Master Thesis
Machine Learning For EMG Data

Martin Colot

2021

Part I
Introduction

Part II
State of the art

0.1 EMG

- What is an EMG signal
- EMG and EEG
- EMG and ENG
  https://pubmed.ncbi.nlm.nih.gov/33091891/
  https://pubmed.ncbi.nlm.nih.gov/29498358/
- sEMG and iEMG sensor
- high density EMG
  https://www.sciencedirect.com/science/article/abs/pii/S1746809419302186
  https://pubmed.ncbi.nlm.nih.gov/22180516/

0.2 Myoelectric hand prosthesis

- Purpose
- Existing brands
  https://ieeexplore.ieee.org/document/8733629
  https://app.dimensions.ai/details/publication/pub.1112252996?and_facet_journal=jour.1041772
- Difficulties
  - limitation of non-invasive sensor
  - Lack of EMG data for amputees
  - Mirrored billateral training
    https://pubmed.ncbi.nlm.nih.gov/22180516/
    https://pubmed.ncbi.nlm.nih.gov/22006428/

0.3 Hand gesture prediction

- Applications (prosthetic, VR)
- Classification and regression

0.3.1 Preprocessing and feature selection

0.3.2 Gesture classification

- Classification techniques
  https://journals.physiology.org/doi/pdf/10.1152/jn.00555.2014
  https://www.nature.com/articles/s41551-016-0025
- Limitations

0.3.3 Movement regression (joint angle classification)

- Needed for a more natural feeling
- 27 degrees of freedom of the hand
- Regression techniques
  https://jneuroengrehab.biomedcentral.com/articles/10.1186/1743-0003-11-122
  https://www.hindawi.com/journals/isrn/2012/604314/
  https://pubmed.ncbi.nlm.nih.gov/22180516/

1 Existing data set of synchronized EMG and hand gesture data

Most of the current work concerning hand gesture prediction from sEMG signal is concentrated on gesutre classification. That is why data sets containing simultaneously sEMG signals and fingers kinematics, in the aim of doing gesture regression, are not so numerous. Moreover, there exist no state of the art benchmark for the data collection protocol which could enable to easily create such data sets.

We present, in this section, two data sets containing synchronized sEMG signals and hand kinematics [4, 5] as well a study presenting its protocol for such data collection (without providing its data set) [7].

1.0.0.1 The NinaPro data set [4] Published in 2014 in *Nature Scientific Data*, this data set is composed of data acquired from 67 intact subjects and 11 who had 1 missing arm (in the aim of using the data set for hand prosthesis control). These subjects were asked to perform 4 kinds of exercises for a total of 61 different gestures (plus the resting gesture). The researchers used 12 sEMG electrodes and a CyberGlove to estimate the hand kinematics.

This data set has been cited in multiple similar works [5, 8]. It is however not perfect. The KIN-MUS UJI data set (presented below) shows three weaknesses that need to be corrected in order to create a reliable and reproducible benchmark.

1. The performed gestures do not correspond to real life movements (ADL)
2. The representation of the hand kinematics data is not representing anatomical angles
3. No indication on the exact sEMG location

1.0.0.2 Continuous and simultaneous estimation of finger kinematics using inputs from an EMG-to-muscle activation model [7]

1.0.0.3 The KIN-MUS UJI data set [5] Also published in *Nature Scientific Data* but in 2019, this data set aims at correcting weaknesses of the *NinaPro* data set and other previews data collection protocols. In particular, it gives more precise informations on the sEMG sensor locations, its gestures are based on ALD and the hand kinematics are represented using a standardisation of the anatomical angles of the hand given by the International Society of Biomechanics (ISB) [10].

1.1 Experimental setup

1.1.1 EMG sensor

1.1.2 Hand pose Estimation

1.2 sEMG Electrodes placement

1.2.1 Muscular activity zone identification by palpation

1.2.2 Pre-identified zones giving all the muscular activity of the forearm

1.2.3 Arm band

1.3 Gestures performed

1.3.1 Single finger motions

1.3.2 Activities of daily living (ADL)

1.3.3 Sign language

1.3.4 Irregular moves

1.3.5 Maximum voluntary contraction (MVC)

1.4 Hand position data representation

1.5 Synchronization

2 Motion capture technologies

In order to train a regression model of the hand kinematics based on EMG signal, we need to provide a ground truth describing the evolution of the hand gesture synchronized with this signal. Two kinds of devices are usually used to record this information: a CyberGlove or a camera motion tracking system.

2.1 CyberGlove

The CyberGlove made by the company CyberGlove Systems (`http://www.cyberglovesystems.com/`) is a motion capture device composed of a glove with multiple strain gauges which capture the motion of the hand. Its latest currently available version, the CyberGlove III, can capture up to 22 joint-angles with a resolution lower than 1 degree. The CyberGlove II was used for the creation of the Ninapro datasets [4, 8] and the KIN-MUS UJI Dataset [5].

Even if this device is really easy to use and provides an accurate reconstruction of the hand joint-angles, it does not allow to easily reproduce the data collection experiment as its price can quickly reach several tens of thousands of euros and it is not easily found in every laboratory. Some people have shared instructions on how to build an homemade CyberGlove for 40 dollars [3] but there is no guarantee on the performance that it can provide.

2.2 3D motion camera tracking

This technique is usually used for cinematographic special effects. It is composed of 2 elements: an ensemble of markers placed on the tracked object (consisting of small white balls) and a set of fixed cameras located

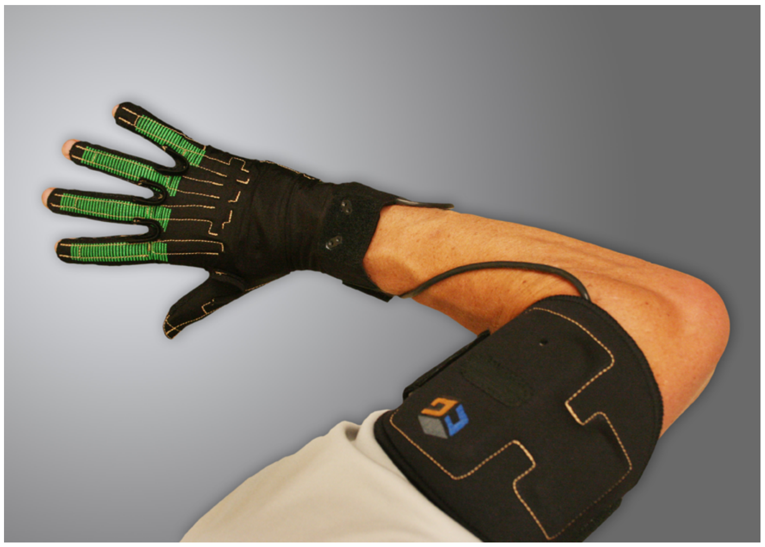

Figure 1: Picture of the CyberGlove III from the CyberGlove System website (`http://www.cyberglovesystems.com/cyberglove-iii/`)

around the subject that each film it from a different point of view. The location of each marker in the 3D space can be computed from its position each cameras vision.

To record the finger kinematics, 22 markers need to be placed as shown in the following figure.

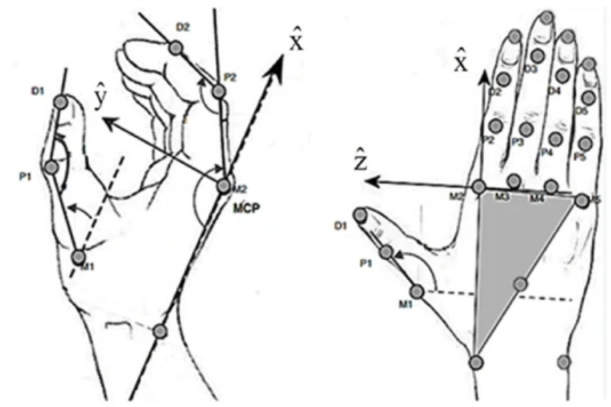

Figure 2: Positions of the 22 markers used to track the fingers kinematics [7]

This device has a precision of less than 0.5mm [7]. However, some study say that it is not reliable enough for finger tracking as their movement is too precise [5].

2.3 Oculus Quest

The Oculus Quest is a virtual reality headset developed by Facebook (`https://www.oculus.com/quest/`). It has 4 infra-red mounted cameras which are oriented to film the user hands from different angles. The headset is then able to reconstruct in real time a pair of 3D hands which accurately match the user gestures. The pose reconstruction technique is similar different from a conventional motion tracking system as it does not need any marker on the subject. It relies on a deep neural network which only takes to camera vision as input and uses an architechture similar to PoseNets; a model from TensorFlow used for human posture estimation (`https://www.tensorflow.org/lite/examples/pose_estimation/overview`) [6, 2].

2.3.1 Advantages of the Oculus Quest device

- It is cheap compared to the others (399 euros)
- It can estimate 26 degrees of freedom for each hand [2]
- The neural network is specifically trained to estimate hand gesture which allows more accurate estimation than conventional motion tracking `https://developer.oculus.com/documentation/unity/unity-utilities-overview/`)
- It is fast to setup for data collection
- It can be controlled remotely using a USB-c cable
- A programming library is available to easily use it (OVR library for the Unity game engine
- The programming library provides additional pieces of information like the quality of the estimation for each finger and a pinching boolean that tells if a given finger is touching the thumb of the same hand.

2.3.2 Disadvantages of the Oculus Quest device

- The precision of the estimation is not the same for all fingers
- If a single joint is incorrectly predicted, it can cause big prediction errors [6]
- If the background color is similar to the skin color, the estimation becomes less accurate [6]
- There is no documentation on the accuracy of the estimation. We can only rely on testers feeling about it [1]
- Its sampling rate is quite low (50 or 60Hz depending on the electrical frequency in the country)

2.3.3 Unity

In order to collect the data of the gesture estimation from the Oculus Quest, we need to use the Unity game engine (`https://unity.com/`). It is a really easy to use framework that enables to build 3D scenes and can be programmed in *C#*. It has a lot of documentation, receives regular update since 2005 and has a huge community of developers. There also exist an asset store which enables to download lots of assets to put in the built software.[9]

2.3.3.1 The OVR library To use the Oculus quest components from a Unity project, we need to use the Unity OVR library (`https://developer.oculus.com/documentation/unity/unity-utilities-overview/`) which provides all the classes that allows to use the hand tracking option of the headset and to collect its data. When it is setup, the user is able to see its hands in the virtual reality and to program can get, at each frame (50 frames per seconds), the rotation of each bone of the 2 hands as well as the additional information provided by the library.

The information that can be collected for each hand and at each frame from the headset is:

1. The position and rotation of the hand in the 3D space
2. The 3D rotation of 17 bones of the hand
3. A boolean telling if the estimation could be processed (false means that the neural network failed to predict the pose)
4. A boolean per finger indicating if the quality of the estimation of its pose is high or low

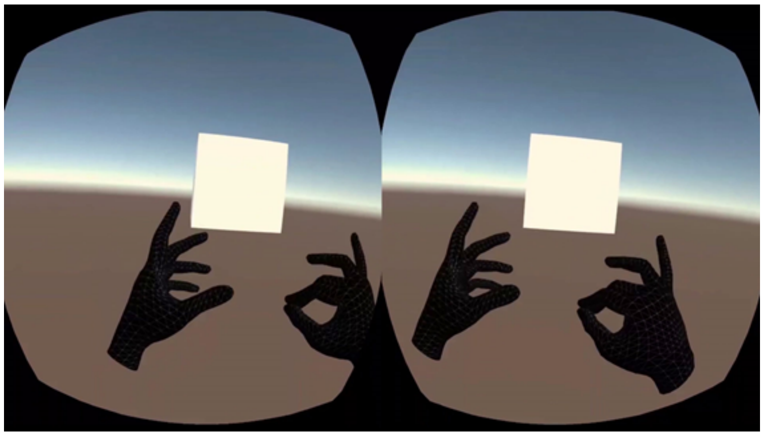
Figure 3: View of the estimated hands from the Oculus quest Virtual Reality headset

5. A boolean per finger, except the thumb, indicating if it is touching the thumb of the same hand (pinching)

The device is also able to compute the UTC timestamps of each frame which can b used for synchronization.

2.3.3.2 The Hand Tracking Gesture Recorder library The OVR library can be completed with a hand gesture data collection library (`https://github.com/jorgejgnz/HandTrackingGestureRecorder`). Made by Jorge Juan González (HCI Researcher at I3A (UCLM)), it enables to record a pose of the hand using the Oculus Quest so that that, later, the program produces a trigger when the user does the same pose. It is possible to save multiple gestures in a file and load them when starting the program on the headset.

This library takes as input all the bones of the hand and tries, based on their relative position to the wrist, to find the pose, among the recorded ones, that is the more probable to be the current pose. If no pose has a high enough probability, it does not output any prediction.

Part III

Realisation: Creation of a synchronized EMG/EEG/hand-gesture data set

3 Experimental setup

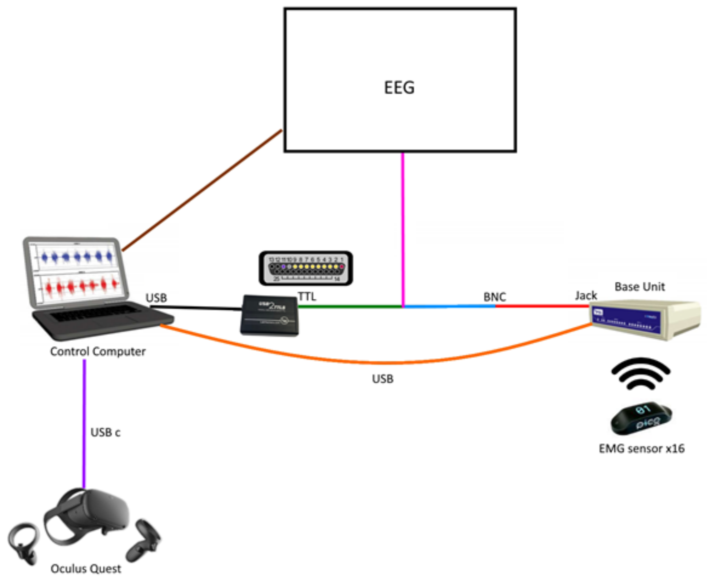

Figure 4: Diagram of the experimental setup for data collection

4 Hand gesture data collection from the Oculus Quest

In this section, we describe the development of a Unity project that can be executed on the Oculus Quest headset to collect hand motion data. The first part of this software is the setup of the OVR library in a new Unity 3D project. When this project is built in an APK file and loaded on the Oculus Quest, it can already display the user hands using motion tracking.

4.1 Data collection

A $C\#$ script called $dataSaving.cs$ is placed in the Unity scene to record all the information of the motion tracking. This script has a function $FixedUpdate()$ which is called by the engine with a regular period (set to a frequency of 50Hz to match the motion tracking frequency). This function has access to the scripts $OVRSkeleton$ and $OVRHand$ of both hands which contain the information on a current hand gesture. It can also get the variable $System.DateTime.UtcNow.Ticks$ to get the UTC of the frame, which will be later used for synchronization.

All the collected data is appended in a csv file (with ";" separator) where each line represent a single frame. A description of this file is given below.

Column index	data
1	UTC
2	Boolean telling if the whole data of the left hand is valid
3	Position (x, y, z) and rotation (x, y, z) of the left hand in the 3D space
4 - 22	Rotation (x, y, z) of 19 bones of the left hand
23 - 26	Boolean telling if the fingers 2 to 5 of the left hand are touching the thumb of the left hand (pinching action)
27 - 31	Boolean telling if the confidence in the quality of the pose estimation of the finger 1 to 5 is high or low
32 - 61	Same as 2 - 31 but for the right hand
62	Name of the gesture recognized by the library *Hand Tracking Gesture Recorder* (`https://github.com/jorgejgnz/HandTrackingGestureRecorder`) among those recorded for the right hand

Table 1: Caption

The indexes of the fingers are:

1. Thumb
2. Index
3. Middle
4. Ring
5. Pinky

The indexed of each finger bone can be found in the documentation of the OVR library (`https://developer.oculus.com/documentation/unity/unity-handtracking/`) and is described in figure 5 and shown in figure 6.

```
Hand_WristRoot   = Hand_Start + 0 // root frame of the hand, where the wrist is located
Hand_ForearmStub = Hand_Start + 1 // frame for user's forearm
Hand_Thumb0      = Hand_Start + 2 // thumb trapezium bone
Hand_Thumb1      = Hand_Start + 3 // thumb metacarpal bone
Hand_Thumb2      = Hand_Start + 4 // thumb proximal phalange bone
Hand_Thumb3      = Hand_Start + 5 // thumb distal phalange bone
Hand_Index1      = Hand_Start + 6 // index proximal phalange bone
Hand_Index2      = Hand_Start + 7 // index intermediate phalange bone
Hand_Index3      = Hand_Start + 8 // index distal phalange bone
Hand_Middle1     = Hand_Start + 9 // middle proximal phalange bone
Hand_Middle2     = Hand_Start + 10 // middle intermediate phalange bone
Hand_Middle3     = Hand_Start + 11 // middle distal phalange bone
Hand_Ring1       = Hand_Start + 12 // ring proximal phalange bone
Hand_Ring2       = Hand_Start + 13 // ring intermediate phalange bone
Hand_Ring3       = Hand_Start + 14 // ring distal phalange bone
Hand_Pinky0      = Hand_Start + 15 // pinky metacarpal bone
Hand_Pinky1      = Hand_Start + 16 // pinky proximal phalange bone
Hand_Pinky2      = Hand_Start + 17 // pinky intermediate phalange bone
Hand_Pinky3      = Hand_Start + 18 // pinky distal phalange bone
```

Figure 5: Indexes of the ones of the OVR hand as given by the OVR library

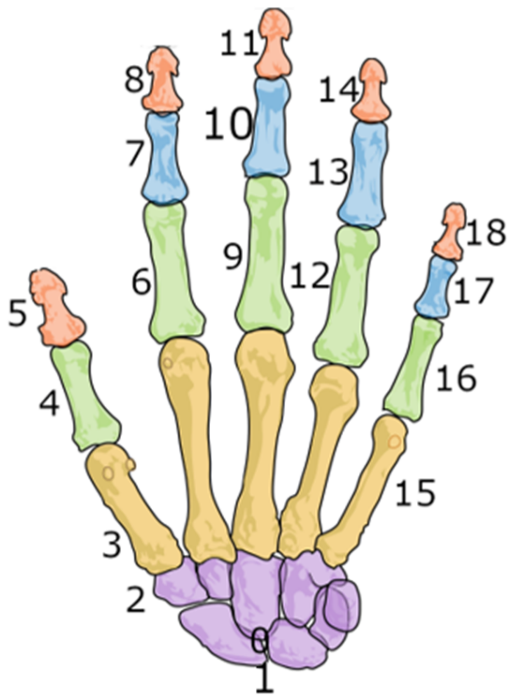

Figure 6: Graphical representation of the indexes of the ones of the OVR hand as given by the OVR library (`https://www.reddit.com/r/OculusQuest/comments/edkp9i/visual_reference_for_hand_tracking_bone_ids/`)

This is saved in a local file on the Oculus Quest and can be exported to an external computer after the recording using an USB-c link.

4.2 Remote control

The experimental setup is made so that the subject does not have to do anything else than the recorded hand gestures. In order to do that, a remote control interface was implemented. It enables to send messages to the Unity program running on the Oculus Quest from an external computer using a USB-c link. Theses messages take the form of files that are pushed in a particular directory, and with a name giving to the action to perform, using *Android Debug Bridge* (ADB: `https://developer.android.com/studio/command-line/adb`).

A python script called *remoteControl.py* was implemented to give an interface to remotely control the Oculus Quest behaviour. The actions that can be done are

- Start and stop the recording of the hand motion. A sphere in the 3D Unity scene, seen from the Oculus Quest, changes color to indicate if the recording is activated (red=off, green=on). Stopping the recording also pulls the record file on the computer running this script.
- Reset the recording
- Save the current gesture of the right hand for the Hand Tracking Gesture Recorder library
- Export the saved right hand gesture in a file (local on the Oculus Quest)
- Import the right hand gesture from a file (local on the external computer)
- Show the next instruction to the user (described later in the document)

4.3 Interface

5 Data synchronization

An important point of the dataset creation is the synchronization of the hand kinematics with the EMG and EEG signals. To perform this synchronization, we make three assumptions on the setup.

1. The recording of the EMG and EEG can be started using a trigger signal with a known delay
2. The timestamps given by the EMG, EEG
3. The UTC of the external computer and those of the Oculus Quest are synchronized

The 2 first assumptions are relying on the official user manuals of the hardware. We know that the Cometa EMG has a starting delay of 14ms when using a triggered start. The delay for the EEG is [TO FIND] . The EEG cannot be started with a trigger but it can include the trigger timestamp in its recorded data. So, we just have to cut the data before the first trigger.

To verify the third assumption, we can use the software *SideQuest* (`https://sidequestvr.com/`) that enables to stream the screen of the Oculus Quest on a computer using a USB-c cable (called Oculus Link). By doing so, we find a shift of approximately 22ms in average with a standart deviation around 100ms between the computers UTC and the Oculus Quests UTC. This time lag corresponds to the known delay of the SideQuest streaming (`https://www.mdpi.com/2073-431X/9/4/92`) which allows us to conclude that they are synchronized.

The algorithm to synchronize the data from the external computer is the following:

1. Start the hand motion recording (which takes unknown time) and EEG recording
2. Wait for the Oculus Quest to send a feedback that its recording has started
3. Save the current UTC
4. Send a trigger start signal on the EMG recorder and the EEG recorder
5. Collect the data until the end of the recording
6. stop all recording (they might not end exactly at the same time but the timestamps in each part of the records removes the need for synchronized end of record)
7. Cut the start of the hand kinematics recording at the UTC of the starting of the EMG/EEG recording
8. Convert the UTC in the hand kinematic file to timestamps in ms starting from 0 (0 corresponds to the UTC saved at the beginning of the recording but there might not be a recorded frame at this exact moment)
9. Shift the EMG recording by 14ms
10. Shift the EEG recording by [TO FIND]
11. Cut the beginning of the EEG recording (before the first trigger)
12. place everything in a common folder

6 Data collection protocol

6.1 Placement of the sEMG electrodes

6.2 Gestures to perform

Part IV
Conclusion and future work

References

[1] How accurate is oculus quest 2 hand-tracking feature? 2 2021. `https://www.youtube.com/watch?v=g8fGShHy3MA&ab_channel=SpookyFairy`.

[2] Facebook AI. Using deep neural networks for accurate hand-tracking on oculus quest. 9 2019.

[3] aloishis89. Build a $30k cyberglove for $40. https://www.instructables.com/Build-a-30k-CyberGlove-for-40-Submitted-by-Bay/.

[4] Castellini Claudio Caputo Barbara Hager Anne-Gabrielle Mittaz Elsig Simone-Giatsidis Giorgio Bassetto Franco Müller Henning Atzori Manfredo, Gijsberts Arjan. Electromyography data for non-invasive naturally-controlled robotic hand prostheses. *Scientific Data*, 12 2014.

[5] Sancho-Bru Joaquín L.-Gracia-Ibáñez Verónica Roda-Sales Alba Jarque-Bou Néstor J., Vergara Margarita. A calibrated database of kinematics and emg of the forearm and hand during activities of daily living. *Scientific Data*, 11 2019.

[6] Ritesh Kanjee. How oculus uses ai for hand tracking. 11 2019.

[7] Jimson G. Ngeo, Tomoya Tamei, and Tomohiro Shibata. Continuous and simultaneous estimation of finger kinematics using inputs from an emg-to-muscle activation model. *Journal of NeuroEngineering and Rehabilitation*, 11(1):122, Aug 2014.

[8] Stefano Pizzolato, Luca Tagliapietra, Matteo Cognolato, Monica Reggiani, Henning Müller, and Manfredo Atzori. Comparison of six electromyography acquisition setups on hand movement classification tasks. *PLOS ONE*, 12:e0186132, 10 2017.

[9] Wikipedia. Unity (game engine). 4 2021. `{https://en.wikipedia.org/w/index.php?title`Unity$_(game_engine)action = history$.

[10] Ge Wu, Frans C.T. van der Helm, H.E.J. (DirkJan) Veeger, Mohsen Makhsous, Peter Van Roy, Carolyn Anglin, Jochem Nagels, Andrew R. Karduna, Kevin McQuade, Xuguang Wang, Frederick W. Werner, and Bryan Buchholz. Isb recommendation on definitions of joint coordinate systems of various joints for the reporting of human joint motion—part ii: shoulder, elbow, wrist and hand. *Journal of Biomechanics*, 38(5):981–992, 2005.

[]