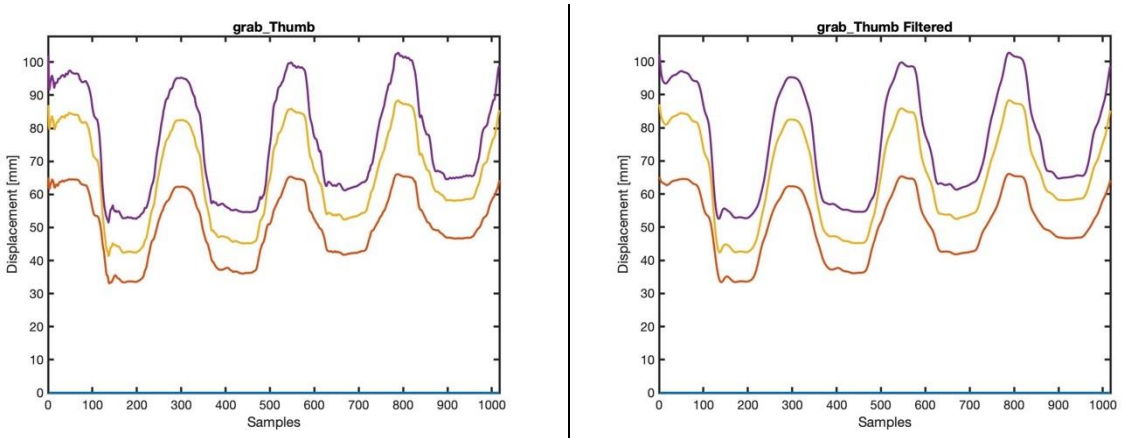


Figure 4.43: Second task results of the Grab of Thumb

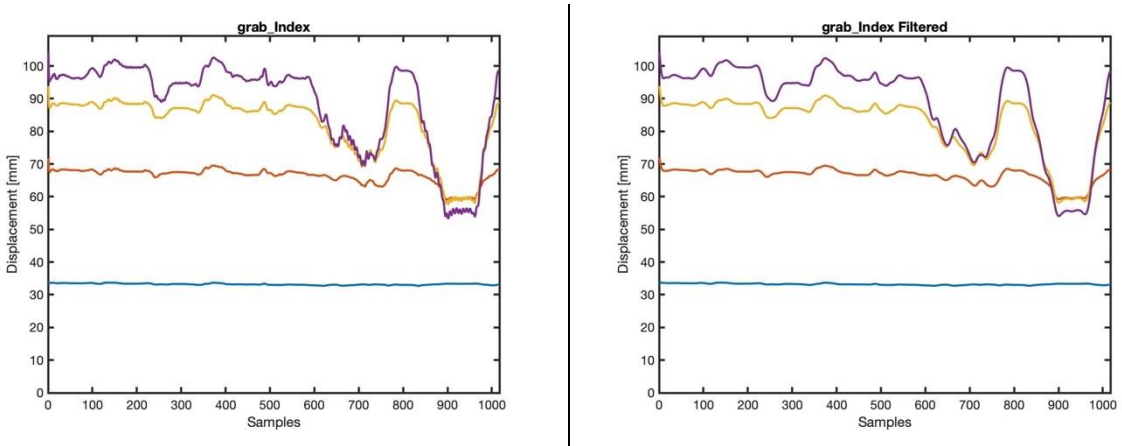


Figure 4.44: Second task results of the Grab of Index

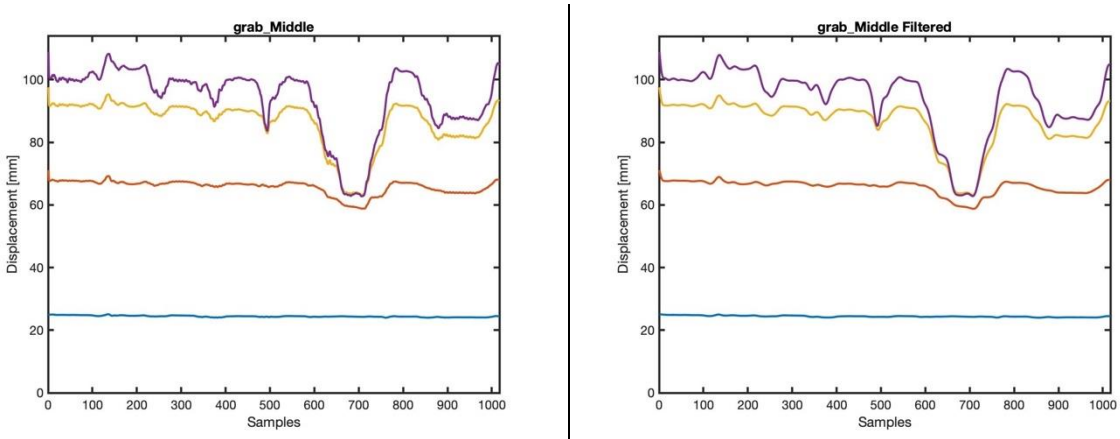


Figure 4.45: Second task results of the Grab of Middle

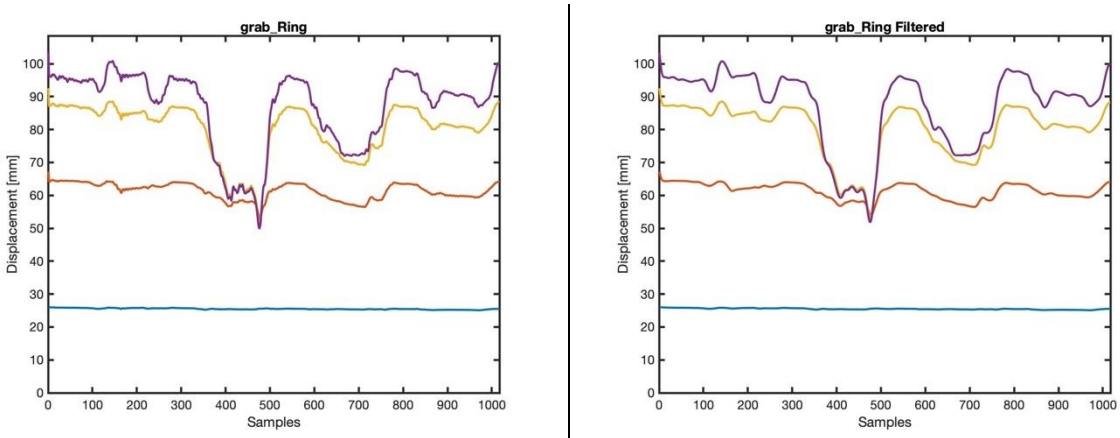


Figure 4.46: Second task results of the Grab of Ring

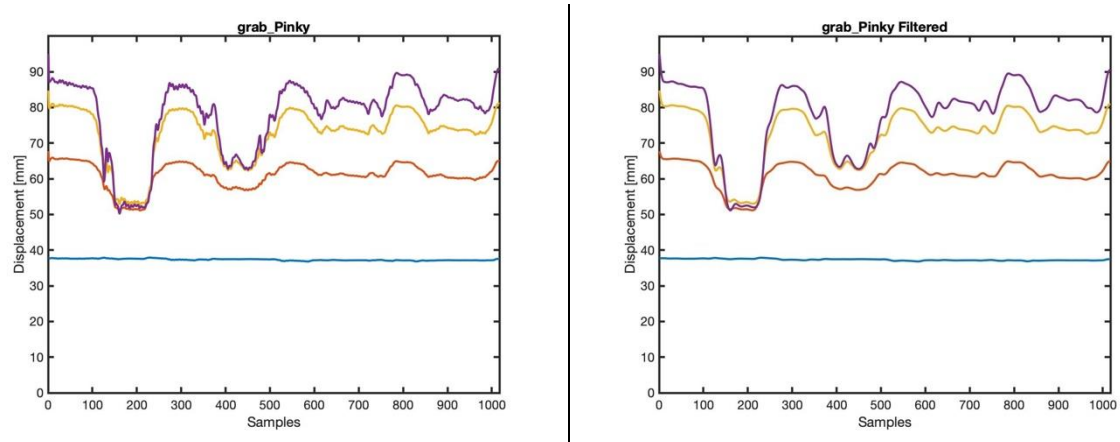


Figure 4.47: Second task results of the Grab of Pinky

4.3.2 by function_thumb_other_digits

The below Figures from 4.48 to 4.51 are composed of three different curves, unfiltered and filtered. The meaning of each graph corresponds: in Yellow the distance between the tips of the thumb and the other fingers, in Blue and Orange, instead, are the distances between the thumb joint, immediately after the tip of the thumb, and the other two joints of each finger (as is shown in Figure 3.4).

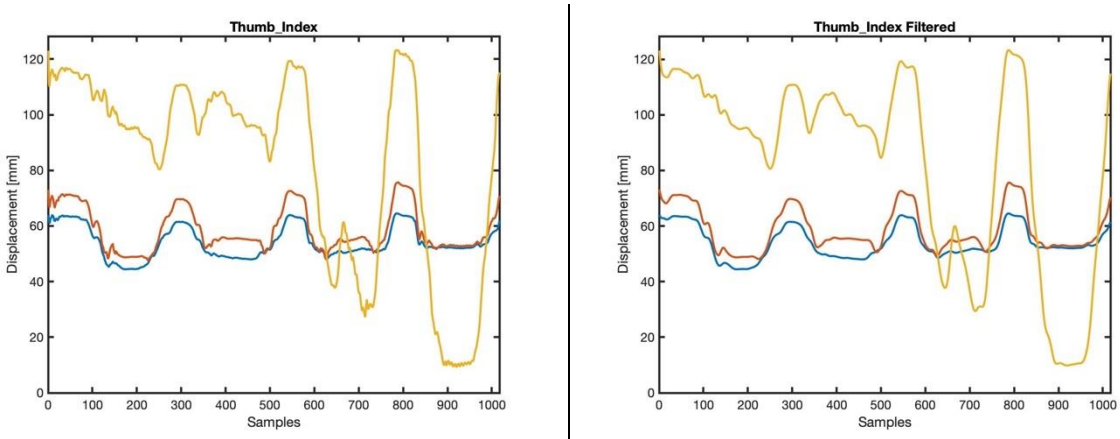


Figure 4.48: Second task results of the Thumb towards Index

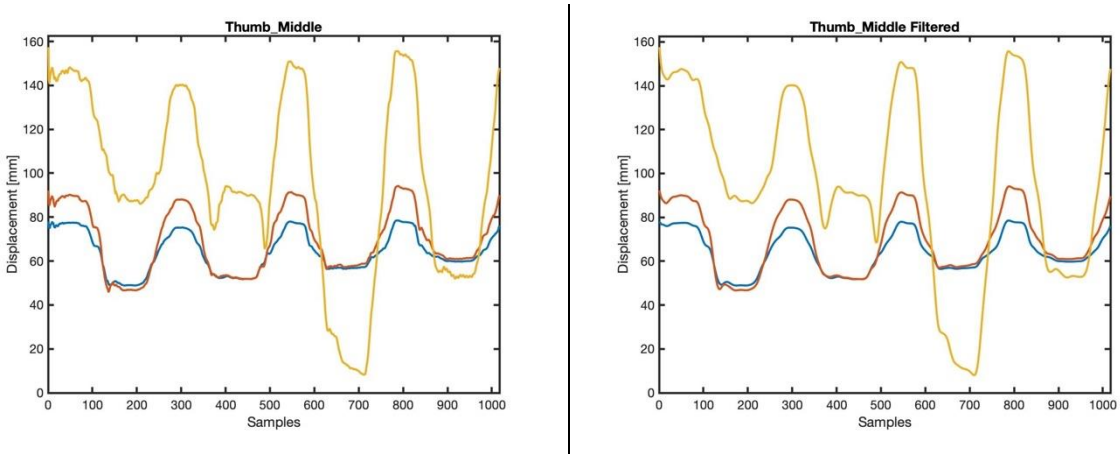


Figure 4.49: Second task results of the Thumb towards Middle

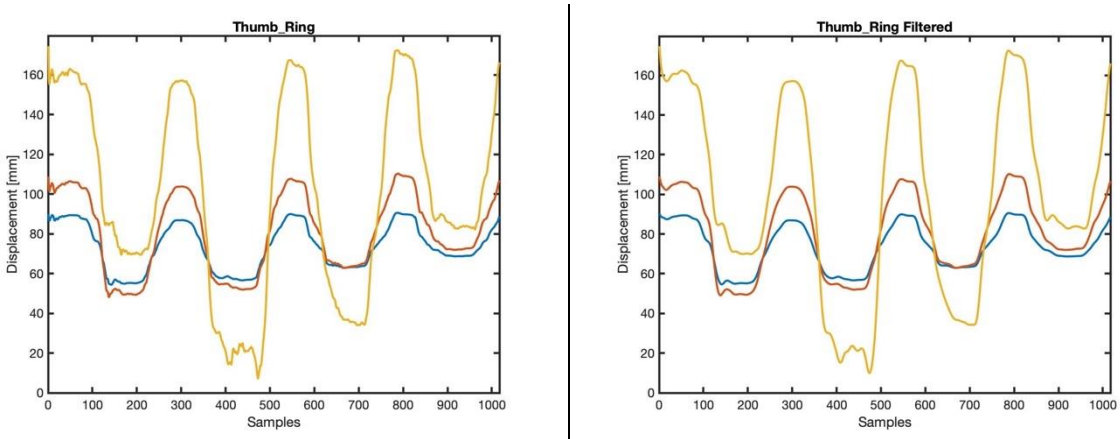


Figure 4.50: Second task results of the Thumb towards Ring

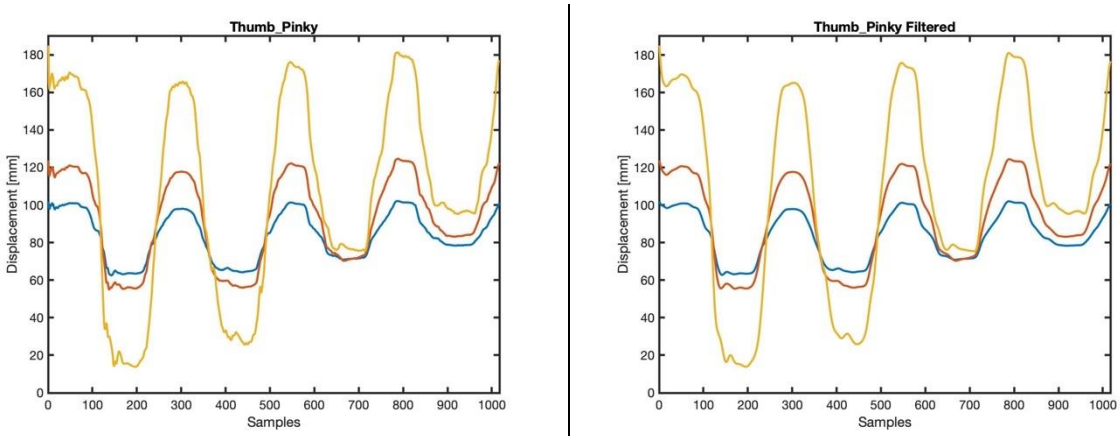


Figure 4.51: Second task results of the Thumb towards Pinky

4.3.3 by pitch_digits

From Figure 4.52 to Figure 4.56, show the results of the pitch angles in each finger. The colors represent: in Blue, the angle between the first and second phalanx, in Orange, the angle between the second and third phalanx, and in Yellow, between the third and fourth phalanx. On the left is the unfiltered data and, on the right, the filtered one,

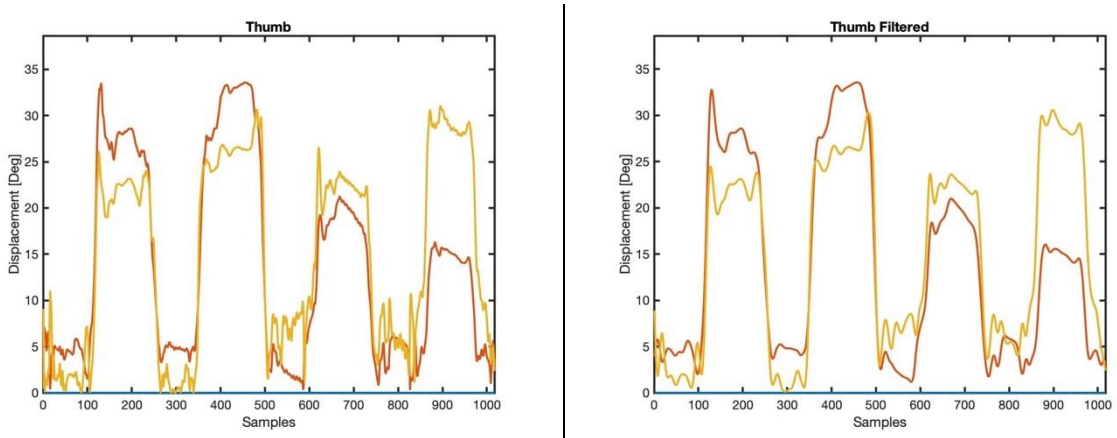


Figure 4.52: The Pitch angles in the Thumb in the second task

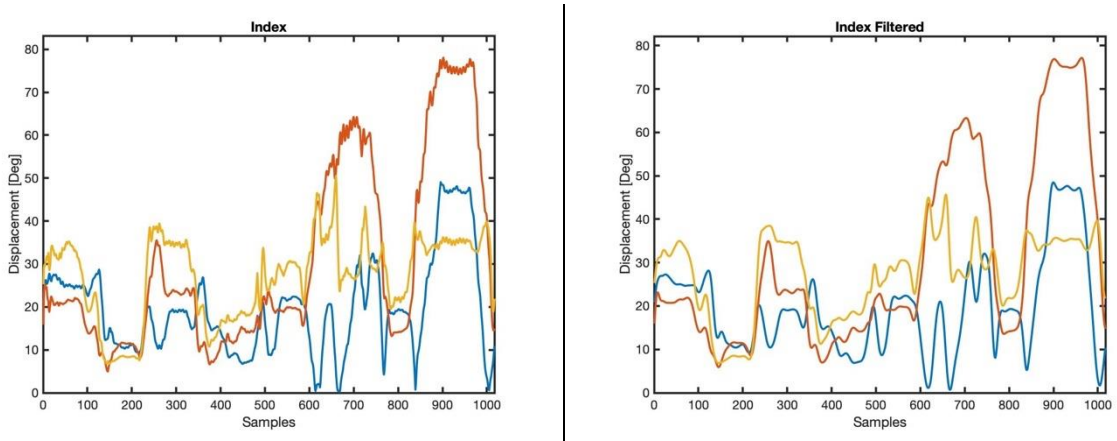


Figure 4.53: The Pitch angles in the Index in the second task

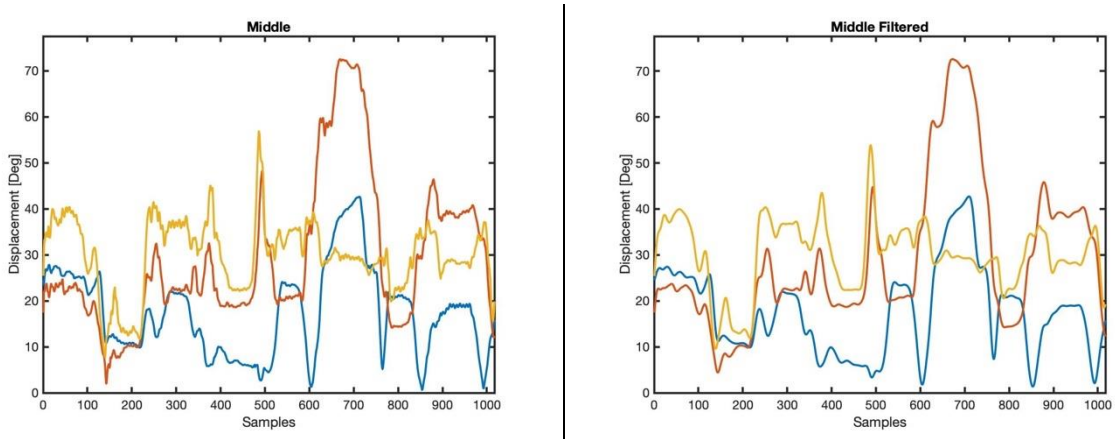


Figure 4.54: The Pitch angles in the Middle in the second task

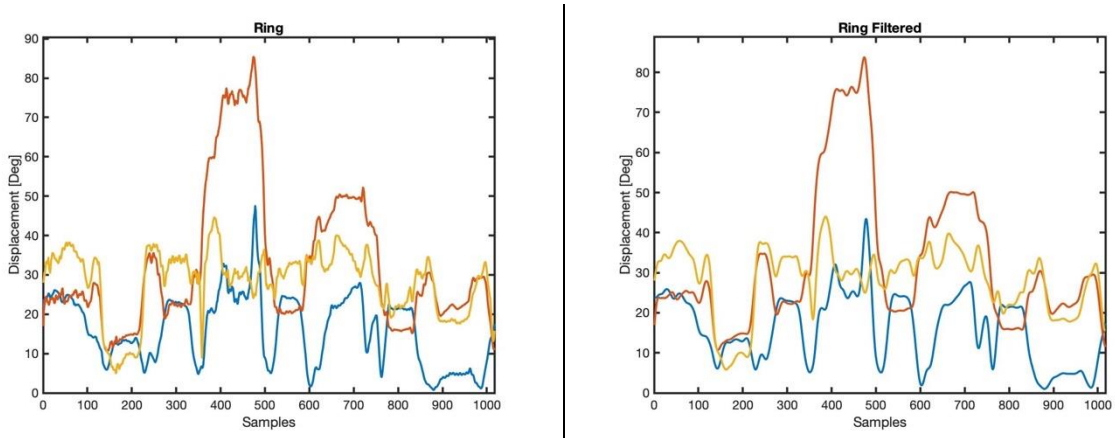


Figure 4.55: The Pitch angles in the Ring in the second task

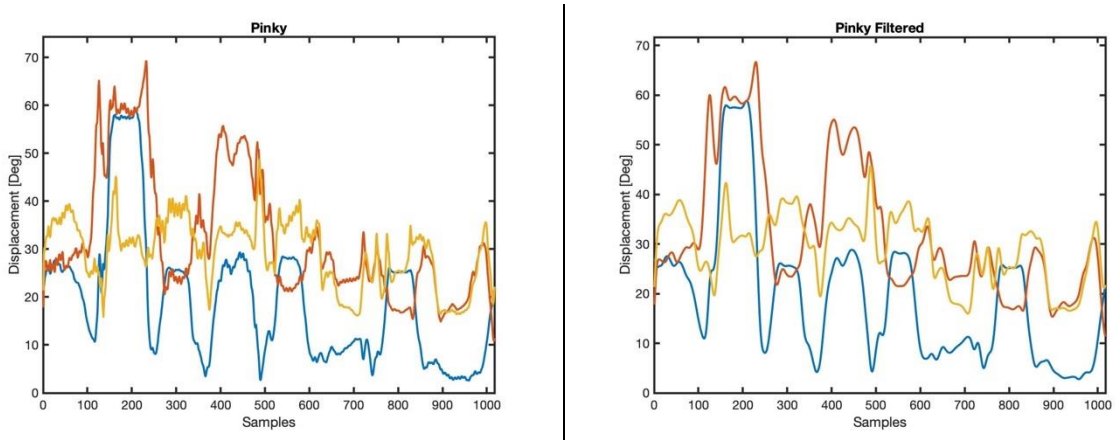


Figure 4.56: The Pitch angles in the Pinky in the second task

4.3.4 by yaw_digits

The resulting graphs from Figure 4.57 to 4.60 are the angle representation between two consecutive fingers. The data on left are unfiltered, instead on right are filtered.

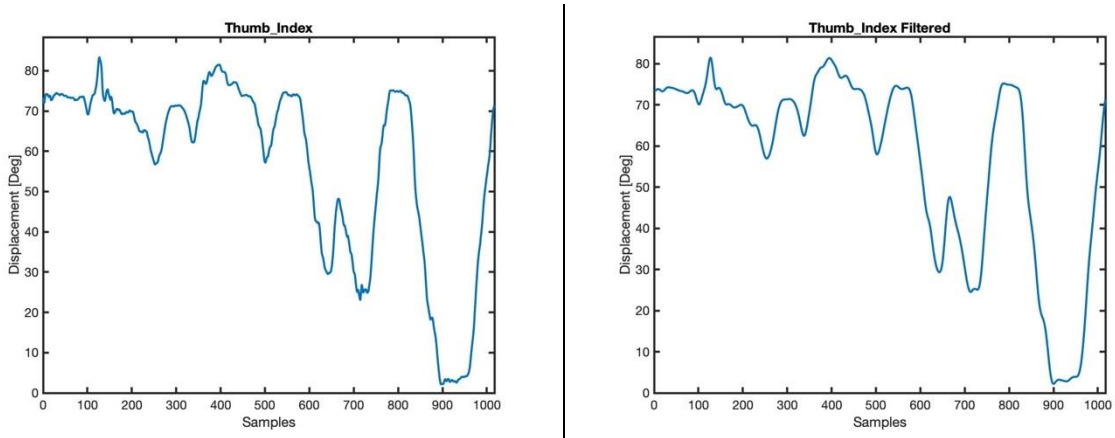


Figure 4.57: The Yaw angles between Thumb and Index in the second task

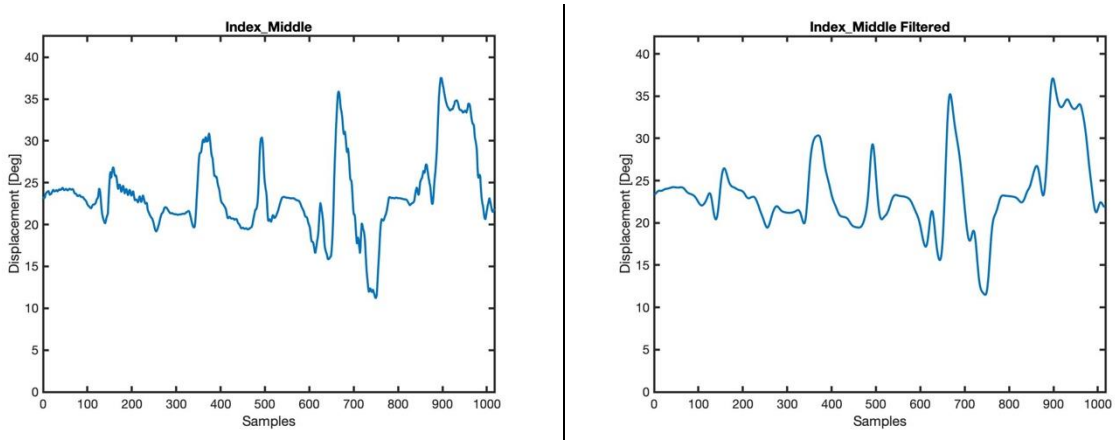


Figure 4.58: The Yaw angles between Index and Middle in the second task

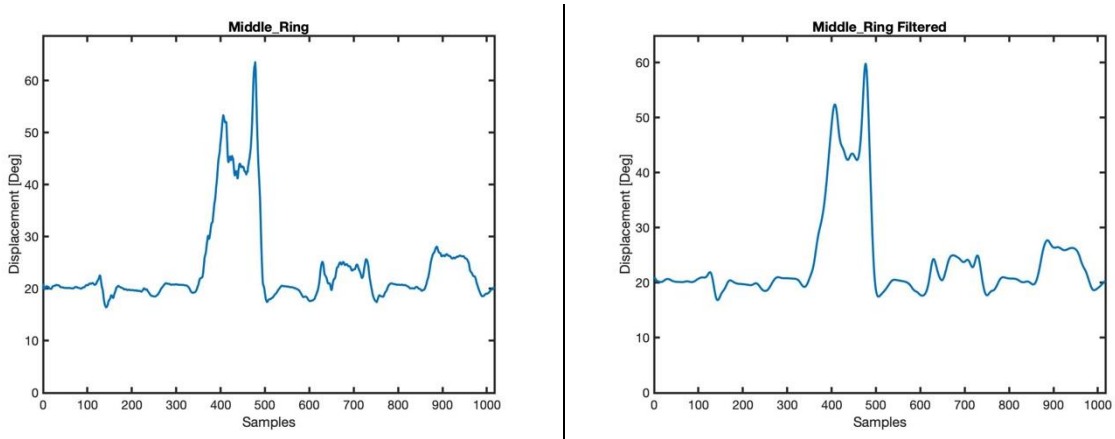


Figure 4.59: The Yaw angles between Middle and Ring in the second task

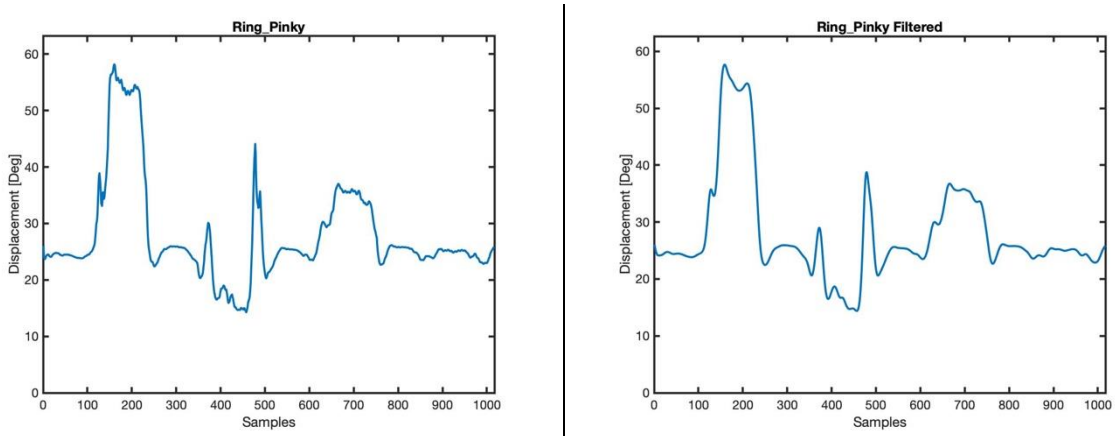


Figure 4.60: The Yaw angles between Ring and Pinky in the second task

4.3.5 by computation_area

The following Figures, from Figure 4.61 to 4.69, are shown the higher displacement graphs (the resulting graphs from Section 4.2.1 and 4.2.2), which have been multiplicated by a constant, because to visualize better those graphs, which explains why the red graphs have an amplitude completely different than their previous ones. In Yellow are shown their first derivative, and in black dots are the corresponding zero-crossing of their first derivative.

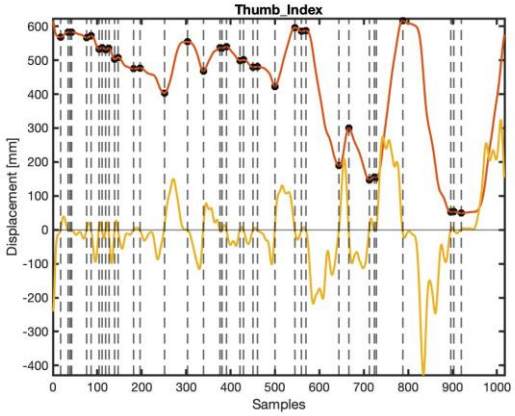


Figure 4.61: Zero-crossing points determination for the Thumb towards Index

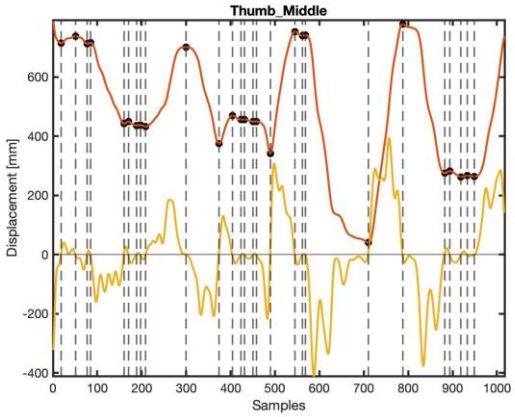


Figure 4.62: Zero-crossing points determination for the Thumb towards Middle

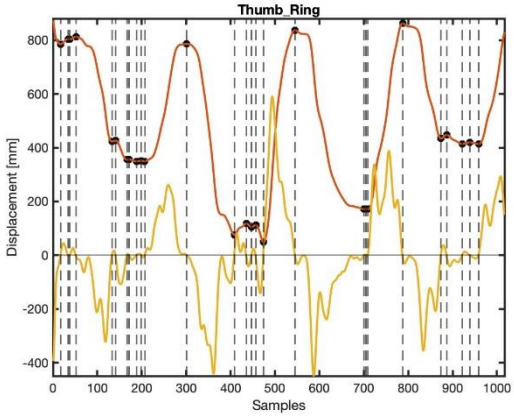


Figure 4.63: Zero-crossing points determination for the Thumb towards Ring

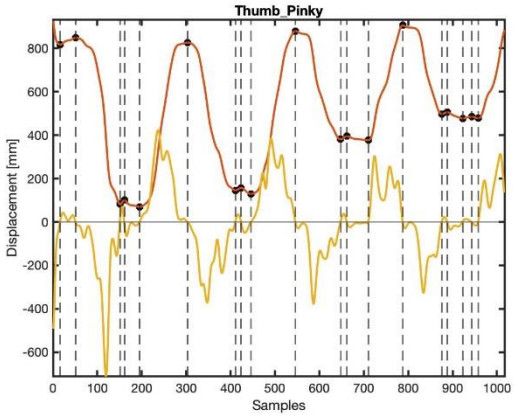


Figure 4.64: Zero-crossing points determination for the Thumb towards Pinky

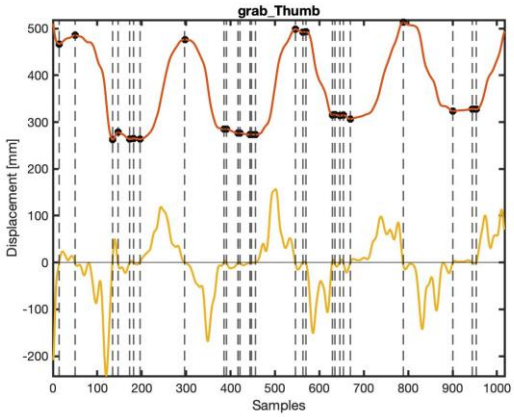


Figure 4.65: Zero-crossing points determination for the Grab of Thumb

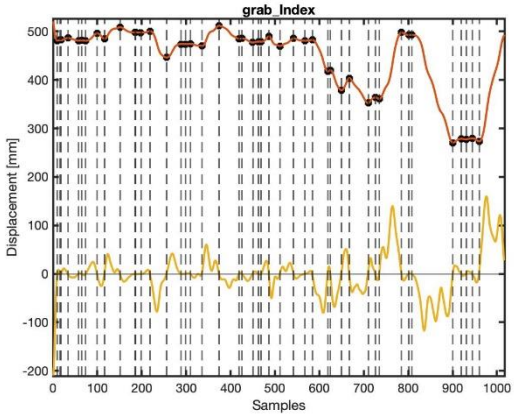


Figure 4.66: Zero-crossing points determination for the Grab of Index

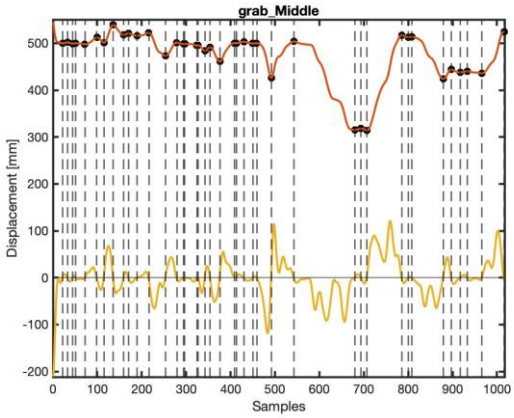


Figure 4.67: Zero-crossing points determination for the Grab of Middle

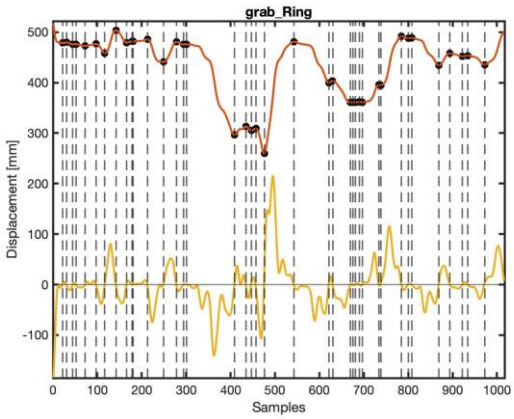


Figure 4.68: Zero-crossing points determination for the Grab of Ring

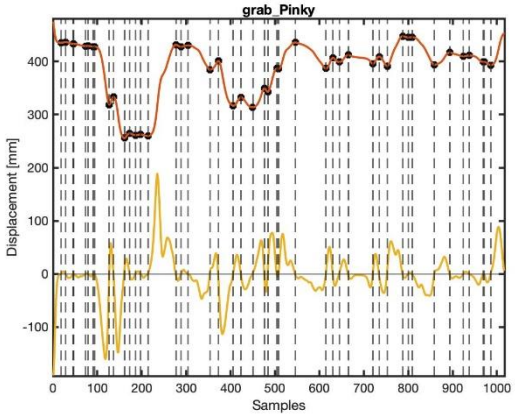


Figure 4.69: Zero-crossing points determination for the Grab of Pinky

4.3.6 by the functions to find the Max and Min points

The Figures from Figure 4.70 to 4.78 show on the left the outputs from the “area” code and on the right the outputs from the “slope” code, both described in Section 3.1.7. the curves inside each graph are the same as specified in the Section 4.3.5. Moreover, for the right graph is expressed in black dots the maximum points, and in red the minimum points, according to the “slope” algorithm.

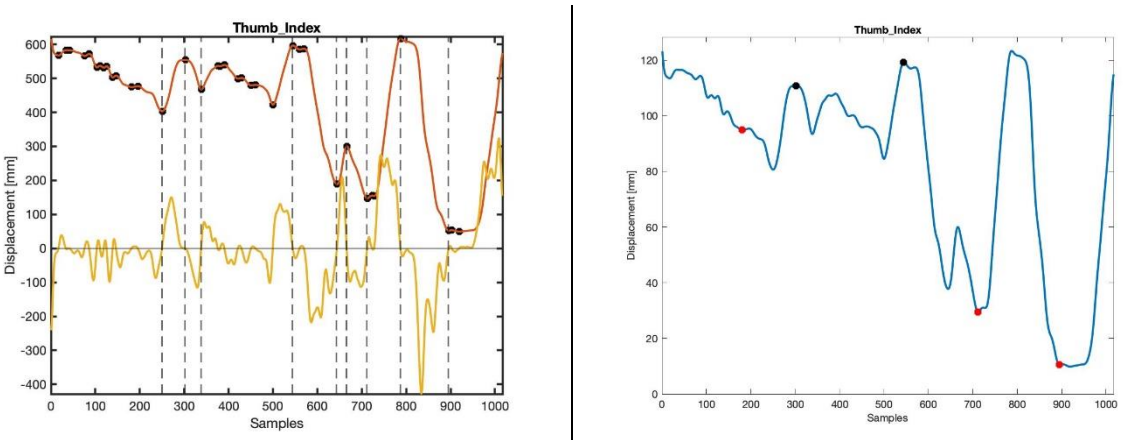


Figure 4.70: Maximum and Minimum points determination in the Second task results of Thumb towards Index

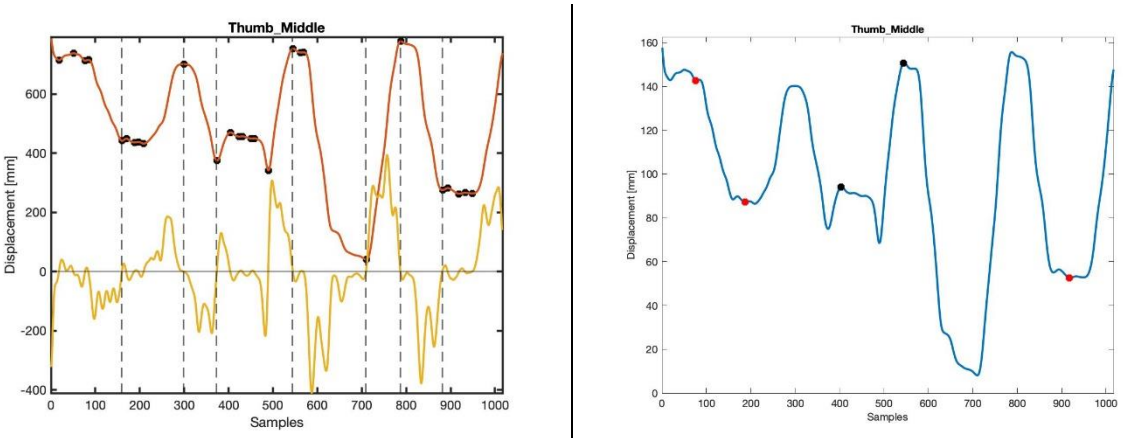


Figure 4.71: Maximum and Minimum points determination in the Second task results of Thumb towards Middle

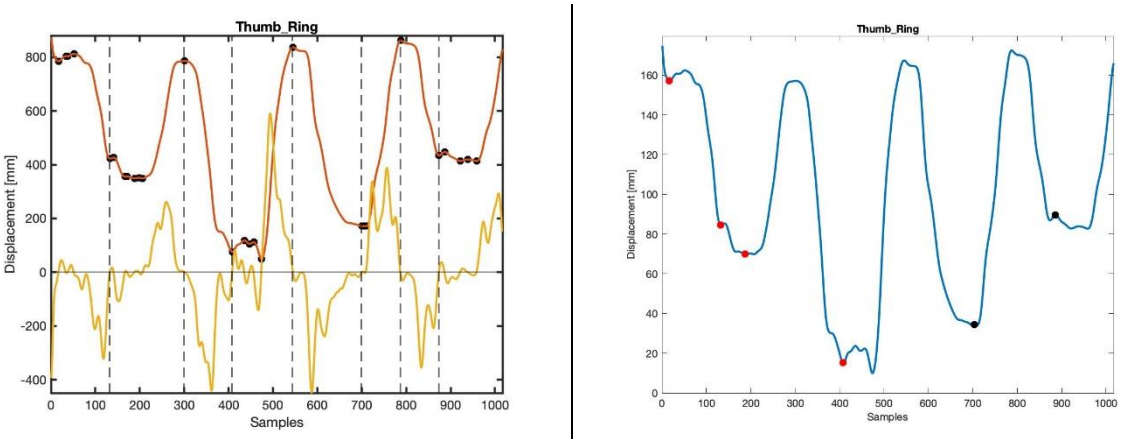


Figure 4.72: Maximum and Minimum points determination in the Second task results of Thumb towards Ring

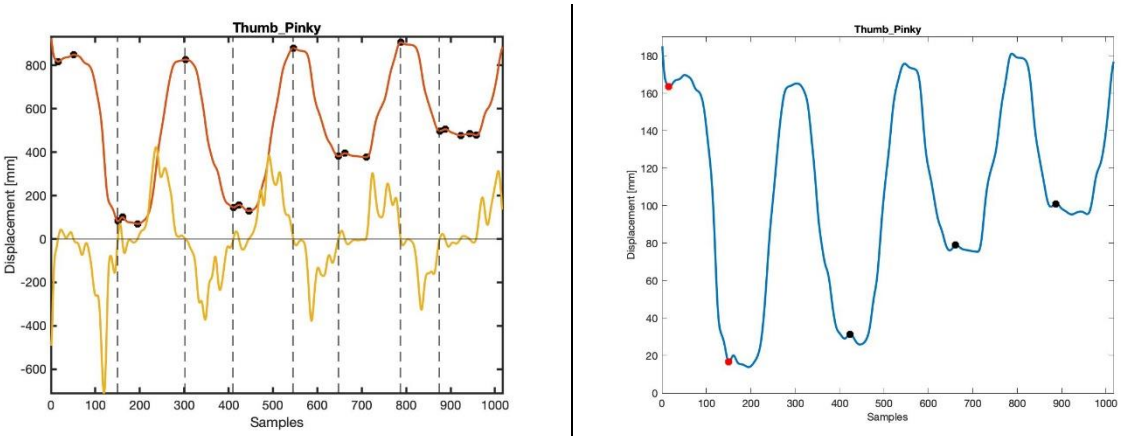


Figure 4.73: Maximum and Minimum points determination in the Second task results of Thumb towards Pinky

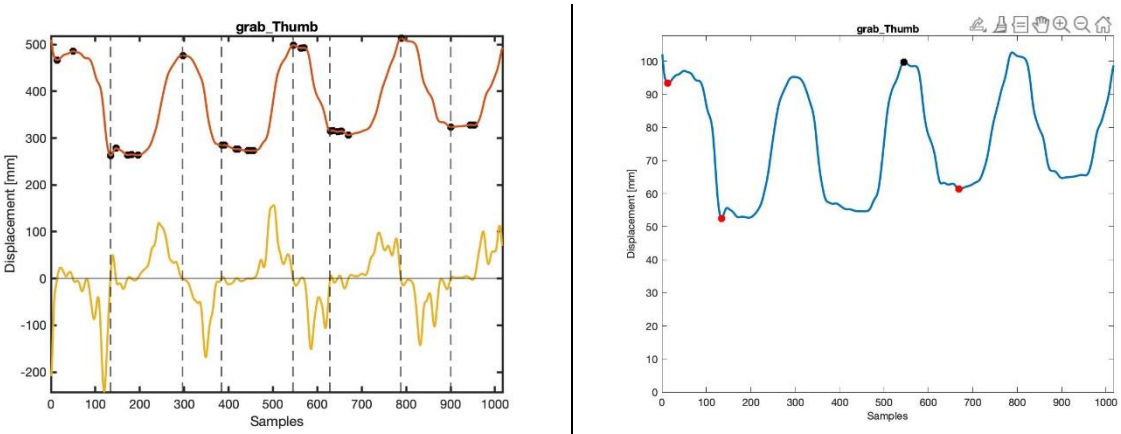


Figure 4.74: Maximum and Minimum points determination in the Second task results of The Grab of Thumb

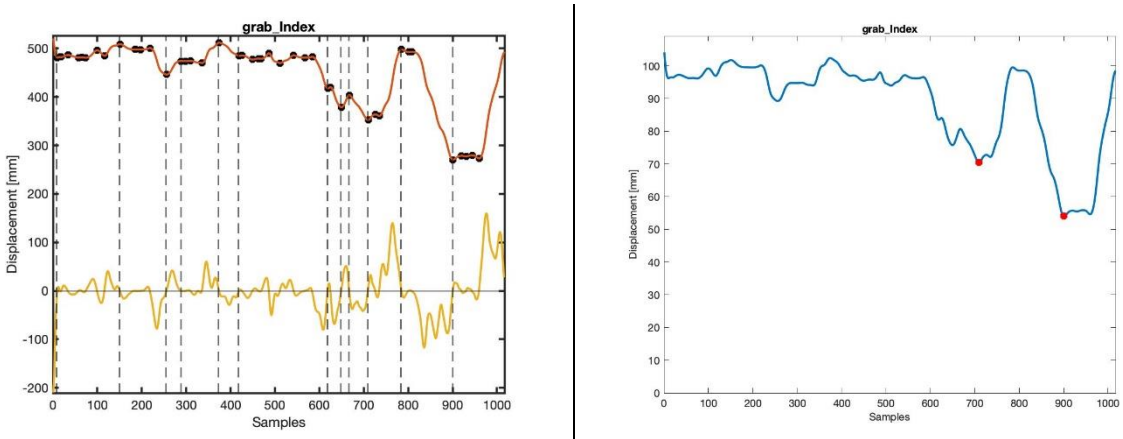


Figure 4.75: Maximum and Minimum points determination in the Second task results of The Grab of Index

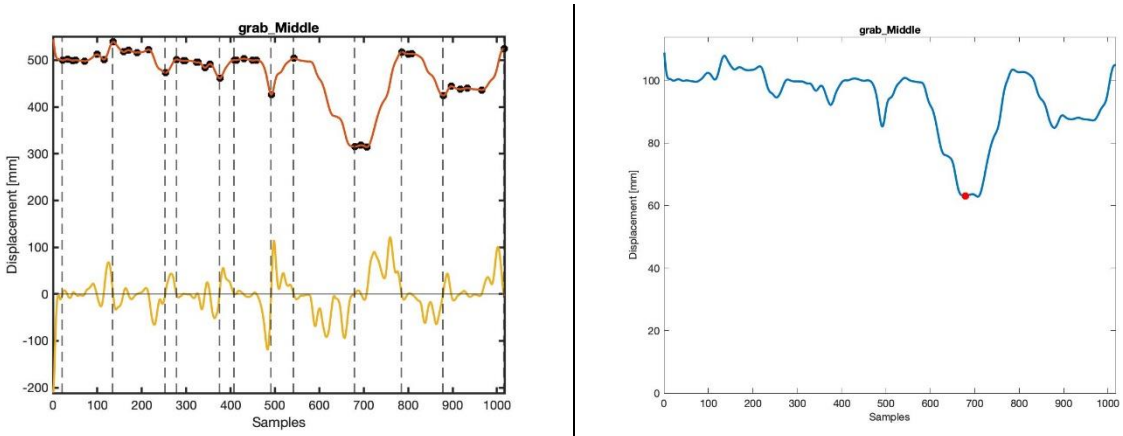


Figure 4.76: Maximum and Minimum points determination in the Second task results of The Grab of Middle

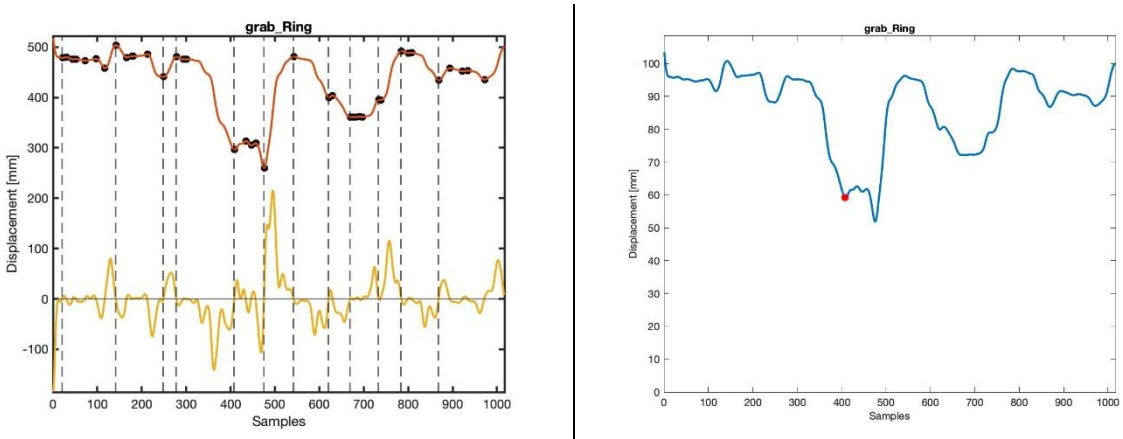


Figure 4.77: Maximum and Minimum points determination in the Second task results of The Grab of Ring

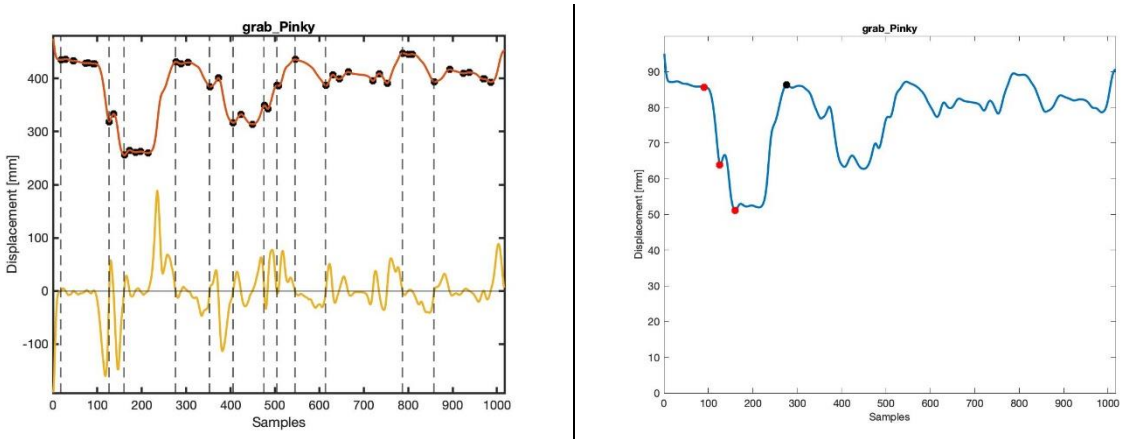


Figure 4.78: Maximum and Minimum points determination in the Second task results of The Grab of Pinky

Chapter 5

5 Analysis of the Results

The analysis is divided according to the results for each specific function used to obtain them, and each following paragraph involve both types of tasks, as explained before. The first graphs for each type of task serve to visualize the hand movement, which is helpful to interpret the rest of the plots and features. As we can see, these plots allow to see the hand movement in every possible viewpoint in the space.

5.1 function_palm_digits

The resulting graphs from Figure 4.2 to Figure 4.6, for the “open and close” task, have a period behavior which is what we expected. Being a periodical task, defined by just opening the hand and closing the hand, these movements determine an increase of the displacement from the palm and a decrease of it when coming back towards the center of the palm. Therefore, talking about which features could be extracted from these signals, we could obtain both amplitude and period.

Instead, in the resulting graphs for the second task from Figure 4.43 to Figure 4.47, we are able to say the same consideration was expressed for the previous task. Because during the pre-surgery acquisition and during the surgery the patient must make the task different times at least in a time interval, which in our acquisition was different from 5 to 10 seconds, however, we are thinking of increasing it until 30 seconds to obtain long signals with different repetitions of the same task, generating every time a sort of periodic signal. Regarding the behavior of these graphs, it corresponds to what we expected: the thumb signals describe a periodic signal in which each period corresponds to a decrease of distance with another finger, and to know which is the finger, it is sufficient to determine, among the other signals, which shows a minimum at the same time as the thumb.

5.2 function_thumb_other_digits

Regarding the open and close task, the graphs from Figure 4.7 to Figure 4.10 have a behavior such as the “function_palm_digits” that is due to the task movement, being an opening and closing of the hand. It generates a change of the displacement in phase with the previously described results.

Moreover, these comments are valid also for the thumb towards the other finger one by one task. The results are shown from Figure 4.48 to Figure 4.51.

5.3 pitch_digits

Consider the Eq. (1), we obtain that the results for pitch and yaw angles is small when the two consecutive segments are almost parallel. Therefore, independently of the studied task in this research, when the fingers have the maximum distance from the center of the palm, the values of pitch are small because each bone/segment which composes the finger is almost aligned with the consecutive one, instead the values are big when the hand is closed (extreme change of relative orientation between the consecutive bones/segments).

What has been described above is presented in the resulting graphs for both types of tasks. When the finger is completely extended, the pitch angles are small, instead when the finger is coming close to the center of the palm, these values become higher. It is necessary to mark that the figures from Figure 4.11 to Figure 4.15 and from Figure 4.52 to 4.56, must be lower than 90 degrees, since they are the result of an inverse cosine, and also for anatomical constraints.

For both Figure 4. 11 and Figure 4.52, there is a blue curve equal to zero during the whole task. This result graph is due to the default configuration of the thumb data from the LMC, i.e., is considered as a finger with three phalanxes as the other fingers but the coordinates of the first and second joint are the same, thus the difference of these coordinates is zero always.

5.4 yaw_digits

The results functions are computed with the same Eq. (1). For the resulting graphs of open and close task, from Figure 4.16 to 4.19, only Figure 4.16 has a trend like the resulting graphs from “function_palm_digits” and “function_thumb_other_digits”, instead the other ones have bigger values when the hand remains close rather than when the hand is open.

On the contrary, in the resulting graphs of the second studied task, from Figure 4.57 to Figure 4.60, the behavior is interesting: according to the touched finger, the yaw values between the touched and its consecutive increases at least twice their normal values (as is clearly shown in Figure 4.59 and Figure 4.60). Instead, for the graph in Figure 4.58, there are large angle changes during the whole task, but when the thumb touches the middle finger there is high displacement (between 650 and 800 samples), and regarding Figure 4.57 we could describe it with the same comments of Figure 4.58 but in this case, the high displacement is present over the 800 samples.

5.5 computation_area

As we can see from Figure 4.20 to Figure 4.28 and from Figure 4.61 to Figure 4.69, depending on the intersection points of the first derivative of the distance signal, which is chosen according to its high sensitivity (higher amplitude among the other computed signal). To detect the points, which are useful to extract the features, is exploited the values in the areas below the first derivative function. According to the sign of the area, positive or negative, we are able to define if the zero-crossing points are max or min. This definition is possible according to the previous area of each zero-cross point, if the area is greater than zero the zero-crossing point is a maximum in the primitive function or is a minimum point if the area is lower than zero. Moreover, there are some graphs where the last points are not selected but is due to the no crossing of zero by its first derivative, as we can see, for instance, in Figure 4.23, or in Figure 4.23.

In the first task, the found points are lower in number than in the second task, due to both the time interval necessary to execute the task and its simplicity. In the Figures from Figure 4.20 to Figure 4.28, there are small other points around the same amplitude (i.e., relative maximum or minimum points), which characterize the oscillation of the patient fingers during the execution of the task. But for these graphs, as we can see, the selection of the points to extract the features must be developed in a way to determine the best max or min inside the selected ones and take care if there are points between the other points, as the Figure 4.28 shows (two points between the possible best maximum and minimum, around at 110 sample).

Regarding the second task, identified the desired points according to the zero crossing, the analysis of them is harder than in the previous case due to the difficulty of this task type. The density of selected points is very high, for the graphs present in Figure 4.66 to Figure 4.69. This high presence of points depends on the relative oscillation between the fingers and the center of the palm, and between the thumb and the other digits as well.

The Figures from 4.29 to 4.37 show the results from the two codes used to find the best max and min to extract the amplitude and period features. Both methods, for the "open and close" task, identify almost all the desired points but there are some differences between the results. The outputs of the script, which uses the value of the area preceding the first derivative point of zero-crossing, on the left of these figures achieves better results. But, in only the output represented in Figure 4.33, there is a missing point in the first part of the graph when the thumb is completely extended. On the contrary, the outputs of the other code, which exploits the line between consecutive zero-crossing points, on the right of these figures have obtained good results as well. All the points in Figure 4.33 are correctly detected, but there are 4 out of 9 without the determination of the last point. This missed detection is the effect of how the code was defined, which takes information from the slope of the interpolation line before the studied point, therefore all last points are removed.

With the above considerations, regarding the open and close task, we can determine that the code which depends on the areas is better than the second one, both from the computational point of view and its simplicity, and because among 9 signals, for

only 1 the code does not identify a point, instead the second got 4 missed determinations of the last points.

Instead, for the second task, “thumb towards the other fingers one by one”, with the application of both codes to find the max and min, the results are shown in the figures from Figure 4.70 to Figure 4.78. In each figure, on the left there are the results from the codes based on the area’s values, and on the right from the second code based on the angular slope.

Due to the oscillation of the fingers during both tasks are detected points while the fingers are in their fully opening or closing. Therefore, we used the two codes to obtain and estimate their results, if are suitable to the extraction of the desired features, in the following are discussed:

- The “area” code got a high number of interesting results, which are shown on the left of the above figures, in almost all “*hand*” signals for both tasks. However, these “best” points determination depends particularly on the complexity of the task, in other words, on the type of the task. Therefore, to overcome this type of possible errors (i.e., determination of undesired points as “best”), we can comment on the necessary to personalize the algorithm according to the type of task to improve the identification by this “area” code, otherwise could obtain wrong values of extracted features specially the period (which is computed between the max and min values of the specific gesture). Regarding, the second studied task, the comment is mainly regarding on their difference. As we can see, the output graphs of “function_thumb_other_digits”, figures from Fig. 4.70 to Fig. 4.73, have a different signal trend, and of “function_palm_digits” as well, figures form Fig. 4.74 to 4.78. Regarding the outputs of the first function (thumb_other_digits), the results in Fig 4.70, are not enough to determine the moments in which the thumb hit the other fingers except to the index and the middle (are determined by the last two identified minimum points). About the other results of this function, are useful to extract the features that we want (but are not determined the first points, which are max in this case). Moreover, on the results of the “palm_digit” function, excepted to the Fig 4.74 results, are completely useless because the identification got too many points, which are not wanted.

- The “angular slope” code (on the right of the above figures), instead, with respect to the “area” code acquired a lower number of points in every resulted graph but there are a high number of miss identification of points, for the second task, “thumb towards the other fingers one by one”. These missing concern especially of global minimum points in some result graphs, for instance Fig 4.71 shown this error (is not acquired the minimum point around 700 sample). Or the selection of some minimum points as if they were maximum points, as shown Fig 4.72, and above all in Fig 4.73. But among these “errors” there are also some interesting results, as Fig 4.76 and Fig. 4.77 shown, i.e., determine the points, with not high accuracy, in which the thumb hit the corresponding finger of the graph, in these case middle and ring. Lastly, comparing the computational view of point of this code their results are not sufficient to improve this type of code respect to the “area” one.

Also, for this task the results of the “area” code are better than the “angular slope” code.

Chapter 6

6 Conclusion and Future Application

The task assessment of the patient during the awake neurosurgery is visually performed by the neuropsychologist, who considers if the patient's capabilities are preserved, and eventually drives the surgeon to reach the target by minimizing the damaging cut. The main problem is that such evaluations are mostly subjective, only qualitative, and therefore subject to probable evaluation errors or missing events. The solution proposed in this thesis represents a pioneering approach to solve this problem: the LMC data are processed to support the neuropsychiatrist's decision during awake neurosurgery. The results obtained are interesting for identification in almost all signals, but the algorithm used to determine the points from which to extract features has some limitations. These limitations relate to the algorithm's ability to recognize what type of point is the selected one because, by not applying a recognition check for these points, the feature extraction algorithm does not provide feasible values to understand whether the task was performed well or poorly. Therefore, possible solutions are to apply specialized algorithms depending on the task, to decrease the risk of misidentification, or to define a single algorithm that can detect and extract features regardless of the type of task. Regarding the proposed approach, interesting results have been obtained for the "area" code in almost all the considered tasks. A possible improvement of this code could be made to generalize the algorithm and apply it for each type of task, or to create several specialized codes, using it as a starting point, depending on the considered task.

Regarding the "open and close" task, we are going to improve the detection with the insertion of an interval condition in amplitude to get the differences between max/min points and points in the middle. The idea could be to exploit these points to determine features by, for example, setting up warning algorithms to support intraoperative decisions regarding the considered task.

Instead, due to the complexity of their graphs (with respect to the first studied task), the "thumb towards other fingers one by one" task is a hard to elaborate. The risk of error

increases as much as the complexity of the task increases, thus should be necessary to apply other algorithms capable to detect other types of features.

The proposed method also includes the study of the angles (pitch and yaw angles), and these types of outputs should be mainly exploited to realize warning functions or used to define a condition to obtain a specialized function to determine the quality of the exercise performed by the patient. Therefore, they could be useful for future applications.

In conclusion, the use of LMC during awake neurosurgery can be applied in the not-too-distant future, because by using this first study as a starting point for the implementation of the LMC data analysis, we are able to properly support the neuropsychiatrist's decisions using the Leap Motion Controller.

Finally, we plan to improve our analysis process, proposed in this study, to arrive at the development of a fully customized support device that can support this type of intervention with more objective decisions.

Reference

[1] R. Felicetti, E. Frontoni, S. Iarlori, L. Migliorelli, L. Scoppolini Massini, A. Monteriù, S. Bonifazi, R. Trignani, S. Vecchioni, *Smartawake: a real-time decision support system for intraoperative assessment in awake surgery*. CRAS, 2022

[2] M. A. Morrison, F. Tam, M. M. Garavaglia, L. Golestanirad, G. Hare, M. D. Cusimano, T. A. Schweizer, S. Das, and S. J. Graham, *A novel tablet computer platform for advanced language mapping during awake craniotomy procedures*. Journal of neurosurgery, vol. 124 4, pp. 938–44, 2016.

[3] N. U. F. Hameed, Z. Zhao, J. Zhang, L. Bu, Y. Zhou, L. Jin, H. Bai, W. Li, J. Tang, J. Lu, J. Wu, and Y. Mao, *A Novel Intraoperative Brain Mapping Integrated Task-Presentation Platform*. Operative Neurosurgery, vol. 20, no. 5, pp. 477–483, 02 2021.

[4] M. Khademi, H. Mousavi Hondori, A. McKenzie, L. Dodakian, C. V. Lopes, and S. C. Cramer, *Free-hand interaction with leap motion controller for stroke rehabilitation*. CHI'14 Extended Abstracts on Human Factors in Computing Systems, 2014, pp. 1663–1668.

[5] J. Wu, N. Yu, Y. Yu, H. Li, F. Wu, Y. Yang, J. Lin, J. Han, and S. Liang, *Disturbance-observer-based control and related methods—An overview*. Parkinson's Disease, vol. 2021, 2021.

[6] https://www.ultraleap.com/datasheets/Leap_Motion_Controller_Datasheet.pdf

[7] Ultraleap site, "*Ultraleap Hand Tracking: Technical Introduction*", 06/29/2020. [Online]. Available: https://cms.ultraleap.com/app/uploads/2020/07/Ultraleap-Hand-Tracking-Technical-Introduction-1.pdf.

[8] https://www.ultraleap.com/company/news/blog/how-hand-tracking-works/

[9] Wu J, Yu N, Yu Y, Li H, Wu F, Yang Y, Lin J, Han J, Liang S, *Intraoperative Quantitative Measurements for Bradykinesia Evaluation during Deep Brain Stimulation Surgery Using Leap Motion Controller: A Pilot Study*. Parkinsons Dis. 2021 Jun 15; 2021:6639762.

[10] Weichert, F., Bachmann, D., Rudak, B., & Fisseler, D. (2013). Analysis of the accuracy and robustness of the Leap Motion Controller. *Sensors (Switzerland)*, *13*(5), 6380–6393.

[11] Smeragliuolo, A. H., Hill, N. J., Disla, L., & Putrino, D. (2016). Validation of the Leap Motion Controller using markered motion capture technology. *Journal of Biomechanics*, *49*(9), 1742–1750.

[12] Kim, M. J., Naydanova, E., Hwang, B. Y., Mills, K. A., Anderson, W. S., & Salimpour, Y. (2020). Quantification of Parkinson's Disease Motor Symptoms: A Wireless Motion Sensing Approach. In *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS* (Vol. 2020-July, pp. 3658–3661). Institute of Electrical and Electronics Engineers Inc.

[13] Butt, A. H., Rovini, E., Dolciotti, C., De Petris, G., Bongioanni, P., Carboncini, M. C., & Cavallo, F. (2018). Objective and automatic classifcation of Parkinson disease with Leap Motion controller. *BioMedical Engineering Online*, *17*(1).

[14] Butt, A. H., Rovini, E., Dolciotti, C., Bongioanni, P., De Petris, G., & Cavallo, F. (2017). Leap motion evaluation for assessment of upper limb motor skills in Parkinson's disease. In *IEEE International Conference on Rehabilitation Robotics* (pp. 116–121). IEEE Computer Society.

[15] Vamsikrishna, K. M., Dogra, D. P., & Desarkar, M. S. (2016). Computer-Vision-Assisted Palm Rehabilitation With Supervised Learning. *IEEE Transactions on Biomedical Engineering*, *63*(5), 991–1001.

[16] Weiss Cohen, M., & Regazzoni, D. (2020). Hand rehabilitation assessment system using leap motion controller. *AI and Society*, *35*(3), 581–594.

[17] Ameur, S., Khalifa, A. B., & Bouhlel, M. S. (2017). A comprehensive leap motion database for hand gesture recognition. In *2016 7th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications, SETIT 2016* (pp. 514–519). Institute of Electrical and Electronics Engineers Inc.

[18] Marin, G., Dominio, F., & Zanuttigh, P. (2016). Hand gesture recognition with jointly calibrated Leap Motion and depth sensor. *Multimedia Tools and Applications*, *75*(22), 14991–15015.

[19] Jia, J., Tu, G., Deng, X., Zhao, C., & Yi, W. (2019). Real-time hand gestures system based on leap motion. In *Concurrency and Computation: Practice and Experience* (Vol. 31). John Wiley and Sons Ltd.

[20] Yang, Q., DIng, W., Zhou, X., Zhao, D., & Yan, S. (2020). Leap Motion Hand Gesture Recognition Based on Deep Neural Network. In *Proceedings of the 32nd Chinese Control and Decision Conference, CCDC 2020* (pp. 2089–2093). Institute of Electrical and Electronics Engineers Inc.

[21] Jian, C., Liu, X., & Zhang, M. (2022). RD-Hand: a real-time regression-based detector for dynamic hand gesture. *Applied Intelligence*, *52*(1), 417–428.

[22] Naguri, C. R., & Bunescu, R. C. (2017). Recognition of dynamic hand gestures from 3D motion data using LSTM and CNN architectures. In *Proceedings - 16th IEEE International Conference on Machine Learning and Applications, ICMLA 2017* (Vol. 2017-December, pp. 1130–1133). Institute of Electrical and Electronics Engineers Inc.

[23] Rossol, N., Cheng, I., & Basu, A. (2016). A Multisensor Technique for Gesture Recognition Through Intelligent Skeletal Pose Analysis. *IEEE Transactions on Human-Machine Systems*, *46*(3), 350–359.

[24] Li, X., Wan, K., Wen, R., & Hu, Y. (2018). Development of finger motion reconstruction system based on leap motion controller. In *CIVEMSA 2018 - 2018 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications, Proceedings*. Institute of Electrical and Electronics Engineers Inc.

[25] Avola, D., Bernardi, M., Cinque, L., Foresti, G. L., & Massaroni, C. (2019). Exploiting Recurrent Neural Networks and Leap Motion Controller for the Recognition of Sign Language and Semaphoric Hand Gestures. *IEEE Transactions on Multimedia*, *21*(1), 234–245.

[26] Chong, T. W., & Lee, B. G. (2018). American sign language recognition using leap motion controller with machine learning approach. *Sensors (Switzerland)*, *18*(10).

[27] Hisham, B., & Hamouda, A. (2017). Arabic static and dynamic gestures recognition using leap motion. *Journal of Computer Science*, *13*(8), 337–354.

[28] Cho, Y., Lee, A., Park, J., Ko, B., & Kim, N. (2018). Enhancement of gesture recognition for contactless interface using a personalized classifier in the operating room. *Computer Methods and Programs in Biomedicine*, *161*, 39–44.

[29] Sun, X., Byrns, S., Cheng, I., Zheng, B., & Basu, A. (2017). Smart Sensor-Based Motion Detection System for Hand Movement Training in Open Surgery. *Journal of Medical Systems*, *41*(2).

[30] https://github.com/jeffsp/matleap

[31] https://docs.ultraleap.com/tracking-api/group/group___structs.html

Acknowledgement

I wish to thank, Prof. Monteriù Andrea to follow me during these months, giving me the needed suggestions, and supporting my ideas to conclude my work in the best possible Also, I would thank Dott. Ing. Felicetti Riccardo and Dott. Ing. Iarlori Sabrina to help me every time that I needed. Moreover, the words that I could spend here are not enough to transmit how much you have given me during my thesis work both in a humanity and professional way.

Now, I desired to thank my FRIENDS, in the last two years there were a lot of changes in my life, some of them due to you. I laughed, I got angry, I cried with you and for you. Thank you to give me your best and to believe in me. Just I can say here, in black and white, is we are brothers of life and for life.

It's time to write to thank my little sister, Kira. Thank you to remain, independently on my side, thank you to hate me when was necessary, thank you to believe in me, and thank you to be my sister.

For my parents. You are the most important people in my life, you gave me every thing of your life for reach this "personal" achievement but I tell you now that is ours. Thank to scolded me when was needed, thank to give me your smiles, your hugs, and your dreams for me and Kira. This thesis is for you, and my future achievements will be your as well.
I Love you, my invincible parents.

Appendix

1. ----------

```
function function_plot(struttura_dati, inizio, frame, figura)
   for i = inizio : frame
      palm_position = struttura_dati(i).hands.palm.position;
      arm_position = struttura_dati(i).hands.arm.next_joint - palm_position;

      color_joint = {'r*', 'g*', 'c*', 'm*', 'k*'};
      color_joint_segment = {'r', 'g', 'c', 'm', 'k'};
      digit_name = {'Thumb', 'Index', 'Middle', 'Ring', 'Pinky'};
      if figura == 1
         figure,
      end
      % ARM Point
%        hold on,
      plot3(-arm_position(3), -arm_position(1), arm_position(2),'+g');
      text(-arm_position(3), -arm_position(1), arm_position(2), 'ARM');
      % PALM Point
%        hold on, plot3(-palm_position(3) + 1, -palm_position(1) + 1, palm_position(2) + 1,'b*');
%        text(-palm_position(3) + 1, -palm_position(1) + 1, palm_position(2) + 1, 'PALM');
      palm_position_zero = palm_position - palm_position;
      hold on, plot3(- palm_position_zero(3), - palm_position_zero(1), palm_position_zero(2),'b*');
      text( 0 , 0, 0, 'PALM');
      % digits
      for k = 1 : 5
         for j = 1 : 4
            if j < 4
```

```
            digit =  struttura_dati(i).hands.digits(k).bones(j).prev_joint - palm_position;

            digit_next =  struttura_dati(i).hands.digits(k).bones(j + 1).prev_joint - palm_position;

            if j < 3

              hold on, plot3(- [palm_position_zero(3);digit(3)], - [palm_position_zero(1);digit(1)],
[palm_position_zero(2);digit(2)],':', 'Color', [225 225 225]/255,'LineWidth',1.5);

              if j == 1

                hold on, plot3(- [arm_position(3);digit(3)], - [arm_position(1);digit(1)],
[arm_position(2);digit(2)],':', 'Color', [225 225 225]/255,'LineWidth',1.5);

              end

            end

            if j == 3 && k == 1

              hold on, plot3(- [palm_position_zero(3);digit(3)], - [palm_position_zero(1);digit(1)],
[palm_position_zero(2);digit(2)],':', 'Color', [225 225 225]/255,'LineWidth',1.5);

            end


          elseif j == 4

            digit =  struttura_dati(i).hands.digits(k).bones(j).next_joint - palm_position;

            hold on, plot3(-digit_next(3), - digit_next(1), digit_next(2), color_joint{k}, 'LineWidth', 3);

            text(-digit(3) + 1, - digit(1) + 1, digit(2) + 1, digit_name{k});

          end

          hold on, plot3(-digit(3), - digit(1), digit(2), color_joint{k}, 'LineWidth', 3);

          hold on, plot3(- [digit(3);digit_next(3)], - [digit(1);digit_next(1)], [digit(2);digit_next(2)],
color_joint_segment{k})

          if k > 1 && k < 5 && j == 2

            digit_near =  struttura_dati(i).hands.digits(k + 1).bones(j).prev_joint - palm_position;

            hold on, plot3(- [digit(3);digit_near(3)], - [digit(1);digit_near(1)], [digit(2);digit_near(2)], ':', 'Color',
[225 225 225]/255,'LineWidth',1.5)

          end

          if k == 1 && j == 3

            digit_near =  struttura_dati(i).hands.digits(k + 1).bones(j - 1).prev_joint - palm_position;

            hold on, plot3(- [digit(3);digit_near(3)], - [digit(1);digit_near(1)], [digit(2);digit_near(2)], ':', 'Color',
[225 225 225]/255,'LineWidth',1.5)

          end
```

```
        end
    end
    % Hand Normal
    hold on,
    p = struttura_dati(i).hands.palm.normal;
    plot3(-  [p(3);palm_position_zero(3)],  -[p(1);palm_position_zero(1)],  [p(2);palm_position_zero(2)],'m-
.','LineWidth',1.5);
    % Hand direction
    hold on,
    d = struttura_dati(i).hands.palm.direction;
    plot3(- [d(3); palm_position_zero(3)], -[d(1); palm_position_zero(1)], [d(2); palm_position_zero(2)],'c--
','LineWidth',1.5);


%       x_line = (0 : 5)';
%       y_line = (0 : 5)';
%       z_line = (0 : 5)';
%       vuoto = zeros(6,1);
%       hold on, plot3(z_line, vuoto, vuoto,'-.','LineWidth',1.5); text(10,0,0, 'Z axis')
%       hold on, plot3(vuoto, x_line, vuoto,'-.','LineWidth',1.5); text(0,10,0, 'X axis')
%       hold on, plot3(vuoto, vuoto, y_line,'-.','LineWidth',1.5); text(0,0,10, 'Y axis')
    I = [1, 0, 0]';
    J = [0, 1, 0]';
    K = [0, 0, 1]';
    u = 20;
    x_axis = u * [I, zeros(3,1),zeros(3,1)];
    y_axis = u *[zeros(3,1), J, zeros(3,1)];
    z_axis = u *[zeros(3,1), zeros(3,1), K];
    hold on,
    plot3(z_axis , x_axis, y_axis)
    text(10,0,0, 'Z axis')
    text(0,10,0, 'X axis')
```

```
        text(0,0,10, 'Y axis')
        xlabel('Z [mm]')
        ylabel('X [mm]')
        zlabel('Y [mm]')
        hold off
    end
end
```

---

2. ----------

```
function [grab_Thumb, grab_Index, grab_Middle, grab_Ring, grab_Pinky] = function_palm_digits(struttura_dati, frame)

   grabbing = zeros(4,5);

   i = frame;

   palm = struttura_dati(i).hands.palm.position;

   for k = 2 : 4

      for j = 1 : 5

         digit = struttura_dati(i).hands.digits(j).bones(k).prev_joint;

         grabbing(k-1,j) = norm(digit - palm);

         if j == 1 && k == 2

            grabbing(k-1,j) = 0;

         end

         if k == 4

            digit_next = struttura_dati(i).hands.digits(j).bones(k).next_joint;

            grabbing(k,j) = norm(digit_next - palm);

         end

      end

   end

   grab_Thumb = grabbing(:,1);

   grab_Index = grabbing(:,2);

   grab_Middle = grabbing(:,3);

   grab_Ring = grabbing(:,4);

   grab_Pinky = grabbing(:,5);

end
```

---

3. ----------

```
function [Thumb_Index, Thumb_Middle, Thumb_Ring, Thumb_Pinky] = function_thumb_other_digits(struttura_dati, frame)
    Thumb_Index = zeros(3,1); Thumb_Middle = zeros(3,1);
    Thumb_Ring = zeros(3,1); Thumb_Pinky = zeros(3,1);
    k = 1; i = frame;
    for o = 3 : 4
        Thumb = struttura_dati(i).hands.digits(1).bones(o).prev_joint;
        for j = 2 : 5
            digit = struttura_dati(i).hands.digits(j).bones(2).prev_joint;
            if j == 2
                % distance between thumb and Index
                Thumb_Index(k) = norm(digit - Thumb);
            end
            if j == 3
                % distance between thumb and Middle
                Thumb_Middle(k) = norm(digit - Thumb);
            end
            if j == 4
                % distance between thumb and Ring
                Thumb_Ring(k) = norm(digit - Thumb);
            end
            if j == 5
                % distance between thumb and Pinky
                Thumb_Pinky(k) = norm(digit - Thumb);
            end
        end
        k = k + 1;
    end
    thumb_next =  struttura_dati(i).hands.digits(1).bones(4).next_joint;
```

```
        for j = 2 : 5
            digit = struttura_dati(i).hands.digits(j).bones(4).next_joint;
            if j == 2
                % distance between thumb and Index
                Thumb_Index(k) = norm(digit - thumb_next);
            end
            if j == 3
                % distance between thumb and Middle
                Thumb_Middle(k) = norm(digit - thumb_next);
            end
            if j == 4
                % distance between thumb and Ring
                Thumb_Ring(k) = norm(digit - thumb_next);
            end
            if j == 5
                % distance between thumb and Pinky
                Thumb_Pinky(k) = norm(digit - thumb_next);
            end
        end
    end
    ------------------------------------------------
```

4. ----------

```
function [pitch] = pitch_digits(struttura_dati, frame)

   i = frame;
   pitch = struct;
   digit_name = {'Thumb', 'Index', 'Middle', 'Ring', 'Pinky'};

   for j = 1 : 5
      for p = 1 : 3

         digit_ = struttura_dati(i).hands.digits(j).bones(p).prev_joint;
         digit_next_prev = struttura_dati(i).hands.digits(j).bones(p).next_joint;

         digit_next_ = struttura_dati(i).hands.digits(j).bones(p + 1).prev_joint;
         digit_next_next = struttura_dati(i).hands.digits(j).bones(p + 1).next_joint;

         digit_next = digit_next_next - digit_next_prev;
         digit = digit_next_ - digit_;

         pitch.(char(digit_name(j)))(p) = acos( dot(digit_next, digit)/(norm(digit_next) * norm(digit))) * 180/pi;
      end
   end
end
```

---

5. ----------

```
function [yaw] = yaw_digit(struttura_dati, frame)

    i = frame;
    yaw = struct;
    digit_name = {'Thumb_Index', 'Index_Middle', 'Middle_Ring', 'Ring_Pinky'};
    palm = struttura_dati(i).hands.palm.position;

    for j = 1 : 4

        digit = struttura_dati(i).hands.digits(j).bones(4).next_joint - palm;
        digit_next = struttura_dati(i).hands.digits(j + 1).bones(4).next_joint - palm;

        yaw.(char(digit_name(j))) =  acos( dot(digit, digit_next)/(norm(digit) * norm(digit_next))) * 180/pi;
    end

end
```

---

6. ----------

```
function [Valori_filtrati] = Filter_Function(valori)
% Filter Design specific:
n = 2; % o 3;
fc = 6;
fs = 120;
[b, a] = butter(n, fc/(fs/2), 'low');

Type = fields(valori);

for i = 1 : length(Type)
   Signal = fields(valori.(char(Type(i))));
   for j = 1 : length(Signal)
      Valori_filtrati.(char(Type(i))).(char(Signal(j))) = filtfilt(b, a, valori.(char(Type(i))).(char(Signal(j))));
   end
end
end
```

---

7. ----------

```
function [Area, derivative] = Computation_area(valori)
Type = fields(valori);
Area = struct;
derivative = struct;

for i = 1 : length(Type)
   Signals = fields(valori.(char(Type(i))));

   for n = 1 : length(Signals)
      derivative.(char(Type(i))).(char(Signals(n))) = diff(valori.(char(Type(i))).(char(Signals(n))))*120;
      s = derivative.(char(Type(i))).(char(Signals(n)));
      [~, c] = size(s);
      k = 1;
      l = 1;
      area = 0;
      while k < length(s(:,c))
         area = area + s(k,c);
         if s(k + 1,c) > 0 && area < 0
            Area.(char(Type(i))).(char(Signals(n)))(l,:) = [area, k];
            area = 0;
            l = l + 1;
         elseif s(k + 1,c) < 0 && area > 0
            Area.(char(Type(i))).(char(Signals(n)))(l,:) = [area, k];
            area = 0;
            l = l + 1;
         end
         k = k + 1;
      end
```

```
    end
end
end
```

---

8. ----------

```
function [Maxima, Minima] = Finding_Max_Min(valori, aree) % valori =
Maxima = struct;
Minima = struct;
campo = fields(valori);
for i = 1 : length(campo)
   campo_2 = fields(valori.(char(campo(i))));
   for j = 1 : length(campo_2)
      if i == 1
         s = valori.(char(campo(i))).(char(campo_2(j)))(:,end);
      else
         s = valori.(char(campo(i))).(char(campo_2(j)));
      end
      area_value = aree.(char(campo(i))).(char(campo_2(j)))(:,1);
      peaks = aree.(char(campo(i))).(char(campo_2(j)))(:,2);

      peak = [1; peaks; length(s)]; % X
      amplitude = s(peak); % Y

      m = zeros(length(amplitude)-1,1);
      q = zeros(length(amplitude)-1,1);

      for k = 2 : length(peak)
         m(k-1) = (amplitude(k)-amplitude(k-1))/(peak(k)-peak(k-1));
         q(k-1) = amplitude(k-1) - m(k-1)*peak(k-1);
      end

      area = 0; k1 = 1; k2 = 1; k3 = 1; k4 = 1;
      max_ = []; max_k = []; min_ = []; min_k = [];
```

```
for l = 1 : length(area_value)
    area = area + area_value(l);
    if area > 0
        if abs(m(l+1)) < 0.5
            if abs(amplitude(l+1)-amplitude(l)) > abs(amplitude(l+1)-amplitude(l))/20
                max_(k1,1) = peak(l+1);
                max_(k1,2) = amplitude(l+1);
                k1 = k1 + 1;
            end
        end
        if abs(m(l+1)) > 0.5 && ~isempty(max_)
            [max_k(k3,1), o] = max(s(max_(:,1)));
            max_k(k3,2) = max_(o,1);
            Maxima.(char(campo(i))).(char(campo_2(j))) = max_k;
            k3 = k3 + 1;
            area = 0;
            max_ = [];
            k1 = 1;
        end
    end
    if area < 0
        if abs(m(l+1)) < 0.5
            if abs(amplitude(l+1)-amplitude(l)) > abs(amplitude(l+1)-amplitude(l))/20
                min_(k2,1) = peak(l+1);
                min_(k2,2) = amplitude(l+1);
                k2 = k2 + 1;
            end
        end
        if abs(m(l+1)) > 0.5 && ~isempty(min_)
```

```
                [min_k(k4,1), o] = min(s(min_(:,1)));
                min_k(k4,2) = min_(o,1);

                Minima.(char(campo(i))).(char(campo_2(j))) = min_k;

                k4 = k4 + 1;
                area = 0;
                min_ = [];
                k2 = 1;
            end
        end
    end
  end
end
```

---

9. ----------

```
figure("Position",[50, 50, 1500, 500])

for i = 1 : 5 : length(frame_list)
subplot(1,2,1)
palm = frame_list(i).hands.palm.position;
arm = frame_list(i).hands.arm.next_joint - palm;
color_joint = {'r*', 'g*', 'c*', 'm*', 'k*'};
color_joint_segment = {'r', 'g', 'c', 'm', 'k'};
digit_name = {'Thumb', 'Index', 'Middle', 'Ring', 'Pinky'};

%%%%%%%%%%%%%%%%%%%%%
%%%%%% ARM Point %%%%%%
%%%%%%%%%%%%%%%%%%%%%

plot3(-arm(3), -arm(1), arm(2), '+', 'LineWidth', 2);
axis([-100 100 -100 100 -100 100])

%%%%%%%%%%%%%%%%%%%%%
%%%%%% PALM Point %%%%%%
%%%%%%%%%%%%%%%%%%%%%

palm_zero = palm - palm;
hold on,
plot3( -palm_zero(3), -palm_zero(1), palm_zero(2), 'bo', 'LineWidth', 2);

%%%%%%%%%%%%%%%%%%%%%
%%%%%% Digits Point %%%%%%
%%%%%%%%%%%%%%%%%%%%%
```

```
for k = 1 : 5
   for j = 1 : 4
      if j < 4
         digit =  frame_list(i).hands.digits(k).bones(j).prev_joint - palm;
         digit_next =  frame_list(i).hands.digits(k).bones(j + 1).prev_joint - palm;

         if k > 1 && k < 5 && j == 2
            digit_near =  frame_list(i).hands.digits(k + 1).bones(j).prev_joint - palm;
            hold on, plot3( -[digit(3); digit_near(3)], - [digit(1); digit_near(1)], [digit(2); digit_near(2)], ':', 'Color',
[225 225 225]/255, 'LineWidth', 1.5)
         end
         if k == 1 && j == 3
            digit_near =  frame_list(i).hands.digits(k + 1).bones(j - 1).prev_joint - palm;
            hold on, plot3( -[digit(3); digit_near(3)], -[digit(1); digit_near(1)], [digit(2); digit_near(2)], ':', 'Color',
[225 225 225]/255, 'LineWidth', 1.5)
         end
         if j < 3
            hold on, plot3( -[palm_zero(3); digit(3)], -[palm_zero(1); digit(1)], [palm_zero(2); digit(2)], ':', 'Color',
[225 225 225]/255, 'LineWidth', 1.5);
            if j == 1
               hold on, plot3( -[arm(3); digit(3)], -[arm(1); digit(1)], [arm(2); digit(2)], ':', 'Color', [225 225
225]/255, 'LineWidth', 1.5);
            end
         end
         if j == 3 && k == 1
            hold on,
            plot3( -[palm_zero(3); digit(3)], -[palm_zero(1); digit(1)], [palm_zero(2); digit(2)], ':', 'Color', [225
225 225]/255, 'LineWidth', 1.5);
         end
      elseif j == 4
         digit =  frame_list(i).hands.digits(k).bones(j).next_joint - palm;
         hold on, plot3( -digit_next(3), -digit_next(1), digit_next(2), color_joint{k}, 'LineWidth', 4);
```

```
        text( -digit(3) - 5, -digit(1) - 5, digit(2) + 5, digit_name{k});

    end

    hold on, plot3( -digit(3), -digit(1), digit(2), color_joint{k}, 'LineWidth', 4);

    hold on, plot3( -[digit(3); digit_next(3)], -[digit(1); digit_next(1)], [digit(2); digit_next(2)],
color_joint_segment{k}, 'LineWidth', 2);


  end

end


I = [1, 0, 0]';

J = [0, 1, 0]';

K = [0, 0, 1]';


u = 20;

x_axis = u * [I, zeros(3,1),zeros(3,1)];

y_axis = u *[zeros(3,1), J, zeros(3,1)];

z_axis = u *[zeros(3,1), zeros(3,1), K];


hold on,

plot3(z_axis , x_axis, y_axis)

text(10,0,0, 'Z')

text(0,10,0, 'X')

text(0,0,10, 'Y')

xlabel('Z [mm]')

ylabel('X [mm]')

zlabel('Y [mm]')

hold off

set(gca, 'CameraPosition', [552.442694778547,820.495128834866,326.4101615137753]);

legend('ARM', 'PALM')


drawnow limitrate
```

```
% okok


% okok

%view([0, 1, 1])

%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

%%%%%%%%%%%%%%%% displacement graph %%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

displacement = struct;

digit_name = {'Thumb', 'Index', 'Middle', 'Ring', 'Pinky'};

for j  = 1 : 5

    displacement.(char(digit_name(j))) = [];

    for p = 1 : length(frame_list)

        displacement.(char(digit_name(j)))                    =                    [displacement.(char(digit_name(j)));
results.distance(p).grab.(char(digit_name(j)))'];

    end

end


subplot(5,2,2)

plot(displacement.Thumb(1:i,2:4))

axis([0 length(displacement.Thumb(:,4)) 0 max(displacement.Thumb(:,4))])

title('Thumb')

subplot(5,2,4)

plot(displacement.Index(1:i,2:4))

axis([0 length(displacement.Index(:,4)) 0 max(displacement.Index(:,4))])

title('Index')

subplot(5,2,6)

plot(displacement.Middle(1:i,2:4))
```

```
axis([0 length(displacement.Middle(:,4)) 0 max(displacement.Middle(:,4))])
title('Middle')
subplot(5,2,8)
plot(displacement.Ring(1:i,2:4))
axis([0 length(displacement.Ring(:,4)) 0 max(displacement.Ring(:,4))])
title('Ring')
subplot(5,2,10)
plot(displacement.Pinky(1:i,2:4))
axis([0 length(displacement.Pinky(:,4)) 0 max(displacement.Pinky(:,4))])
title('Pinky')
l = 1;
end
drawnow
--------------------------------------------------
```