# A novel Learning Database for Sign Language Animation Synthesis from Quaternion estimations

Víctor Ubieto Nogales

**Abstract**

Avatar synthesis is one of the most important and challenging tasks when it comes to sign language synthesis. The scarce useful data in the field and its multidisciplinary requirements make this task very challenging and without many solutions. In this project we will present a system to generate sign animations from sign language videos. In particular, a novel automatic system to generate a dataset will be proposed and implemented. This approach is divided in two steps, a pose estimation, and a method to create the animations in a BVH (Biovision Hierarchy) file. Unlike the others, the presented approach estimates the rotations of each joint in quaternions instead of estimating the 3D absolute position of the joints, which are directly used to represent virtual animations. Finally, an evaluation of different state of the art approaches of generating realistic sign language animations will also be presented and compared to the presented method, both qualitatively and quantitatively.

**Index Terms**

Animation, Sign Language Synthesis, Sign Language Recognition, Sign Language Avatars, Motion Capture, Quaternions

## I. Introduction

### A. Problem Definition

SIGN Languages (SLs) are the primary means of communication for deaf people all around the world. These languages have their own syntax and differ drastically from oral languages.

In this Master Thesis we focus on producing SL using animated 3D synthetic characters, called signing avatars or virtual signers. In general, 3D synthetic characters used in conversation are called embodied conversational agents (ECAs). Kacorri's PhD thesis [1] cites several reasons to prefer animated sign language avatars over videos of human signers for presenting information: "It is often prohibitively expensive to re-film a human performing sign language for information that is frequently updated, thus leading to out-of-date information. Automatically synthesised animations allow for frequent updating as well as presenting content that is dynamically generated from a database query. Assembling video clips of individual signs together into sentences does not produce high-quality results. Animation offers additional flexibility through customised signing speed/style, avatar appearance, and view perspective. It preserves author anonymity through the use of avatars and enables the collaboration between multiple authors for scripting a message in sign language." ([1], pages 1-2). Another significant advantage is that they could be interactive, facilitating conversation.

Using virtual signers for translation is controversial for Deaf and Hard of Hearing (DHH) communities, as illustrated by the joint declaration of WDF and WASLI on signing avatars [1]. They have always felt alienated from new changes in society, and this feeling was increased with the large amount of new emerging technologies. This has made them reluctant to accept the low-cost work that they usually receive, especially in virtual interpreters. A very recent paper by Quandt et al. [2] discusses different aspects of the perception of avatars by DHH users, which hint at the very nuanced different dimensions of the issue.

Nevertheless, a key reason for the insufficient quality of signing avatars is the lack of sign language data, which is required when working on machine learning or deep learning techniques, Indeed, Bragg et al. [3] recent interdisciplinary survey and perspective on SL research and challenges, explicitly mentions the "scarcity" of public motion-capture datasets. The paper stresses that developing an interactive system for SLs requires expertise in a wide range of fields, including computer vision, computer graphics, natural language processing, human-computer interaction, linguistics, and Deaf culture. In particular, when developing a flexible interactive system for SL translation three different components would be necessary: a recogniser of the sign, a translator into a symbolic representation of the signed content, and the synthesis of a virtual avatar from this symbolic representation. The task is thus very challenging. The research challenges identified by Bragg et al. [3] with respect to **Avatars and Computer Graphics** are: *Uncanny Valley*, *Realistic Transitions*, *Modeling Modulations*, *Finding Model Holes* and the already mentioned *Public Motion-Capture Datasets*.

Author: Víctor Ubieto Nogales, victoremilio.ubieto@upf.edu
Advisor 1: Coloma Ballester, Information and Communication Technologies, UPF
Advisor 2: Gloria Haro, Information and Communication Technologies, UPF
Advisor 3: Josep Blat, Information and Communication Technologies, UPF
Thesis dissertation submitted: September 2022
[1]https://wfdeaf.org/news/wfd-wasli-issue-statement-signing-avatars/, retrieved Aug18 2022

### B. Thesis Objectives

In this master thesis we will present a process of generating an avatar animation from a SL video. This is an end-to-end system, which can be useful to create a database of signs, and be later integrated in a translation system where animations are generated from symbolic descriptions (as intended in the European project **SignON**[2], to which this thesis is related).

The avatar animation from an SL video can be divided into a two-step process: the first step is a pose estimation problem to obtain the absolute 2D position of the human skeleton (bones + joints) of the signer performing the sign, and the second step is to use the captured data to create the animation that will move the avatar. The animation of a skeleton usually requires to provide the rotations of the joints at each frame. Turning the positions into rotations can then be tackled in different ways. In this thesis we propose the use of a neural network model to estimate quaternions of each joint of the skeleton from the list of absolute positions retrieved from the pose estimator, and store them as an animation in the BVH format file, which is, according to Wikipedia[3], widely used by most 3D graphics applications. We present a new dataset and provide the corresponding script used to create it in order to be able to increase the amount of data if desired. The dataset will contain relations between skeleton poses represented as a list of 2D landmarks, and a list of quaternions of all joints that sets the skeleton in the same posture. The latter represents a step towards addressing the challenge respect to the specific issue Avatars and Computer Graphics identified by Bragg et al.'s [3] mentioned earlier, scarcity of public motion-capture datasets; it should also help in addressing another issue, namely, realistic transitions of the animations.

As the overall goal is to achieve the most realistic animations possible, we research and compare the common strategies of Motion Capture, namely, marker-based, sensor-based, and methods based on Machine Learning. We also specifically compare our approach with state of the art approaches for the second step of obtaining avatar animations, such as Brock et al. [4] and Grishchenko [5], which will be reviewed in Section II-D.

### C. Results

The focus of this thesis is on the so called *Manual Features* (MFs) of SLs, which can be understood as the animations of arms/hands. *Non Manual Features* (NMFs), such as *facial expressions*, *mouthing*, *gaze* and *torso direction* are also essential components of SLs, but are not the object of this thesis.

The results will be presented in three metrics: in terms of absolute distance between the position of all predicted and ground truth joints; in terms of absolute position difference; and in terms of angular difference. Additionally, as in the reference papers [4][5], both qualitative and quantitative visual plots will be provided to understand the strengths and weaknesses of our proposed system.

### D. Outline of Thesis

The rest of this master thesis is organised as follows. Section II presents state of the art related topics on this work, such as Sign Language Notations, Virtual Avatars, Motion Capture (MoCap), and Related Work, and puts them into context with the SL target application. Section III describes the proposed methodology of our approach, which includes the development of the novel dataset, the training experiments, and the proposed end-to-end system in which the trained neural network model will be put in use. Then, in Section IV we will evaluate the obtained results and discuss their problems and improvements. Finally, in Section V we will present our final conclusions and future work.

## II. State of the art

In this section we introduce first some basics of SLs and SL Notations to represent them. Then we discuss some aspects of the requirements for the skeleton of Virtual Signers to be able to show SL. Then we review and evaluate the different sensors which can be used to capture movements, which led to our choice of ML based systems. Finally, we discuss the two papers most closely related to our approach.

### A. Sign Languages and Sign Language Notations

It seems worth to start with a sentence from Murtagh's PhD thesis [6]: "SLs are visual gestural languages articulated in a signing space. SLs have no written form.". Naert et al. [7] survey on animating signing avatars provides an excellent and comprehensive background on concepts, approaches and existing systems. It covers linguistic aspects of SLs and alternative representations, scripting languages for SLs, approaches for both *sign* (the lexical unit) synthesis and *utterance* (somehow equivalent to sentence) synthesis, as well as existing signing avatars. The paper focuses on the so-called *manual features* (MFs), which comprise *hand configuration* (see in Figure 1 some of the hand configurations of the French Sign Language), *placement*, *movement* and *orientation* (we refer to, for instance, [8] for precise definitions). Kacorri [9] seems yet to be the most authoritative survey on the so-called *Non-manual features* (NMFs), such as facial expressions, mouthing, gaze and torso direction, and her PhD thesis [1] the most significant contribution related to NMFs.

[2]https://signon-project.eu/

[3]https://en.wikipedia.org/wiki/Biovision_Hierarchy, retrieved Aug18 2022

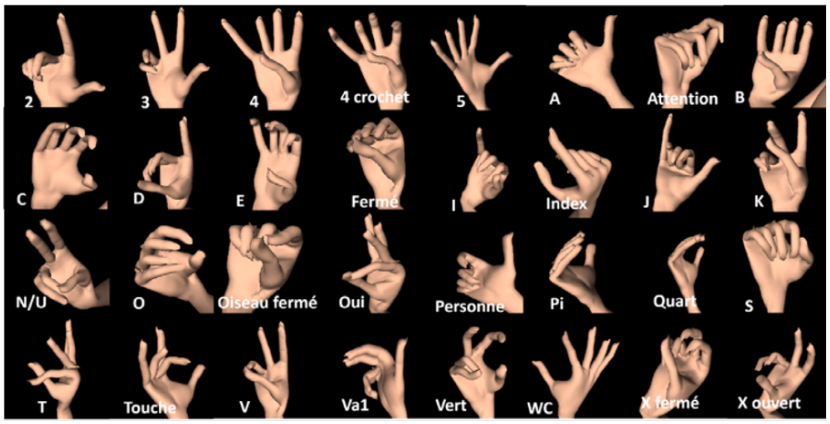



Fig. 1: Several hand configurations of the Langue de Signes Française (LSF) (from [7]).

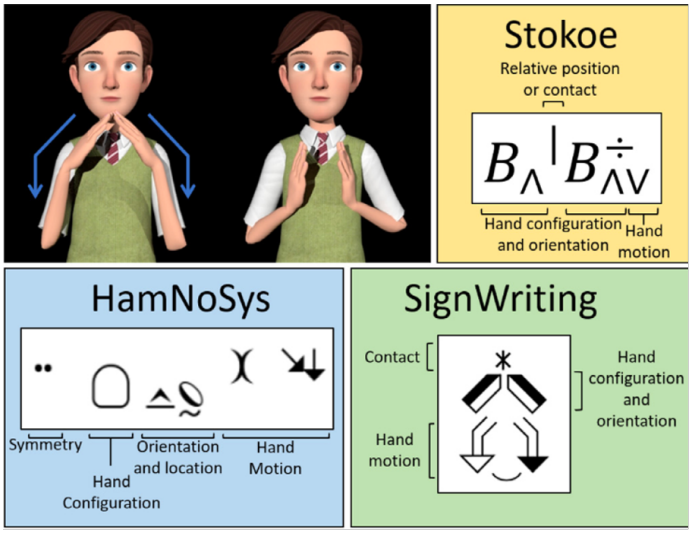



Fig. 2: An American Sign Language (ASL) sign in different notations (from [7]).

The formulation about SLs, stating that a sign can be expressed as a sequence of discrete values of five elements (namely, the hand configuration, the hand placement, the hand motion, the hand orientation, and the non-manual features) goes back to Stokoe [10] and Battison [11], who described the MFs, while the NMFs were introduced later. Based on these sequences, there have been some notations used to represent the signs that the avatar has to perform. In 1960, Stokoe created a first notation describing the ASL signs using a combination of hand configuration, hand placement and hand motion, it was called Stokoe's notation [12]. Unfortunately, there is no standard way of writing SLs, there is no equivalent of latin, cyrilic, ... alphabets. Isolated sign representation is complex, and for utterances one should add grammar complexity.

Later, the Hamburg Notation System or HamNoSys [13] was created, increasing the complexity of Stokoe's notation but also improving its utility. It includes the components of Stokoe's notation with the addition of some facial expressions. The variety of possible values for each of the components makes HamNoSys a much more complete notation system. The key innovation of this notation that made it something to use as a reference even nowadays is that it was designed to be language-independent and extensible to new linguistic mechanisms. Thanks to that, other notations appeared such as SignWriting [14], which is based on dancing gestures and tries to be a more graphical and intuitive notation. Figure 2 illustrates these notations of an American Sign Language (ASL) sign, hinting at their complexity.

However, Stokoe's notation, HamNoSys and SignWriting only deal with the representation of static and independent words or phonemes, but fail to correctly represent dynamic signs, the concatenation of static signs, and the synchronization of the different components.

More recently, SiGML [15] has been initially an XML version of the HamNoSys transcription system. This language was developed within the European projects ViSiCAST and eSIGN that promote Deaf access to information. It was designed with the prospect of animating virtual signers, this is why some aspects of SLs, like timing or very precise orientations, can be specified in SiGML and not with HamNoSys (correcting the mentioned problem). On Figure 3 we reproduce a comprehensive table on different SL representations and systems from [7], showing key features of each.

**Table 1**
Comparison of the sign representation.

| Category | Name | Fidelity | Temporal aspects, synchronization | Non manual features | Flexibility | Understandable by a computer |
|---|---|---|---|---|---|---|
| Visual Representation | Drawings | ✓ | ✗ | ✓ | ✗ | ✗ |
| | Video recordings | ✓✓ | ✓✓ | ✓✓ | ✗ | ✗ |
| Parametric Notation | Stokoe [32] | (✓) | ✗ | ✗ | (✓) | ✗ |
| | HamNoSys [26] | ✓ | (✓) | (✓) | (✓) | (✓) |
| | SLPA [37,39] | ✓ | ✓ | ✗ | (✓) | (✓) |
| | SignWriting [33] | ✓ | (✓) | (✓) | (✓) | ✗ |
| Scripting Language | SiGML (extended) [42] | ✓ | ✓ | (✓) | (✓) | ✓✓ |
| | QualGest [25] | ✓ | ✓ | ✗ | ✓ | ✓✓ |
| | Losson [43] | ✓ | ✓ | (✓) | ✓ | ✓✓ |
| | Zebedee [44] | ✓ | ✓ | ✗ | ✓✓ | ✓✓ |
| | EMBRScript [45] | ✓ | ✓ | ✓ | (✓) | ✓✓ |

**Fidelity**: absence of ambiguity, fidelity to the original movement, precise description of the sign, preservation of the intent of the signer. **Temporal aspects, synchronization**: the dynamics of the movement is specified, the synchronization between the different channels is managed. **Non manual features**: the status of non-manual components (facial expressions, gaze, etc.) is specified/visible. **Flexibility**: the ease with which the representation of a sign is modified to take into account the context of the sentence. A purely visual representation will make the transformation fastidious while some linguistic representations are highly flexible. **Understandable by a computer**: it can be reused as it is at the input of an automatic synthesis engine.

Fig. 3: Table on different SL representations and systems (from [7]).

An alternative for the animation of virtual signers is the use of XML based notation for the animation of embodied conversational agents, ECAs. Indeed, the Behaviour Markup Language (BML), born in the ECAs research community has been used in some signing avatar projects, such as Kacorri's PhD thesis [1], in a specific variant/extension which was more targeted to SLs, EMBR, introduced by Kipp et al. [16]. This is the route intended to be followed in the SignON EU project, to which this thesis is related. BML is based on tags (such as <head>, <gaze>, etc.) that are used together with a set of attributes, which can indicate the synchronization times of each behaviour and the different states (start, ready, relax, end, ...). The synchronization in BML supports absolute or relative timing reference. Absolute refers to the whole duration of the behavior block, and relative refers to other existing behaviors inside the block. Moreover it has synchronization elements that allow other cases such as <wait>, <before>, etc.

### *B. Virtual Avatars*

The animation of people or animals usually considers them as articulated structures and then based on *skeletons*, i.e., hierarchical structures of segments (*bones*) and *joints* (the articulations). Specifically, *rigging* refers to the process of creating the bone structure (skeleton or armature) of a 3D model. This bone structure is used to manipulate the 3D model like a puppet for animation.

The definition of the skeleton of a virtual signer has considered mainly two sources related to SL: the specifications from Murtagh's doctoral thesis [6], and the work on the creation of a corpus for the French SL (LSF) based on Motion Capture [17]. Murtagh's thesis proposed an armature that is able to support the animation of signs. It is composed of 62 skeleton joints distributed as follows:

- 2 joints in the head
- 4 joints in the spinal column
- 4 joints in both legs
- 4 joints in both feet
- 6 joints in both arms
- 42 joints in both hands

In fact, this distribution is similar to common skeletons used in 3D modelling (we will see other skeleton examples in Subsection III-B1, which will be used to train our neural network model), which shows that skeletons that we will use later in this master thesis will be valid to perform sign language gestures without missing detail. Also, it is important to remark the resolution required for hands in comparison to other parts of the body. The amount of joints on hands will be useful to represent SL, more precisely MFs. The joint hierarchy is also provided for the face and shoulders, to support NMFs. However, this thesis is concerned with MFs and we do not consider them further here.

We will also discuss later the list of captured landmarks in pose estimators such as MediaPipe [18] and OpenPose [19]. And we will notice that they also adopted a similar distribution than the 3D models skeletons. This will be very advantageous to ease the relation between the skeleton from pose estimators and the skeletons that moves virtual avatars.b

### *C. Motion Capture*

Heloir and Nunnari [20] can be mentioned as an inspirational work for our end-to-end system. This paper proposed a user-based authoring system for SLs related to the animation of signing avatars. A diagram and a picture illustrating the system proposed are shown in Figure 4. Heloir and Nunnari [20] method was based on motion capture using some input devices,

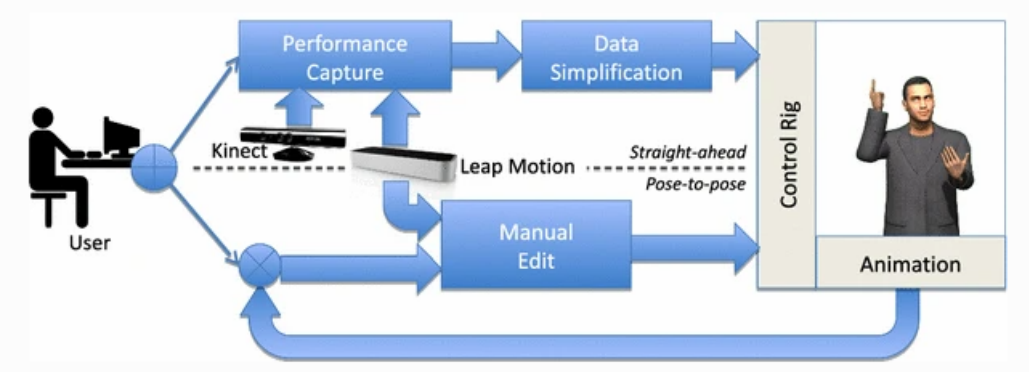



Fig. 4: Diagram of the authoring system proposed by Heloir and Nunnari [20].


Fig. 5: Marker systems with and without gloves.

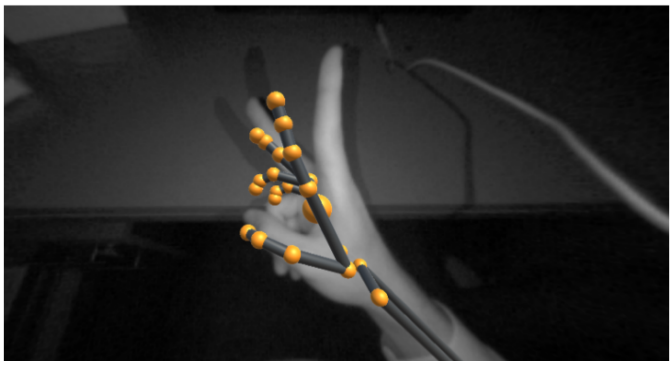


Fig. 6: Sensor-based systems example using Ultraleap Stereo IR 170.

Leap-Motion and Kinect-like, which were then relatively novel. As hand capture is very important for MoCap, we have been exploring different possibilities in order to evaluate which was the best current approach to follow in our end-to-end system. Indeed, we have been testing different methodologies: marker-based, sensor-based, and machine learning-based. And the brief discussion below summarises our main conclusions from our tests and the information available.

*Marker-based* methods are highly intrusive as seen in Figure 5. Signing requires a very precise movement of fingers, which tends to be in contact between them. The existence of markers disturbs the signer to perform some specific hand configurations where fingers have to be placed on the marker's location. In a real-time capture of sign utterances, this problem provokes the imperfection of the captured data. Additionally, sign gesturing mostly implies the occlusion of some markers, which creates quite a lot of errors and noise that need to be removed or treated in a post-processing step. This step usually takes more time than the MoCap session of the signs.

*Sensor-based* methods have a limited range of action, but give a high accuracy of depth. From the sensors researched and the ones we experimented with, they are intended to be used with Virtual Reality technologies, so they are commonly located in the head or face of the user and pointing to their hands. This approach is not suitable for sign language recognition since face and body are as much as important as hands. Moreover, in Figure 6 we can see how it failed at retrieving easy situations and created unnatural poses of hands (which needs to be avoided at signing).

*Machine Learning* methods have recently become very powerful as supported by the strong push that has meant the somehow new research field of deep learning. Movements are estimated from models, which are trained in correct real cases, and actually helps a lot at dealing with occlusions. Among all of the well-known architectures, we tested using OpenPose 2D [19] and MediaPipe [18]. The comparison and evaluation between them will be carried out throughout this document.

Aside from this research, we organised a session with a signer to collect signs using a marker-based *high-end MoCap system* available at the UPF. Experimenting with an existing (high-cost) system as soon as possible, should be useful to identify as many issues as possible to inform the (low-cost) tool we intended, while at the same time see how well a high-cost system performs to be able to compare results. After some difficulties due to the pandemics situation, we ran a capture session at the end of October 2021. The setup was prepared with a non-signer. The planning included the signs in LSC (Llengua de Signes Catalana, Catalan Sign Language) to be performed, so that the different hand configurations were covered. The session allowed us to identify quite a few issues: signs which were especially difficult to capture using even a high-end MoCap system, the complexity of the software associated to process the raw data, and the noisiness and errors of the capture (46 markers, which

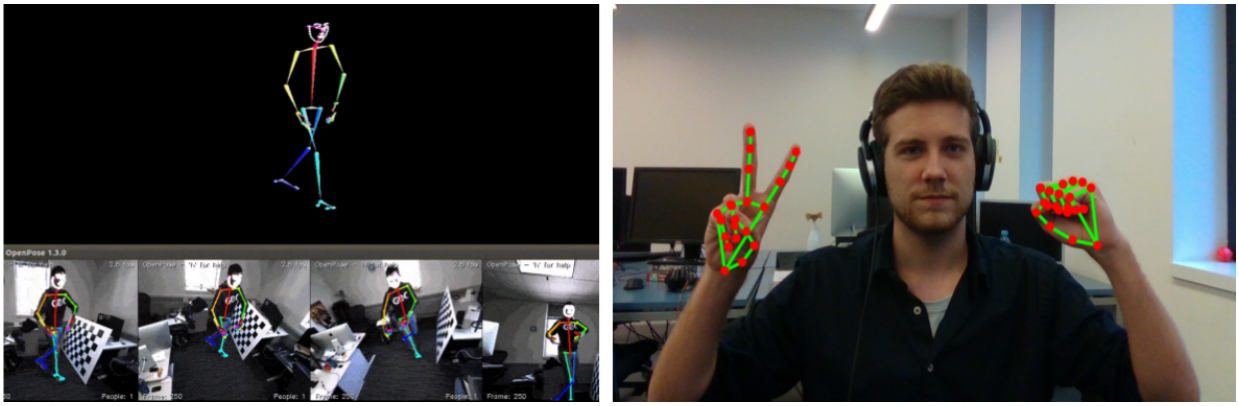


Fig. 7: (left) OpenPose 2D (from [4]); (right) MediaPipe hand recognition.

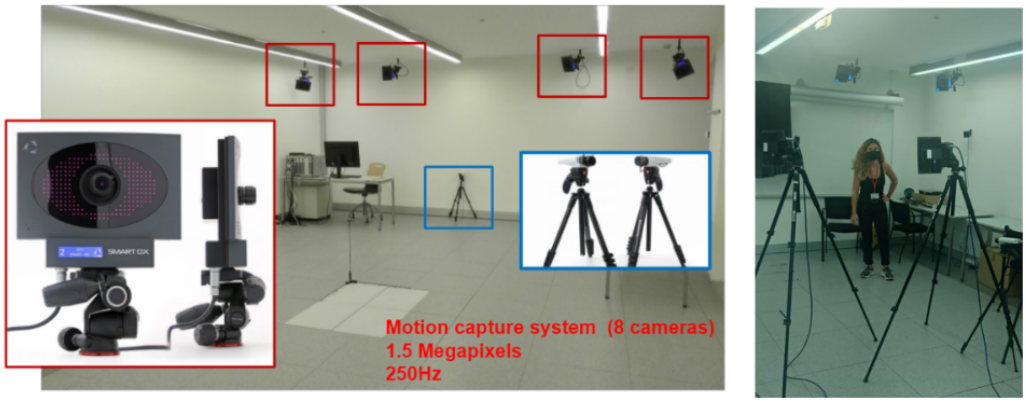



Fig. 8: Pictures of the MoCap Lab and systems used.

should have led to the same number of capture channels, generated approximately 250 channels). Of course, we saw that some improvements could be made in the process of capture. However, it seemed very clear that existing free systems such as MediaPipe or OpenPose were a lot simpler to use, and seem to offer better quality results without the need of learning to use auxiliar softwares (related to the capture systems) to clean and correct the data.

### *D. Related Work*

After presenting the main concerns related to this master thesis problem, we will briefly introduce and discuss the main related contributions up to date, which are state of the art proposals in sign language synthesis. We will see their capability, their results, and their limitations.

*1) Brock et al. approach (April 2020) [4]:* Their work presents a novel pipeline for the generation of 3D skeleton data from sign videos. Their contributions consists on: a neural network for estimating 3D joint positions from single-camera video; extended robotic computation of joint angles of a virtual avatar from the joint positions; combination of the previous novelties into a functional pipeline; and evaluation of the pipeline in the intended application scenarios. Their system is defined by the two-step approach presented in Section I, and a representation made with boxes of their pipeline flow can be seen in Figure 9.

It is worth to mention that their work is based on a dataset they made in previous work (Deep JSLC) [21], namely, a collection of 2,632 annotated sign videos (with an average length of $\approx$ 500 frames) and their corresponding synchronised motion capture files. They use this dataset to train three separate Recurrent Neural Networks (RNN) (one for each body part: stick figure, fingers and face) that infers the depth component from a 2D landmarks input obtained from OpenPose 2D. RNNs are chosen due to their ability of learning and storing sequential information. Once they have the inferred the absolute 3D position of all joints, they use it to compute angular and positional displacements following the Inverse Kinematics calculus in [22]. Finally, the computed displacements are merged into a BVH file [23] for avatar animations accordingly to the hierarchical structure of the avatar skeleton.

They perform three evaluations of their system: the absolute framewise similarity between generated data and the real motion capture data (in centimeters); the Sign Language Recognition (SLR) of the synthesised data; and a user test evaluation in terms

[4]https://cmu-perceptual-computing-lab.github.io/openpose/web/html/doc/index.html

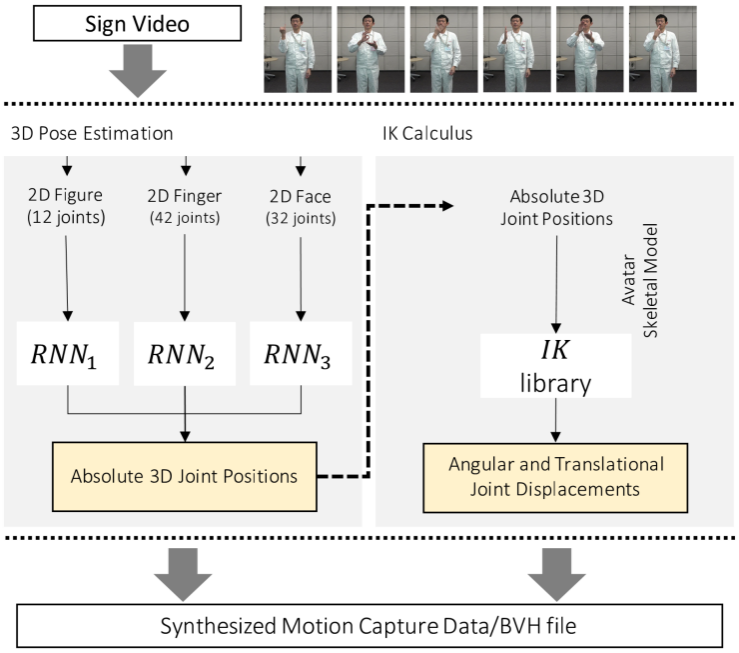



Fig. 9: Overview of Brock's system flow (from [4]).

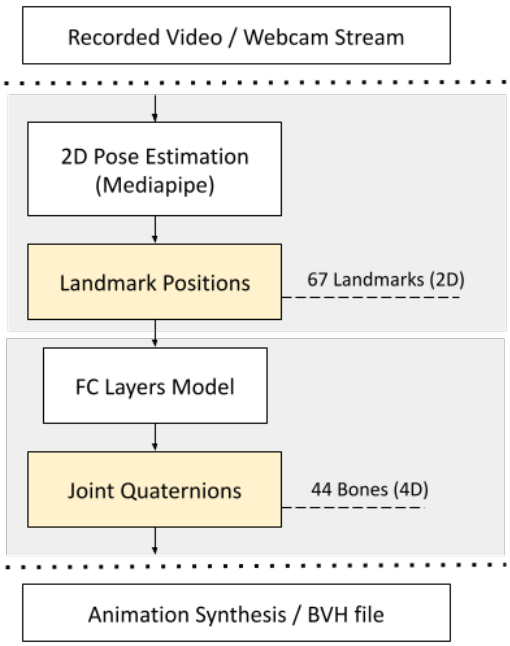



Fig. 10: Overview of the proposed system flow.

of Sign Language Synthesis (SLS) by displaying sign avatar's animations. They note that hand-made animations are preferred by signers, while synthesised avatar animations fall short of natural movements.

*2) Grishchenko et al. approach (June 2022) [5]:* Their work presents a novel lightweight neural network for real-time (15 FPS) 3D human body landmarks estimation, from previous MediaPipe 2D pose estimators. Then, they use another neural network to convert from 3D landmarks to mesh parameters; however, they do not really explain the details about it. Finally, they evaluate their error in terms of Mean per joint positional error (MPJPE).

As they point out, "In contrast to 2D, which can be obtained via human annotation, accurate manual 3D annotation is a uniquely challenging task". They mention the requirement of either a good lab setup with 3D scans or building a synthetic dataset, each of the approaches with additional challenges.

Similar to Brock et al., they use their own dataset to infer the depth component which was created by annotators that specified the depth order between pose skeleton edges. Furthermore, they present demos of avatar movement using their system with the avatars from Mixamo, which is also used in this master thesis approach.

## III. Methodology

We have presented the State of the Art approaches most closely related to the problem we deal with in this TFM, the improvement of the animations to be used in SL synthesis based on ML. In this section, We explain our proposed approach. First, we discuss the different aspects of the creation of a novel dataset of videos representing animations, and for which the ground truth is available. Then, we discuss the training process, where we use an existing system to estimate 2D poses, and train a new neural network model turning these poses into quaternions. Finally we explain the end-to-end application where the approach implemented is integrated.

### *A. Proposed Approach*

As we could see in related work, the usual focus is on the acquisition of high accurate 3D positions of the body pose and hands. This is because of their commitment on creating their own datasets which required a lot of time and the implication of external annotators. In both works the dataset exceeds the 100,000 frames. In this project, the goal is to create an easy-to-access dataset of at least 50,000 frames which can be automatically augmented for future improvement. Our approach will also differ in the data acquired while maintaining the focus on the synthesis of animations.

Brock's et al. approach, focuses on the inference of the absolute joint positions from video, for then obtaining the animation of the virtual skeleton by using existing Inverse Kinematics (IK) calculus [24], which leads the skeleton joints to the specified location at each frame. One issue of this approach is the conversion from 3D landmarks to skeleton joint positions, as both data do not coincide with the same exact locations on the body, so a rough estimation is required. Another significant limitation of this approach is the complexity of the IK algorithms, which need to proceed sequentially along the joints of the hierarchy

and the subsequent possibility of accumulating errors. To be more precise, virtual avatar animations can be described by quaternions, 4D rotations that allow a character to rotate about multiple axes simultaneously, which saves a lot of operations of multiplying rotation matrices when combining several successive rotations. It is also quicker to renormalise a quaternion than it is to renormalise a rotation matrix. Our approach intends to learn the quaternions in a way that is as direct as possible, which should result in less work needed in order to create animations of the movement, and less errors in the post-processing to create the animations. If this is achieved our approach for obtaining quaternions should be more efficient and more accurate to generate virtual avatar movement.

The less direct approaches, as the one mentioned above, probably come from the fact that the databases of videos used for learning do not have ground truth quaternions. In this TFM we envisaged the possibility of creating a dataset, where the ground truth quaternions are known, by using existing quality database animations, from which the resulting rendered videos could be the starting point for learning. Then, we try to tackle these issues and present an approach based on this novel idea.

It is necessary to carry out different stages of work: the creation and preparation of this novel dataset with the previously mentioned conditions; the training of a model that learns to convert from 2D landmarks to quaternions; and an end-to-end application that allows us to put in practice the proposed approach. We can see a visual pipeline in Figure 10. All these phases are explained in detail in the following subsections III-B, III-C, and III-D.

### *B. Dataset Preparation*

This subsection explains the generation of the dataset used in the project, along with its constraints and specifications, while discussing different details.

The dataset should contain information about realistic animations and the positions of their skeleton joints in a 2D plane. The animations are retrieved from the web of Mixamo, a company within the Adobe group, which currently offers free open-source animations [25]. Mixamo is widely popular due to its free usage and the high-quality animations, made by artists. Despite these animations not being related to SL, we thought that the large amount of skeleton positions available and the corresponding ground truth quaternions should help to predict a wide variety of possible states, both related and unrelated to SL.

From the (rendered) animations resulting into 2D videos, the absolute position of the joints is obtained with a pose estimator. As mentioned in Section II-D, in Brock et al.'s paper they evaluate some state of the art pose estimators and only two of them provide finger information on top of the overall stick figure of the body, OpenPose 2D[5] [19] and MediaPipe[6] [18]. Additionally, MediaPipe estimates a rough depth information from single-camera videos. However, their updated documentation stated that "depth should be discarded as currently the model is not fully trained to predict depth", even though it is something they have on the roadmap. On the other hand, OpenPose 3D only works in multi-camera systems, which is not suitable for our final application, where we intend that native signers use desktop or smartphones as a source. So we decided to use MediaPipe, because it provides working solutions for desktop, mobile and web while also being more accurate and performant compared to OpenPose 2D, specifically in situations of hand-hand or hand-face overlapping which are directly related to the problem faced in this project (see as well a similar point of view in [26]).

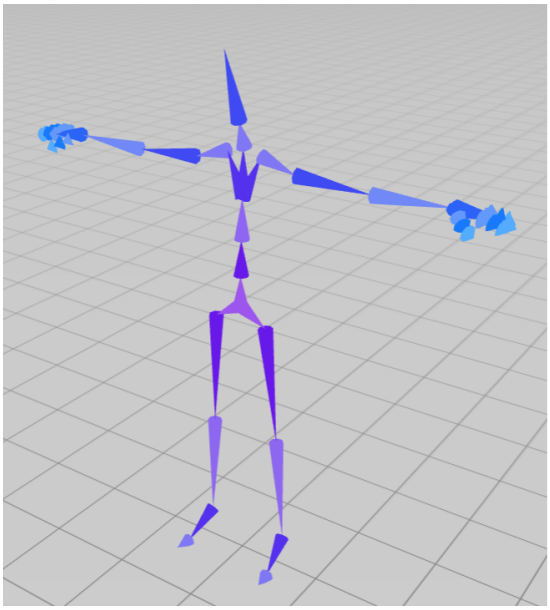
*1) Data Vectors:* The vectors of this dataset are the 2D landmarks retrieved from the pose estimator (input vector), and the quaternions of each joint (output vector).

Let us deal with the quaternions first. The quaternions can be easily obtained as ground truth data from the Mixamo animations. We just need to extract the information in the desired composition. The length of this vector is the number of bones of the skeleton multiplied by the 4 dimensions of the quaternion. The skeleton used by Mixamo has 52 bones, however, in order to help the neural network model to learn more easily, we carry out a preprocessing step to reduce the number of bones to 44 (for a detailed explanation, see III-B4). The result is a vector of length 176 (for each frame). Figure 11 presents the list of bones in a common skeleton from Mixamo.

The landmarks require some additional work to be obtained. Based on the data that we can get from Mixamo, we use Blender (a free and open-source 3D software toolset which can be used in most of the Computer Graphics tasks such as animation, 3D modeling, etc.) [27], to create a 3D scene and display the animation of a character or avatar within it. Then, we set up a virtual camera targeting the avatar and we render the scene in a texture. This texture is used as input image for a pose estimator (Holistics, from MediaPipe [28]), which infers 3D coordinates for a set of landmarks for body and fingers, distributed as in Figure 12. It also obtains landmarks for the face, but NMFs are out of the scope of this project, and thus we skip them. In the same way as with the output vector, we also skip some landmarks that could negatively impact on the training process. Therefore, we end up with 21 positions per hand, plus 25 for the pose, multiplied by 2 (horizontal and vertical axis, as depth is discarded, as discussed earlier) gives us a vector of length 134, for each frame.

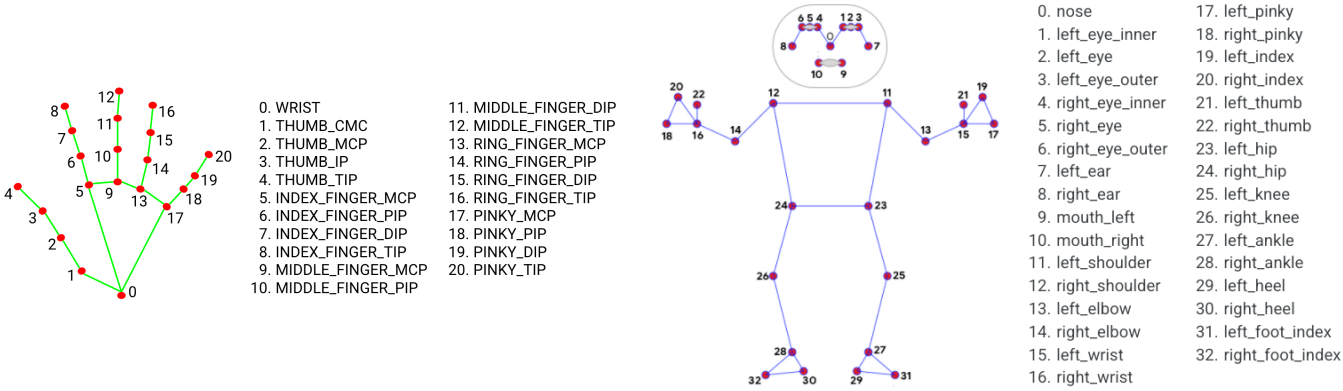
[5]See the documentation in https://cmu-perceptual-computing-lab.github.io/openpose/web/html/doc/index.html

[6]See the documentation in https://google.github.io/mediapipe/solutions/holistic



0. mixamorigHips
1. mixamorigSpine
2. mixamorigSpine1
3. mixamorigSpine2
4. mixamorigNeck
5. mixamorigHead
6. mixamorigLeftShoulder
7. mixamorigLeftArm
8. mixamorigLeftForeArm
9. mixamorigLeftHand
10. mixamorigLeftHandThumb1
11. mixamorigLeftHandThumb2
12. mixamorigLeftHandThumb3
13. mixamorigLeftHandIndex1
14. mixamorigLeftHandIndex2
15. mixamorigLeftHandIndex3
16. mixamorigLeftHandMiddle1
17. mixamorigLeftHandMiddle2
18. mixamorigLeftHandMiddle3
19. mixamorigLeftHandRing1
20. mixamorigLeftHandRing2
21. mixamorigLeftHandRing3
22. mixamorigLeftHandPinky1
23. mixamorigLeftHandPinky2
24. mixamorigLeftHandPinky3
25. mixamorigRightShoulder
26. mixamorigRightArm
27. mixamorigRightForeArm
28. mixamorigRightHand
29. mixamorigRightHandThumb1
30. mixamorigRightHandThumb2
31. mixamorigRightHandThumb3
32. mixamorigRightHandIndex1
33. mixamorigRightHandIndex2
34. mixamorigRightHandIndex3
35. mixamorigRightHandMiddle1
36. mixamorigRightHandMiddle2
37. mixamorigRightHandMiddle3
38. mixamorigRightHandRing1
39. mixamorigRightHandRing2
40. mixamorigRightHandRing3
41. mixamorigRightHandPinky1
42. mixamorigRightHandPinky2
43. mixamorigRightHandPinky1
44. mixamorigLeftUpLeg
45. mixamorigLeftLeg
46. mixamorigLeftFoot
47. mixamorigLeftToeBase
48. mixamorigRightUpLeg
49. mixamorigRightLeg
50. mixamorigRightFoot
51. mixamorigRightToeBase

Fig. 11: List of bones of a common Mixamo's skeleton.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Fig. 12: Distribution of Landmarks for Hand (left) and Pose (right) (from [7]).

Taking advantage of the need to use Blender, all the mentioned computations to obtain the input and output vectors have been coded in Blender Scripting, which can interpret programming scripts coded in Python, and has an API to interact and make use of all the tools that Blender provides. Besides unifying and simplifying all the process in one single script, we also are able to do other complex tasks such as 3D rendering and MediaPipe library usage.

Something to mention is that we are not using MediaPipe with a video stream but with single images, which is a drawback that we need to take into account and compensate later on in the preparation of the rendered scene. We cannot use a video stream directly in MediaPipe because we have quaternion information per frame, and the frame rate of the animations from Mixamo is fixed to 30Hz or 60Hz. Then, in order to obtain the correct pair of the quaternion data and the render time, we need to evaluate the animation only in the times where there is the correct information of the quaternion. Also, MediaPipe with frames does not have tracking unlike with a video stream, which would greatly help in not losing the pose when a quick movement is performed and maintaining the fidelity of all body parts in time.

Focusing now on the script details, it has several flags and global variables that can be easily modified to obtain different outputs if desired. There exists two flags to control states of the execution:

- *"use_editor_camera"* enables the usage of the camera of Blender in Layout workspace. This way the user can easily move and set-up a new camera if desired in a very intuitive way, unlike if it had to be specified blindly in world coordinates.
- *"debug_mediapipe"* controls the display of the images rendered by the camera at each frame. This is a well-trusted way to visually see if the script is working correctly or if it contains too many errors.

[7]https://google.github.io/mediapipe/images/mobile/hand_landmarks.png and https://google.github.io/mediapipe/images/mobile/pose_tracking_full_body_landmarks.png

Fig. 13: Render of Kate with the default (left) and corrected (right) textures.

Both functionalities will be discussed more in detail in section III-B2.

Apart from that, we can modify three main variables that change the output vectors and allow the user to make use of the dataset in other situations than the one tackled in this project:

- *"resolutionX"* and *"resolutionY"* controls the number of pixels of the image rendered. This is important because this same image is used as input in MediaPipe, affecting its accuracy and performance.
- *"skip_quats"* is a list of indices of the quaternions that will be skipped when filling the ground truth vector. The list is written in strings that match the names of the bones in the animation files (*"mixamorig:<Bone Name>"*). Check Figure 11 to see the list of bones.
- *"skip_lms"* is a list of indices of the landmarks that will be skipped when filling the vector created from MediaPipe data. The indices are sorted as: pose landmarks, right hand landmarks, left hand landmarks. Check Figure 12 to see the list of landmarks.

*2) Scene Render:* The configuration of the scene and the render viewport are essential aspects to consider in order to generate a correct dataset, i.e., one that can be used to address the problem that this project aims to solve. We explain next how and why the rendered images need to be created.

Pose estimators are computer vision techniques that detect human figures in images and videos and provide some 2D-3D position estimations of key landmarks. Our approach relies on the assumption that we can generate renders with enough quality for a program to detect a virtual avatar as a real person within its goals - in this case, a pose estimator. In order to achieve that, we need to "help" the pose estimator as much as possible by setting up a comfortable environment.

Regarding the avatar, Mixamo counts with a large set of virtual avatars that can be moved by any of the animations. However, there is no need to use all of them as input to try to generalise the neural network model to support different people, because the image is processed first by a pose estimator, already trained with different people. This means that creating the dataset using a single virtual avatar will ensure that the neural models we create work correctly for all people. Indeed, otherwise the problem would fall into the pose estimator module, which is external to this master thesis work and the way out would be to replace it by an alternative without much difficulty.

Then, we selected the most realistic human avatar among all those available and additionally, whose movements would be more salient (both because of the clothes or because of possible bone constraints). Apart from that, a pre-processing step was needed to correct a problem in the design of the textures of the avatar. There was a miss-matching in the file format between the exporter of Mixamo and the importer of Blender, Mixamo does not export the data required in the system Physically-Based Rendering (PBR), therefore Blender sets some values to default ones, leading to visual errors. The selected avatar Kate and the correction on textures can be seen in Figure 13.

Regarding the scene, we used some techniques to improve the render quality: a specific three-points lighting composition

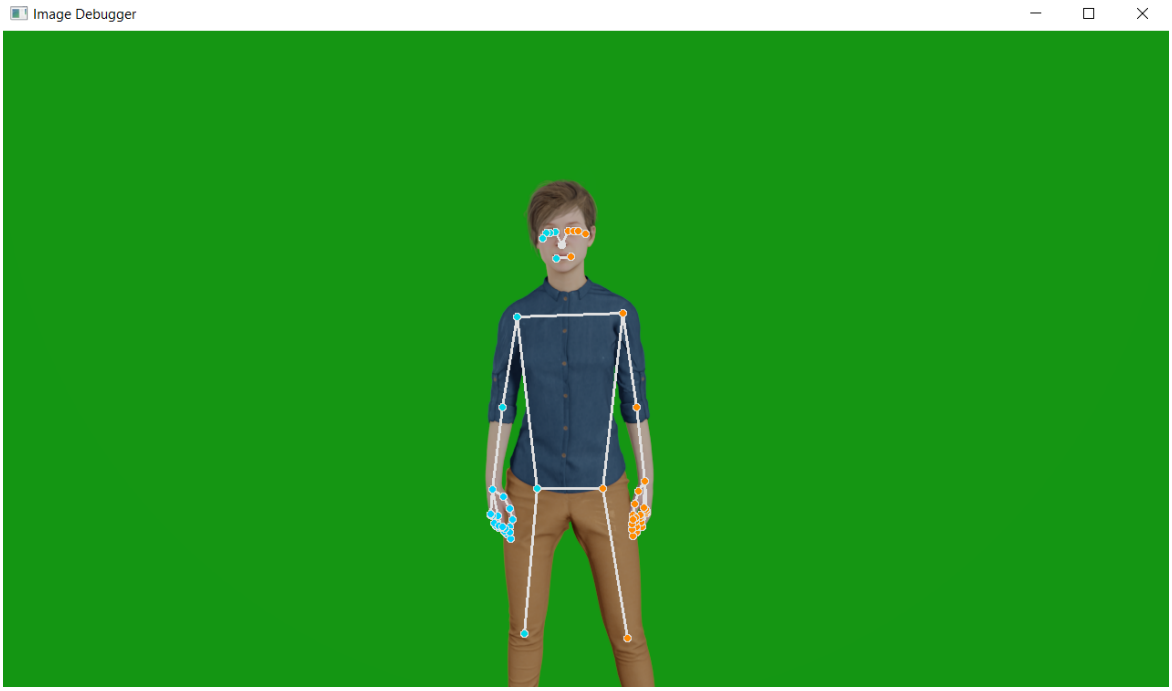



Fig. 14: Rendered image used as input in the pose estimator.

and a plain green-screen background. The lighting composition includes the *key* light, targeting the frontal face of the avatar; the *fill* light, a less powerful complementary light to the key light that illuminates the other side of the front face; and the *rim* light, which illuminates the model from behind and helps separating the avatar from the background by highlighting the contours. A plain green-screen background has been in place to increase the difference between the foreground (avatar) and the background and help the pose estimator.

Regarding the camera, the configurations do not only affect the quality of the dataset, but the actual learning of the neural model. Indeed, this was shown by our first implementation, whose results were not satisfactory. In our first implementation, every animation was rendered with a different *camera set-up* in order to maximise the number of frames captured by the pose estimator (e.g. if the animation is facing to the left, move the camera so that it is located in front of the virtual avatar). Even though it might seem the correct way to proceed. Landmarks are stored in a range [0,1] between the width and height of the frame. However, an animation facing straight will have different quaternions than a very similar animation facing upwards. Rendering each of them with a different camera, we would create a similarity in the inputs (landmarks) despite having a very different output (quaternions). This was corrected in the subsequent implementations. The camera should be fixed for all animations according to the use case it will be used for. We decided to set the camera in front of the model along the frontal axis, a bit elevated (imitating webcam set-ups), and slightly tilted towards the chest of the model. This decision prevented us from using any animation that is not facing straight along the frontal axis. Fortunately, this restriction did not prevent the project from moving forward.

Other related issues were the *resolution* and the actual camera *frustum* that encompasses the information that should appear in the image frame and in which conditions. More precisely, the resolution influences the amount of pixels that will be used to render the scene and in which distribution. In this project use case, since it is expected to work in webcam resolutions, we used the same dimensions than the standard HD (1280x720 pixels), but in fact, MediaPipe output is normalised between [0,1], so it would work in other standard resolutions that meet the same aspect ratio of 16:9 (Full HD, 1920x1080 pixels; 2K, 2560x1440 pixels). We kept in the middle of the trade-off about using more pixels (which helps the pose estimator) and the computational cost. With respect to the frustum, evaluations to maximise MediaPipe accuracy were also needed because after some experiments we concluded that MediaPipe performs better if including the legs in the image. As mentioned earlier in Subsection III-B1, the pose estimator is not able to do tracking between the frames of the animation, so all adjustments to ease the detection of the avatar are improvements to the dataset and to the training of the final neural model. In Figure 14 we can see the typical rendered images we used as input for MediaPipe.

*3) Dataset Visualization:* The purpose of this subsection is to present the tool developed to qualitatively assess/understand the data of the dataset created. The application is available at the following link: *https://victorubieto.github.io/animationLoader/*

It is very common to split the dataset into train, validation, and test, when training the NN model. And this split should

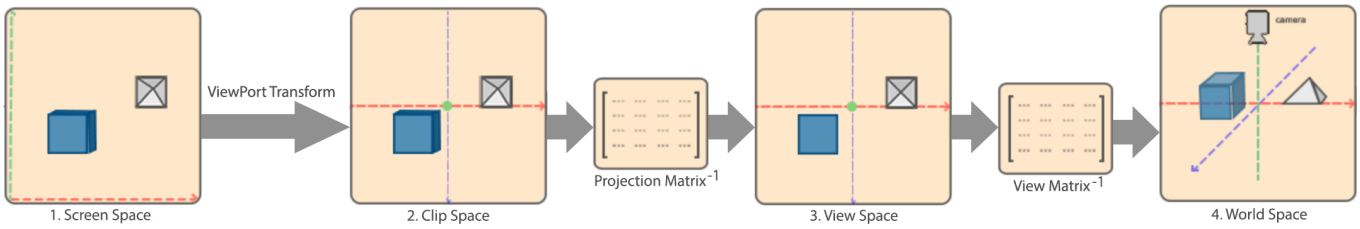



Fig. 15: Inverse projection pipeline from Screen Space to World Space.

be consistent, which means that the researcher would rather keep the same division for all training experiments, so that the evaluation of the results is not impacted by the dataset split itself (we will go into more detail in Section III-C). Then, it is always useful to be able to visualise somehow the data from the dataset in order to check whether it is correct, or to see whether (for instance) the test set is contaminated by bad animations. The only way to visualise the data used in this project, quaternions, is to display a skeleton performing the animation within a 3D scene.

We created a web-based tool in JavaScript with an interactive 3D scene using Three.js, an easy to use, cross-browser, general purpose 3D library [29]. The tool interface is based on a simple scene with a 10x10 grid (each square representing $1m^2$) that helps measuring the amount of movement, and the skeleton used in rest pose in the center of the space. On the top-right, a drop-down folder called "Evaluate Dataset" contains all the functionalities to evaluate both qualitatively and quantitatively our dataset. In the following bullet list we present a brief explanation of each functionality:

- *Reset Camera:* It sets the camera in the same configuration as the virtual camera used to create the dataset. This is useful to visualise correctly the landmarks because they do not have depth and otherwise we could not assess its accuracy with respect to the animation.
- *Reset Animation:* It restarts the animation uploaded. This is useful when comparing the ground truth animation and the prediction, to match the play time.
- *Load Ground Truth:* This option allows the user to load an array of quaternions which represents an animation. As mentioned earlier, our vector of quaternions skip the lower-body information, however it does not create error in the load because the indices of the legs corresponds to the last indices when creating an animation. Then, the remaining values (leg bones) are set to 0, which implies no movement.
- *Load Landmarks:* This option adds a set of 2D circles in each MediaPipe landmark location. In order to display correctly the landmarks in the scene space, we perform an inverse projection to each position. This can be achieved by using the projection and the view matrices, obtained from the camera and the scene conditions. This whole process is visually shown in Figure 15. As mentioned in the first bullet point, in order to see the landmarks properly the user has to put the camera in the same place where the camera with which we captured the dataset was.
- *Load Prediction:* This allows the user to load an animation in the same format as the "Load Ground Truth" option. Then, the animation is displayed in a new auxiliary skeleton that appears on the right of the ground truth skeleton.
- *Evaluate Prediction:* This option requires that the user has uploaded both the ground truth animation and the prediction, if not it will ask for those missing. If the condition is met, the application computes two arrays of data (both of them in centimeters): one stores the world position locations of each bone; and the other computes the error distance of each bone between the two animations. These data will be used later on in the Section IV-C. However, in order to obtain them, we need to generate the avatar in a virtual space with the same coordinate system.
- *Landmarks Color:* It changes the color of the displayed landmarks. By default is set to red.
- *Show Flags:* This set of flags allows the user to control which information wants to show or hide in the scene. By default all of them are set to True.

*4) Pre-process / Post-process:* This subsection contains the details about all the pre-processing and post-processing steps in the different stages of the project. For simplicity we will first explain the processes in the training stage, and then those applied to the end-to-end system.

In training, three important processes were necessary before fitting the model. First, as mentioned earlier in Section III-B1, the reduction of the input and output vectors, skipping the depth and visibility information from MediaPipe, and skipping all the landmarks and quaternions belonging to the lower-body. Second, since the pose estimator might not obtain a result for all the frames, we need to remove the quaternions from the ground truth of the "lost" frames accordingly. And third, at each training experiment, one data augmentation transformation was performed. Because it is compulsory to have the camera fixed when generating the dataset (as explained in Subsection III-B2), we need to generalise the model for other possible input camera set-ups. Specifically, we analised the possible situations that could exist in a real case example, and saw that there may

be two variations: that the person is not exactly located in the middle of the camera; and that the person is located in different depth positions. The first situation will be tackled as pre-processing in the final system, but the depth positioning of the camera has been derived to the data augmentation. To implement it, we estimated the closest and furthest positions of the camera (for instance, the camera will not be located at near positions where it cannot capture the arms), and we computed their distances values using the world coordinates of Blender scene (where we had our camera of reference). With that, we obtained a rate that would indicate the maximum scale to simulate the camera variance along depth. Luckily, since we located our virtual camera very far from the avatar in Blender (in order to capture its legs too), we considered that position as the furthest, so the scale only applies moving the camera closer to the person. Finally, the process to augment the data is: convert landmark data from [0,1] range to [-1,1]; apply a random scale factor between the computed range of values; normalise again the data to [0,1].

Once we trained our model and implemented it in the end-to-end system, we also need some processes to correct possible problems. For simplicity, we first present the pre-processes, and then the post-processes.

The first two pre-processes are related to the lack of input information. Of course, if the pose estimator fails at capturing the landmarks, the rest of the system cannot retrieve any possible output. For that reason, we correct this issue in two ways, depending on the temporal location of the lost frames. If the lost frames are at the beginning of the input video stream we cannot recover the movement, thus we cut the video until MediaPipe finds some information of the pose and fingers, and make that frame the time 0 of our input video. The same process but inverted is carried out for the last unknown frames of the video. On the other hand, when we lose a couple of frames due to occlusion or other problems but we have information about the previous frames and the following ones, we can predict the intermediate frames by using a simple 1D linear interpolation. In fact, this correction is used in other approaches [4] mentioned in the related work.

As indicated at the beginning of this subsection, the dataset is created with a set of animations where the virtual avatar is located at the middle of the frame, but it may not be the case in real-use situations. For this reason, before passing the list of landmarks to the neural network model, we center them by applying an offset displacement equivalent to the distance from the ”hips” landmark to the desired center.

Once we are able to generate a prediction, we need to apply a set of corrections to the quaternions. First of all, quaternions must meet two conditions: that the coordinates of the positions have a range [-1,1]; and that the quaternions are unitary, so their norm is equal to 1. But the output does not always meet these conditions, so we need to normalise between this range. Then, a low-pass filter in the temporal dimension is applied to the quaternions of each joint in order to reduce the noise of the inferred animation. As a matter of fact, noise is caused because there is no continuity between frames. The method used is a Savitzky Golay Filter [30], which is well adapted for data smoothing. Finally, we normalise again the quaternions to obtain a correct set of values to use. Evaluation of this process is carried out in Section III-C. The last post-process corresponds to the usability of the result for other virtual avatars. Until now it was stated that we used a unique skeleton to train the model (Kate), therefore, it will only be able to predict its bone hierarchy. However, in the computer graphics field there are techniques that map an animation created specifically for a single model to others; this type of techniques is called retargeting. This technique uses a bind pose to connect the bones from the source skeleton to the target one. Then, the animation is created by applying a transformation at each frame that relates both skeletons. Once this is done for all frames, we just use the target skeleton and the new animations generated from the source. The quality of the resulting animations depends on the degree of difference between the two skeletons. For instance, we will most likely fail if we try to retarget an animation from a humanoid to a quadruped. Nevertheless, this project problem only targets human avatars, whose skeletons differ only in minor nuances. In the future, We may encounter difficulties in extending the animations to models other than adult humans, such as the elderly and children. However, we have adopted the retargeting methodology and we will study its limitations in such cases, which are beyond the scope of this project.

### *C. Neural network model and training*

*1) Overview:* Once discussed the dataset created, we discuss first the architecture of the neural network model that we will use for training, namely, the overall functioning of the training process, and an easy setup of hyperparameters that we will be using in the training experiments that we detail later.

Summarising, we use the retrieved list of landmarks, consisting in an array of 134 values (2 coordinates for 67 landmarks) obtained from MediaPipe, as input data of our neural network model. Our model is composed of a chain of Fully Connected Layers (FCL) that estimates the best quaternions that rotate the skeleton into the position closest to that specified by the landmarks. The outcome of the system is a list of 176 quaternions (4 coordinates for 44 joints).

The loss function used is the *Mean Square Error* (MSE), as it is appropriate for the case of our problem, based on a regression model. The MSE is computed for each batch, which is composed by an amount of input arrays specified by the *batch size*. In Equation 1 we can see the formula of the loss, where $y^{true}$ is a matrix of ground truth arrays of quaternions, and $y^{pred}$ is a matrix of arrays of the estimated quaternions. The number of rows of the matrices is defined by the *batch size*,

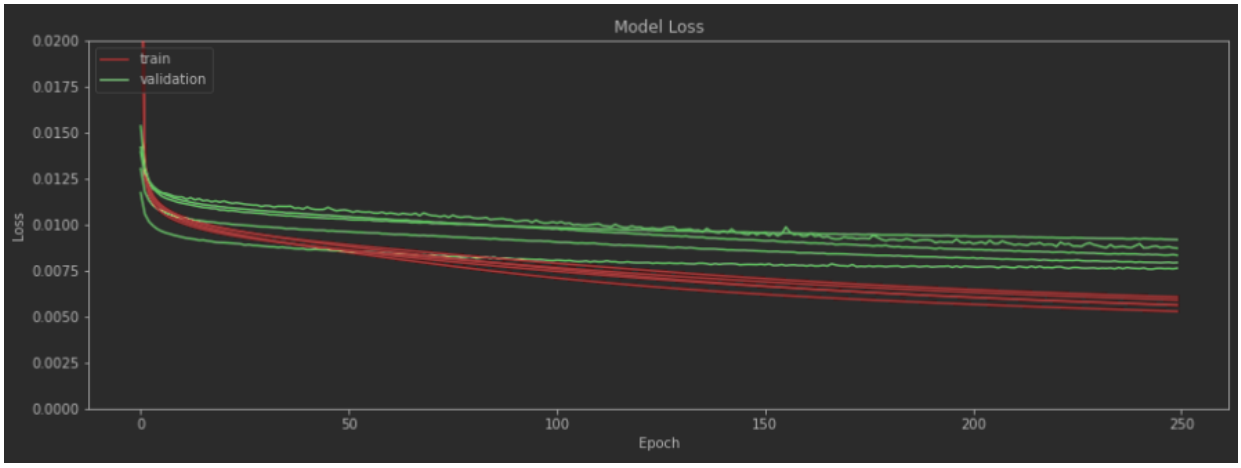



Fig. 16: Comparison of losses for the 5 different split sets.
The green curves correspond to the 5 validation losses, and the red curves correspond to the 5 training losses.

each row representing a frame of an animation. Then, $y_i$ represents each of the values of the matrix. And finally, $n$ represents the amount of values of the matrix, which means that the MSE is computed for the whole batch. Then, the expression of the loss is the following:

$$\sum_{i=1}^{n} \frac{(y_i^{true} - y_i^{pred})^2}{n} \tag{1}$$

From this simple setup, we can start performing training experiments oriented to optimise the learning of the neural network model itself. These experiments are:

- a validation of the three splits made to the dataset (train, validation, test);
- an evaluation of the loss function and a reformulation towards the usage of the Weighted Mean Squared Error (WMSE), used in the subsequent experiments;
- an evaluation of the effect of smoothing applied as post-processing;

These experiments do not focus on fine tuning of hyperparameters such as the *optimizer*, *batch size*, *learning rate*, or *number of layers*; we use the simple setup presented and we will maintain it for all the experiments. Instead, our experiments intend to tackle several aspects that could have important effects on the results, so that we improve the neural model itself, its training and consequent results.

*2) Splitting the data:* At the initial training stages of the project, we used a function to split the whole data into training, validation, and test sets. However, the function had two main problems: the data was randomly split at each training experiment, which is not desirable when evaluating the performance of the model across experiments; and the data was divided as frames instead of animations. In fact, the second problem was worse than initially expected, because of the similarity of temporally near frames of the animations/videos. By splitting random frames of a whole animation, we were having similar frames in the three split sets, so that the loss value was very low, thus it was difficult to generalise the model. Indeed, the model performed really well in the animations from the dataset, but failed at estimating unseen new animations.

This problem was solved by splitting the data as animations instead of frames. Nevertheless, splitting as animations could cause a new problem. Even though the splitting is performed with a commonly used ratio $70/20/10$ (train, validation, and test sets, respectively), if we evaluate the data as frames we obtain a rough division of $37,000$, $10,000$, and $6,000$ frames respectively, but if we evaluate the data as animations the resulting rough division is $148$, $42$, and $22$ animations, respectively. Thus, we saw the possibility of having a split of bad animations that could ”foul” our results (by bad animations we mean animations for example that have no movement on the fingers, or that only have movement on a specific part of the body). Consequently, we ran an experiment to confirm the correct functioning of the split and the validity of the results obtained with it (or otherwise).

We evaluated the loss of five models trained with five different splits (but with the same division ratio). In Figure 16 one can see that the curves of all losses are similar, the error is also quantitatively similar, which indicates that the split of the data is working fine and that the split as animations are not invalidating our results. With that confirmation, we could proceed to improve the training of the neural network model.

*3) Loss Function:* Once we validated the way the data was split we perceived, from the initial results, that we needed to understand the impact on the learning of key aspects of the animations, and of their representation by means of quaternions. Thus, with the following experiment we intended to evaluate whether the variation of values of each quaternion, and whether any of them had more impact than other ones on the results or not. This would allow us to potential improvements in the loss function, in order to specialise it in the learning of the quaternions of the desired joints.

To understand the results of the experiment, we visually show and compare individually for each joint the quaternions ground truth and the quaternions estimated using a trained model with a test loss $= 0,00779$. We can compare the difference between different joints as well. We show figures representing the curves of the values of the quaternion coordinates X,Y,Z together along the animation frames and of W separately, for any joint, both of the ground truth, and of the estimated result. Actually, in this experiment we used animations of the test split which have significant movement of the body and fingers, as we expect to have more salient outcomes, so that differences or errors are more pronounced. We show results for our *Agreeing* animation, which fits this description. The Figures 17a and 17b correspond to the joints $< Head >$ and $< RightArm >$, respectively, of the *Agreeing* animation. In both of them, we see the same phenomenon: the W value has a different behaviour from that of X,Y,Z (indeed, this is why we chose the visual representation). Unlike X,Y,Z, whose values are around 0, those of W are close to 1 (never exceeding it, because the rotations are represented by *unit* quaternions), which is in fact related to the functioning of quaternions. The W coordinate corresponds to the real term of the quaternions, and represents the magnitude of the rotation angle ($\theta$) as in the Formula 2. In terms of visual appearance of the animation, the W affects how large the angle is. Since the quaternions are unitary, $W = 1$ means that the rest of the coordinates will be 0, which means no rotation applied. Then, quaternions with $W >= 0.95$ (for instance) will have a narrower angle than if $W < 0.95$ (unless the only coordinate with value greater than 0 is the axis of the rotation).

$$W = cos(\frac{\theta}{2}) \tag{2}$$

Returning to the two figures, the two joints exhibit different behaviour. The arm has significant more variation than the head (for instance the head range of W is $[0.99, 1]$ and the range of W for the arm is $[0.84, 1]$), and, in fact the quaternion curves for the arm appear to be more precisely estimated than the results of the head, as can be seen by comparing the similarity between the ground truth curves and the predicted ones of both joints.

Thus, from the observation of the plots, and the previous arguments, we have two hypotheses: quaternion curves seem to fit more closely to joints that have more variation; and the W coordinate seems more important than the others in determining the magnitude of the rotation (limiting its error possibilities). We discuss how to tackle each issue next.

Regarding the first concern, it seems that larger variations of some joints are "occupying" most effort of the learning, since they create more error, therefore, improving these joints reduces the error the most. For this reason, one possible strategy to improve the overall learning for each joint, seems to be balancing the importance of all joints depending on the observed range of values. Indeed, the error of joints that have a lot of movement, as the arm, should be taken less into account than those for other joint with observed low variation, like the head, to improve the learning for them. This could be solved by using custom weights in our custom loss function, transforming the MSE into a Weighted Mean Squared Error (WMSE).

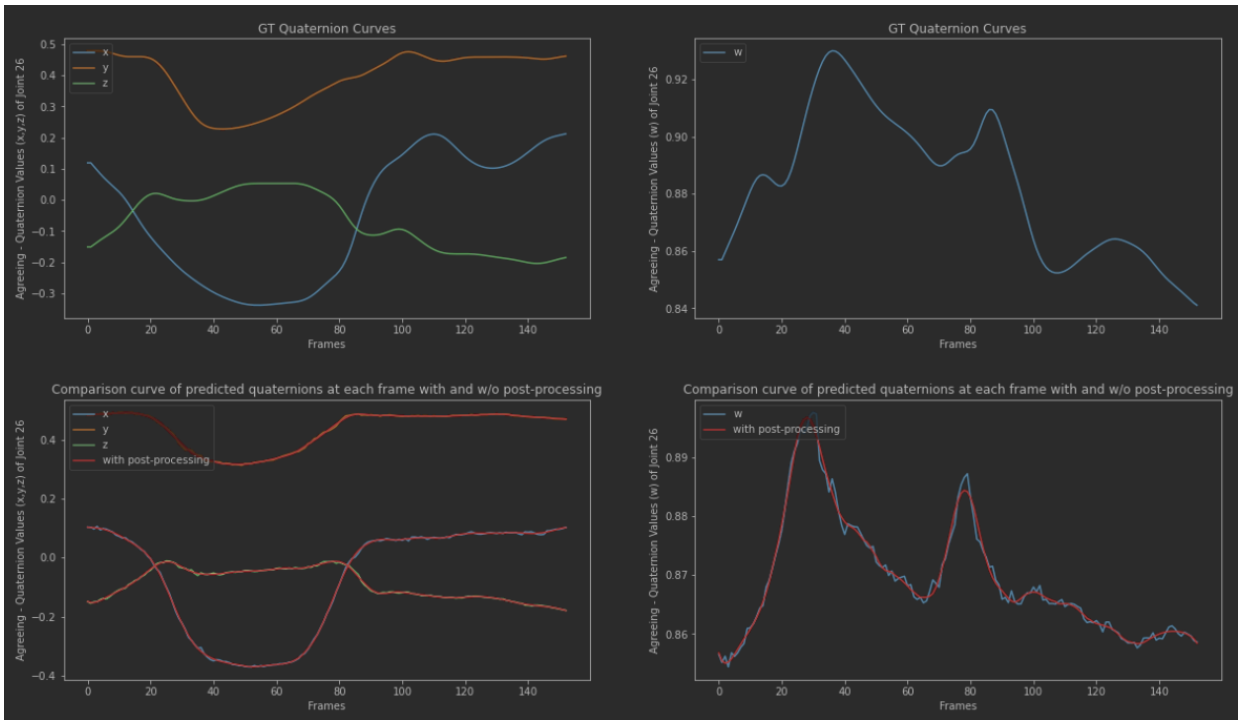
Figure 18 shows a comparison in the training and validation losses obtained from both models, MSE and WMSE, respectively. The former takes equal weights $1.0$, meaning that all the joints are equally important. The latter one sets a custom weight of $0.1$ at the coordinates of the joints: $< LeftArm >$, $< RightArm >$, $< LeftForeArm >$, $< RightForeArm >$, $< LeftHand >$, and $< RightHand >$. The results seem to confirm our hypothesis, since we reduce the error from a Test Loss = $0,00778$ to a Test Loss = $0,0068$.

As for the second concern mentioned, we recall that in Figure 17 we see that the estimated W curve is more similar to the ground truth than the curves of the other coordinates to their ground truth ones. This is aligned with the explanation that W is more important when estimating a quaternion as best as possible than the other coordinates. We have a problem similar to the previous one, there are some data that might be considered in a more important way than other ones. Again, we can tackle this problem with custom weights of the loss function. More precisely, we decrease the weights of the X,Y,Z coordinates to $0.1$, and keep a weight of $1.0$ on W.

The evaluation of this hypothesis is performed as before, in Figure 19 we can see a comparison in the train and validation losses obtained using the previous best model and another with the specified new weights. The results are even better than the previous tests, since we get to reduce the error from a Test Loss = $0,0068$ to a Test Loss = $0,0044$.

*4) Smooth Effect:* Unlike the previous experiments, oriented to improve the model, this test intends to improve the post-processing effect of smoothness in the estimated quaternions, so that we get a better visual appearance of the animations, to be able to qualitatively evaluate them. This evaluation is important, and too much noise might result in making it more difficult the visual assessment of the animations of the skeletons. As seen in the previous Figure 17, the prediction curves tends to be noisy. The main reason is that the estimation of quaternion are done frame-wise, therefore their values are not shared between


(a) Quaternion curves along frames for the joint $< Head >$ from *Agreeing*.




(b) Quaternion curves along frames for the joint $< RightArm >$ from *Agreeing*.

Fig. 17: Initial results when using MSE. Left plots are for the X, Y, and Z coordinates together, and right plots for the coordinate W. Top plots show ground truth, bottom ones estimated values

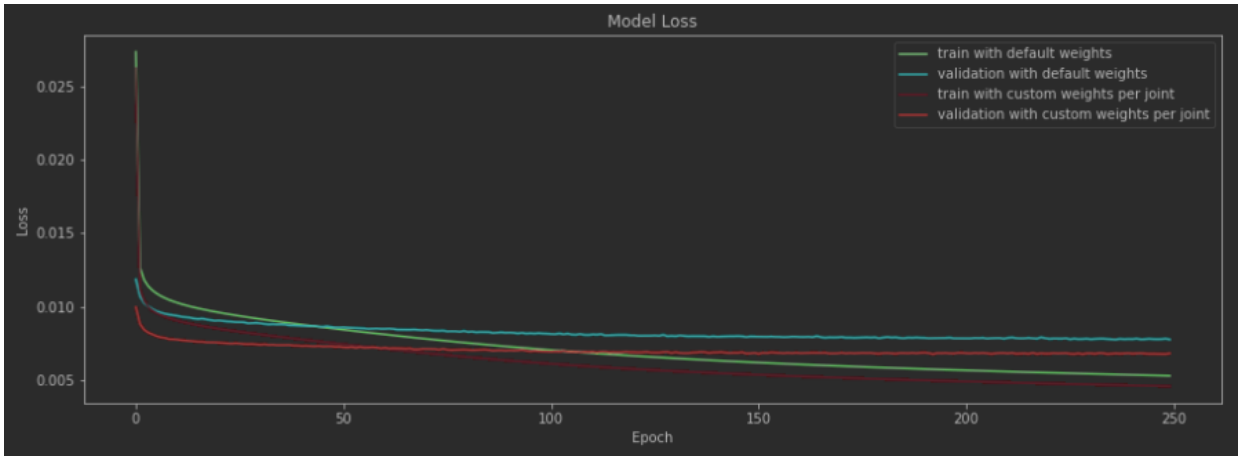



Fig. 18: Train and validation loss for two models using MSE and WMSE, respectively. The green and blue curves correspond to the default MSE loss function. The purple and red curves correspond to a trained model with custom weights per joints in the loss function.

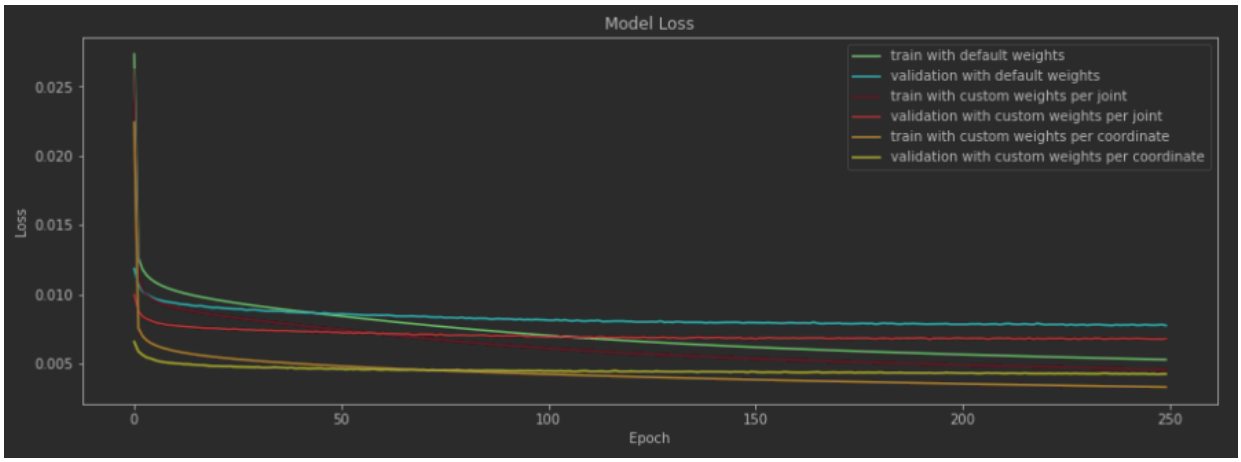
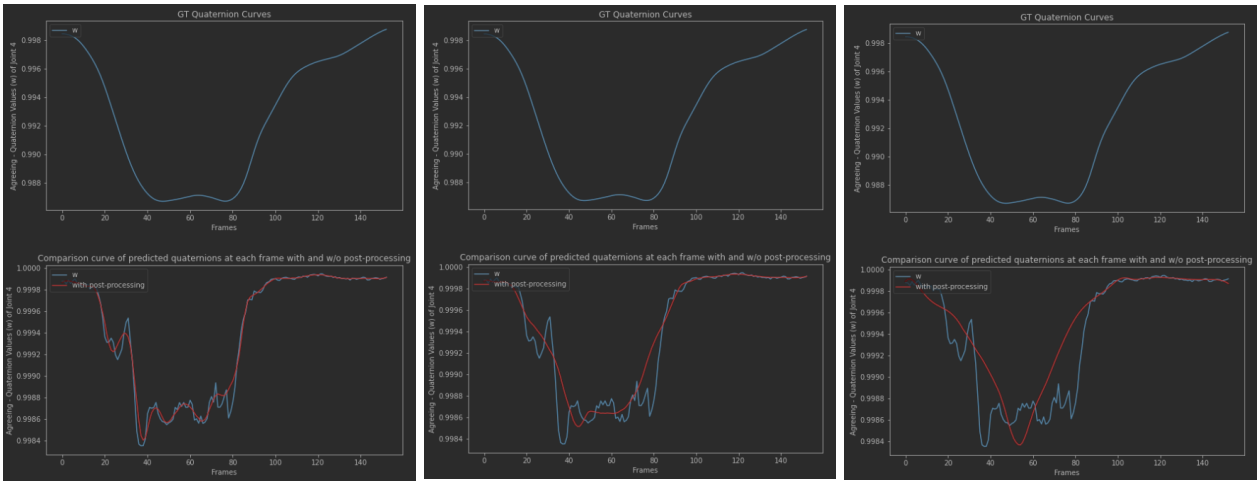



Fig. 19: Train and validation loss curves for three models. The first two models correspond to MSE and joint-WMSE, as in Figure 18. The third one is represented by the orange and yellow curves, whith custom weights per coordinate in the loss function.

time-contiguous frames. Thus, we smooth the quaternion curves using a filter, namely, the Savitzky Golay filter [31]. This algorithm was presented in 1964, and despite being easy to apply it was very popularized, being one of the most widely cited papers in the journal Analytical Chemistry [32]. This filter smooths the data by using a polynomial, which is defined by the *window size* of the data that covers, and the *order* of the polynomial, the higher the order of the polynomial the better the filter fits the data. This experiment focuses on the search of the best parameters that remove the noise, but maintain the important movement represented by maxima and minima in the quaternion curves. We visualise the results, overlapping the filtered and unfiltered quaternions curves.

In this case, since it is a post-processing step, the improvement will not be discussed as a reduction of the loss in the neural model, but by two different techniques, namely, by visually evaluating the noise suppression, and by the reduction of the absolute joint position difference. For simplicity, we leave the explanation and evaluation of the absolute joint position for Section IV. For ease the visual smooth effect of this experiment we will use values from a joint were our estimation maintains some of the form of the correct curve, but with noise error (which is what we are aiming to correct).

With respect to the window size, Figure 20 shows the plot of the W coordinate of the joint $< Spine2 >$. The increase of the window size means that more data is used to fit the smoothed curve. If we use a too small window, we do not use enough data, so that the new curve just follows the original one leaving the noise. But if we use a too big window, the smoothing effect will be too strong and important variations of the curves that represent movement of the joints will be erased. As examples, plot 20a seems to keep more information than needed, and fails at erasing the noise; on the other hand, plot 20c does not achieve a good




(a) Window Size = 15, Order = 3 (b) Window Size = 31, Order = 3 (c) Window Size = 55, Order = 3

(d) Window Size = 31, Order = 1 (e) Window Size = 31, Order = 3 (f) Window Size = 31, Order = 6

Fig. 20: Comparison plot of the ground truth values of the coordinate W from a quaternion through all the frames, and its estimated values. On red, the smooth curve obtained after applying the Savitzky Golay filter. Subplots a) to c) show the effect of the *window size*, and subplots d) to f) show the effect of the *order* at different values.

result, it misses most of the information. Plot 20b shows the result that reduces most the distance error ($window\_size = 31$), erasing most of the noise without losing the form of the initial curve.

As for the order of the polynomial, we evaluated until the $6^{th}$ degree, higher values were not creating any significant smoothing effect on the data. Figures 20d, 20e, and 20f show the changes on the smoothing effect at different order values. In this case, a low order polynomial is not complex enough to represent important maxima and minima of the data. On the other hand, a high order is too complex and falls into replicating the noise. From the tests carried out, it appears that order 3 is the most suitable for our problem.

### *D. Proposed System*

This final subsection is devoted to explain the end-to-end system that has been created to put the work resulting from our proposed approach to use, capturing SL animations to be able to create a database of signs which can feed into a translation system using signing avatars as output.

The application is web-based: it can be shared easily, and users do not need specific system requirements, nor download code on their respective desktops. Therefore, it is coded in JavaScript, which is compatible with models created with *TensorFlow.js*, a version which can use machine learning in JavaScript.

The initial view allows the user to select the desired input between multiple choices. The first design of the system used a webcam stream as input to MediaPipe, as it was oriented to be part of a system to be used interactively by users. In the course of the project, the benefit of using every type of data available as a source to generate SL animations became clear, and

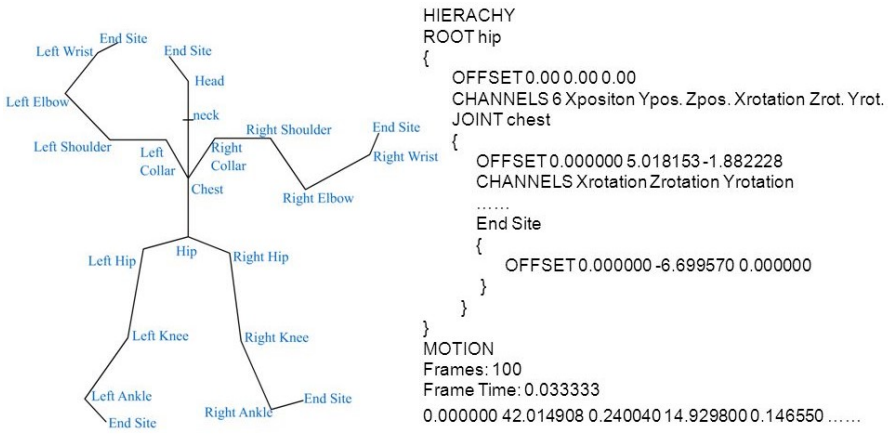



Fig. 21: Example skeleton with its information hierarchically structured in the file (from [8]).

further work was carried out to support different inputs in the end-to-end system, in particular, prerecorded videos. While the steps are the same, this is another logical approach. Since each of the animations built with the application can be exported, we found the idea of being able to load and import directly any animation into the system interesting. This can be used to skip some steps and makes the pipeline of the application more agile and flexible, as well as giving the possibility to continue previous unfinished work.

The Figure 23 shows a screenshot of the edition station where the user can crop the video exactly in the meaningful times of the movement. This is necessary because the application requires user commands to proceed (clicking the ”Capture” button to start the capture), and to communicate with sign language both hands are needed. Therefore, there is a gap between the time the user starts the recording and is prepared to perform the sign. Once the input is prepared, MediaPipe estimates the pose and finger 2D landmarks for the video. Additionally, in this step is where we apply the pre-processes of cleaning or correcting lost frames mentioned in Section III-B4.

Next, the array of landmarks is injected as input of the trained model and we obtain a list of quaternions. The system is prepared to hold a model that is independent of the application. This allows to work separately on improving the accuracy of the model and replace it with the one in use. The resulting animation is then applied to a virtual avatar. As introduced in previous sections, we need to translate the movement from an auxiliary virtual avatar (which is the one we used for training, Kate) to a desired virtual avatar to use (Eva), and this is achieved by using a retargeting technique.

Once the animation is generated by the rotations from the neural network, the next view of the application is another edition station to manage the possible imperfections generated. The design is based on an animation keyframe edition system, with the specificity of looking at the edition system as a complementary tool to the neural network steps and not a replacement, since both steps are important and necessary to create proper sign animations in a scalable way.

The interface provides the user a set of tools such as the hierarchy of the bones, a timeline of the animation, and the values of the positions and quaternions of the bones at each frame. These values can be edited by using the *gizmo* that represents the 3D axes of its transformation space. We will not go more into detail since the specific techniques used in the edition tools fall outside of this project objective.

Once the user is happy with the resulting animation, this is stored in the BioVision Hierarchy (BVH) format file. The BVH format stores the rotations of each joint in hierarchical order per frame at a specified frame rate. This structure is useful to distinguish parent bones and their children. See an example in Figure 21. Also, Figures 22, 23, and 24 show the different views of the application. The application is free-to-use and is available at the following link: *https://webglstudio.org/users/vubieto/SignON-editor/?load=TFM*

## IV. Results and Discussion

In this section we evaluate some of the work presented in this master thesis, beyond what we previously discussed. Indeed, throughout the project, we need to provide both quantitative and qualitative evaluations to ensure the best quality of our results, with respect to our initial goal, capturing animations, towards their use in translation involving signing avatars. Besides

[8]https://player.slideplayer.com/17/5265715/


Fig. 22: View of the input selection menu.

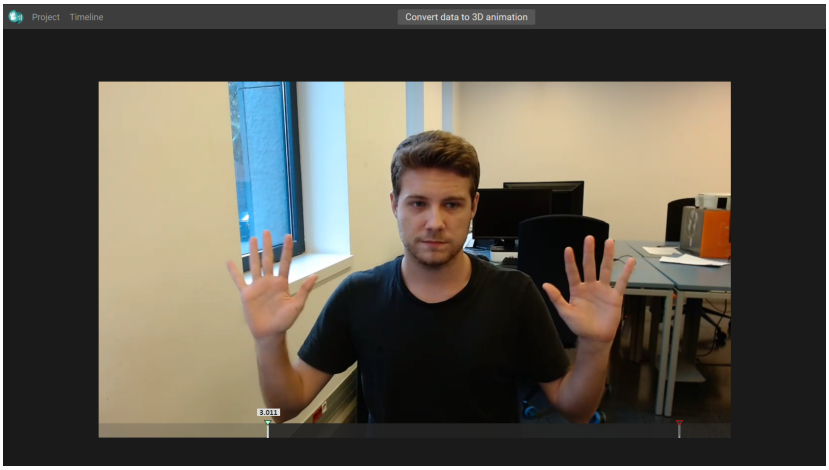



Fig. 23: View of the video cropping station.


Fig. 24: View of the edition station for animations.

the evaluation of results, we discuss as well the techniques used in the evaluation, some borrowed from the literature, some proposed here. Before dealing with the results, we evaluate the performance of our automatic tool to generate a suitable dataset, in two different dimensions, and discuss how we have set some of the hyperparameters we have been using for training the neural network model. Then, we evaluate quantitatively and qualitatively, using the techniques discussed, our estimated results compared with the ground truth. Finally, we discuss that our approach does not reach yet what has been achieved by other approaches, but still has room for improvement to achieve better results.

### *A. Dataset Generator Suitability*

We believed that a quick evaluation of the performance of the dataset generator was needed at the beginning of the process. It should assess how much data is ”wasted”, i.e., is of little use further along the process, and if more efforts to improve the script are required. The evaluation was done in terms of lost data and data correctly retrieved at the time of creating the dataset. The dataset used in the project consists of 106 animations (212 if we count that each one was mirrored). At the creation of the dataset, we retrieved some $90\%$ of the frames, leaving us with a dataset of $52,207$ frames. We checked that the lost frames were poses that could not be interpreted by the pose recogniser, such as occlusion of hands with the body, or body rotations in the *up* axis.

Additionally, we tried to make sure that the dataset did not contain wrong data that would be propagated through all the work. To do so, we visually evaluated of the landmarks retrieved from MediaPipe. Indeed, we use MediaPipe to capture a virtual 3D scene, which is not its intended use. Thus, we needed to check whether the results obtained from MediaPipe correspond to the source animations. We did this through a visual tool showing the landmarks retrieved in parallel to the animation being captured.

Both evaluations seemed to offer reassuring results, and we proceeded with the project work.

### *B. Neural Network Model Loss*

After evaluating the estimated animations, it is worth showing the best parameters and results obtained from the training of the neural network model.

During all the work we have used the Stochastic Gradient Descent (SGD) as our optimizer, since it has been performing better than others for this project problem. This might be because of the "small" dataset, which many times causes a neural network model to overfit at training. Therefore, by selecting small random sets of data, it helped the neural network to learn little by little but without overfitting to the training data. The batch size was settled at 32, since lower values provoked overfitting at learning, highly decreasing the error for the train data, but maintaining a constant loss for the validation data. For the learning rate we used the build exponential decay function of TensorFlow, but it did not seem to affect very much the results so it was set to initial learning rate of $0.01$ with a decay of $0.97$ each $100.000$ steps, which are standard values for these parameters. Finally, we applied the same strategy to the number of layers and their weights. By increasing the number of layers, the model was learning from very specific relations, which far from helping to generalise, probably belonged to the specificities of training data, causing the overfitting.

### *C. Quantitative Error*

As stated in [4], "The application targeted in this work constitutes a very unique research problem. For this reason, no common dataset, nor common baseline technology is available that could be used for evaluation for the proposed system.". To allow as much as possible for comparison with related work, we use the absolute framewise similarity between the generated data and the real motion captured data, as the paper does. Unlike the paper authors, who estimated directly 3D positions, we use quaternions, and thus we implement a 3D scene where we display two different animations using a common skeleton: one that performs the ground truth animation; and another one whose movements are based on the predicted quaternions. The most similar measure to the paper, to try to compare results, seems to be to provide the error distance between the bones of the animations. All measurements are evaluated in centimeters. For simplicity, in Table I we show different distance error measures for the *Agreeing* animation, which, as indicated before, is representative because it contains significant movement for all joints, and we can discuss more clearly the obtained results.

The first thing that we can appreciate is that the error is accumulated along the hierarchy of the skeleton joints. This effect also appears in [4], which makes sense since the joints are connected via a hierarchical structure of parents and children. Therefore, the error from the parents will also affect the distance error of its child bones. Even so, this measure works to understand quantitatively how good the results are. In the case of the animation *Agreeing*, the joints from the spine column have an error between $1cm$ and $4cm$. However, the last bones of the hierarchy (the fingers) have an error between $10cm$ and $16cm$. It is also important to note that the best estimated joints seem to be those corresponding to the arms, which have an error between $4cm$ and $7cm$. As mentioned in III-C3, these joints have a big variation and are those carrying more movement; therefore it seems from the results that the quality of the estimates is better for these joints than for the other ones.

In order to see clearly which joints contain more error than the others, we propose another quantitative measurement which evaluates each joint error independent from the hierarchical error. This is the error in angles between the ground truth quaternions and the predicted ones. To measure this, we convert each quaternion into its three Euler angles (yaw, pitch, roll), and compute the distance between the prediction and the correct angles; these angles are more easily interpreted than quaternion components. To see the difference, we will show the error from one of the joints with largest mean error from Table I which is $< LeftHandPinky3 >$ and its error in angles compared to other joints.

In Figure 25 we can see that the angles are more accurate for $< LeftHandPinky3 >$ than for $< Head >$, despite having more distance error than $< Head >$. This shows us that the head was not too well predicted in this animation, which is not possible to conclude from Table I. This evaluation also helps us to understand which joints the model finds harder to estimate than other ones. In fact, this also show us that the length of each joint is a very important factor to take into account, since small rotations in a large bone will represent larger distance in the world space than the same rotation in a short bone.

### *D. Qualitative Error*

Despite the quantitative evaluation give us an overview, and detail, of the overall performance of the results, the main focus of this work is that the animations have the best possible visual quality, and to do so we need to evaluate the results qualitatively.

| JOINT | MIN Error | MAX Error | MEAN Error | TOTAL Error |
|---|---|---|---|---|
| Hips | 0.000 cm | 0.000 cm | 0.000 cm | 0.000 cm |
| Spine | 0.615 cm | 1.921 cm | 1.011 cm | 154.642 cm |
| Spine1 | 0.955 cm | 4.045 cm | 2.179 cm | 333.347 cm |
| Spine2 | 1.293 cm | 6.590 cm | 3.652 cm | 558.700 cm |
| Neck | 1.424 cm | 9.302 cm | 5.577 cm | 853.223 cm |
| Head | 1.000 cm | 9.013 cm | 5.612 cm | 858.648 cm |
| LeftShoulder | 1.487 cm | 8.722 cm | 5.202 cm | 795.975 cm |
| LeftArm | 0.212 cm | 6.233 cm | 2.947 cm | 450.925 cm |
| LeftForeArm | 2.563 cm | 12.496 cm | 6.841 cm | 1046.668 cm |
| LeftHand | 3.544 cm | 20.978 cm | 12.686 cm | 1940.915 cm |
| LeftHandThumb1 | 1.964 cm | 21.298 cm | 12.372 cm | 1892.966 cm |
| LeftHandThumb2 | 1.949 cm | 21.936 cm | 12.672 cm | 1938.778 cm |
| LeftHandThumb3 | 1.620 cm | 23.767 cm | 13.194 cm | 2018.708 cm |
| LeftHandIndex1 | 2.041 cm | 26.086 cm | 14.047 cm | 2149.260 cm |
| LeftHandIndex2 | 1.357 cm | 27.763 cm | 14.545 cm | 2225.415 cm |
| LeftHandIndex3 | 1.044 cm | 28.268 cm | 14.878 cm | 2276.302 cm |
| LeftHandMiddle1 | 2.427 cm | 26.425 cm | 14.554 cm | 2226.751 cm |
| LeftHandMiddle2 | 2.249 cm | 27.765 cm | 15.048 cm | 2302.289 cm |
| LeftHandMiddle3 | 2.206 cm | 27.721 cm | 15.332 cm | 2345.851 cm |
| LeftHandRing1 | 3.832 cm | 26.747 cm | 15.120 cm | 2313.425 cm |
| LeftHandRing2 | 3.523 cm | 27.379 cm | 15.703 cm | 2402.598 cm |
| LeftHandRing3 | 3.374 cm | 26.579 cm | 16.092 cm | 2462.143 cm |
| LeftHandPinky1 | 5.482 cm | 26.684 cm | 15.656 cm | 2395.317 cm |
| LeftHandPinky2 | 5.484 cm | 26.977 cm | 16.307 cm | 2494.935 cm |
| LeftHandPinky3 | 5.278 cm | 26.359 cm | 16.674 cm | 2551.066 cm |
| RightShoulder | 1.321 cm | 9.266 cm | 5.511 cm | 843.116 cm |
| RightArm | 1.725 cm | 6.986 cm | 4.309 cm | 659.219 cm |
| RightForeArm | 2.841 cm | 10.142 cm | 5.619 cm | 859.769 cm |
| RightHand | 1.970 cm | 17.458 cm | 10.302 cm | 1576.228 cm |
| RightHandThumb1 | 2.354 cm | 18.462 cm | 10.719 cm | 1640.068 cm |
| RightHandThumb2 | 1.720 cm | 19.230 cm | 11.182 cm | 1710.816 cm |
| RightHandThumb3 | 1.042 cm | 20.741 cm | 11.939 cm | 1826.596 cm |
| RightHandIndex1 | 1.474 cm | 22.444 cm | 12.600 cm | 1927.756 cm |
| RightHandIndex2 | 1.865 cm | 23.832 cm | 13.648 cm | 2088.094 cm |
| RightHandIndex3 | 2.684 cm | 24.664 cm | 14.776 cm | 2260.792 cm |
| RightHandMiddle1 | 1.660 cm | 22.758 cm | 12.727 cm | 1947.229 cm |
| RightHandMiddle2 | 1.212 cm | 24.211 cm | 13.779 cm | 2108.184 cm |
| RightHandMiddle3 | 1.865 cm | 25.103 cm | 14.844 cm | 2271.169 cm |
| RightHandRing1 | 2.120 cm | 23.036 cm | 12.916 cm | 1976.135 cm |
| RightHandRing2 | 1.522 cm | 24.089 cm | 13.705 cm | 2096.805 cm |
| RightHandRing3 | 0.821 cm | 24.571 cm | 14.518 cm | 2221.301 cm |
| RightHandPinky1 | 2.188 cm | 22.551 cm | 12.847 cm | 1965.559 cm |
| RightHandPinky2 | 2.124 cm | 23.547 cm | 13.730 cm | 2100.684 cm |
| RightHandPinky3 | 2.703 cm | 23.929 cm | 14.422 cm | 2206.627 cm |

TABLE I: Distance errors per joint between the prediction and the ground truth for the *Agreeing* animation.

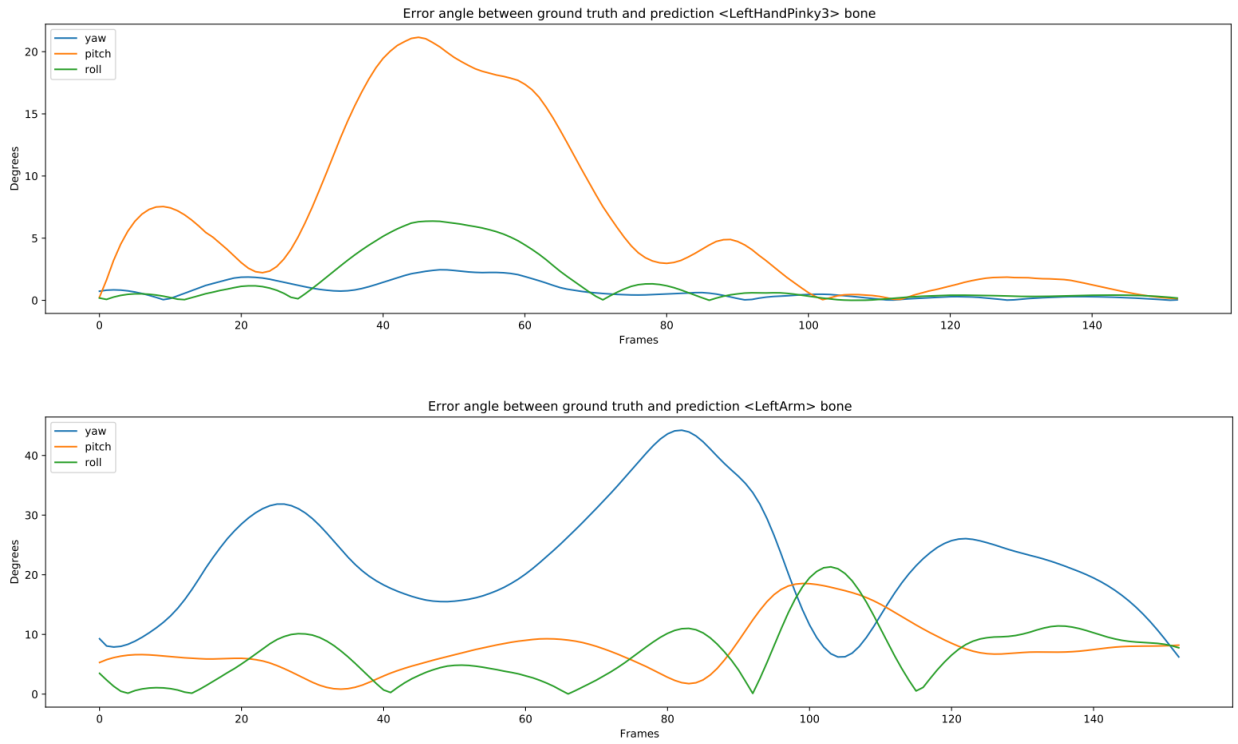



Fig. 25: Roll, pitch, yaw values in degrees of joints $< LeftHandPinky3 >$ and $< Head >$ for all frames in *Agreeing*.

On one hand, following the technique proposed in [4], Figure 26 compares curves of the ground truth absolute positions of joints and of their predictions along frames. This helps us to understand better than before the performance for each joint, by comparing the form of the true curve with its estimate, because the error from the hierarchy only leads to a common displacement without modifying the curve form. Also, in Figure 26 we can see again the conclusion discussed in the quantitative evaluation subsection, that is that the $< LeftHandPinky3 >$ prediction seems to fit better the true result, than the $< Head >$ one, despite having more distance error in terms of centimeters (see axes from the plots). This evaluation tells us that if we visually compare both animations, $< LeftHandPinky3 >$ will be more similar to the ground truth than $< Head >$. Additionally, if we compared $< LeftHandPinky3 >$ with $< LeftHandPinky1 >$ we would see the same curves, but with less error in centimeters between the estimation and the ground truth.

On the other hand, a qualitative evaluation during this project was made by using the visualization tool presented. We uploaded both the ground truth animations and the estimated ones side by side, and compared their similarities/discrepancies. As hinted at in the quantitative evaluation, the overall movement of the body is maintained, but it fails at estimating correctly some bones such as the fingers. Even though we have been able to show some similar overall behaviours, the current trained model does not satisfactorily fit the fingers to recreate prominent hand gestures such as clenching the fist, or pointing with a single finger. The weight of such bones needs to be improved towards a more satisfactory model. Figure 27 shows the effect mentioned: on the left, the skeleton has the pinky finger ”contracted”, however our estimation does not replicate well this state.

### *E. Discussion*

From the evaluation, our approach still falls short at reaching the results from Brock et al. [4]. In a way, this makes sense since they are using Inverse Kinematics to move their skeleton to an exact position. Consequently, they will most likely get results very close the desired location. On the other hand, our approach completely skips the need of that extra algorithm and we are able to retrieve a similar animation from a list of 2D landmarks. Another error found is the one from Figure 28, where the estimated hands do not replicate correctly the rest position of the true hands, which seems to be caused by lack of depth information at the training phase. Indeed, the hands pose is correct, but the neural network model did not learn to predict fully the depth.

So the evaluation tells us that we are still far to accomplish quality from Brock [4]. Nevertheless, as we have shown with the experiments presented and the discussions in the previous evaluations, our implementation still has room for improvement. There are further important experiments that need to be performed to keep understanding better the details of our approach and refine it to maximise its full potential.

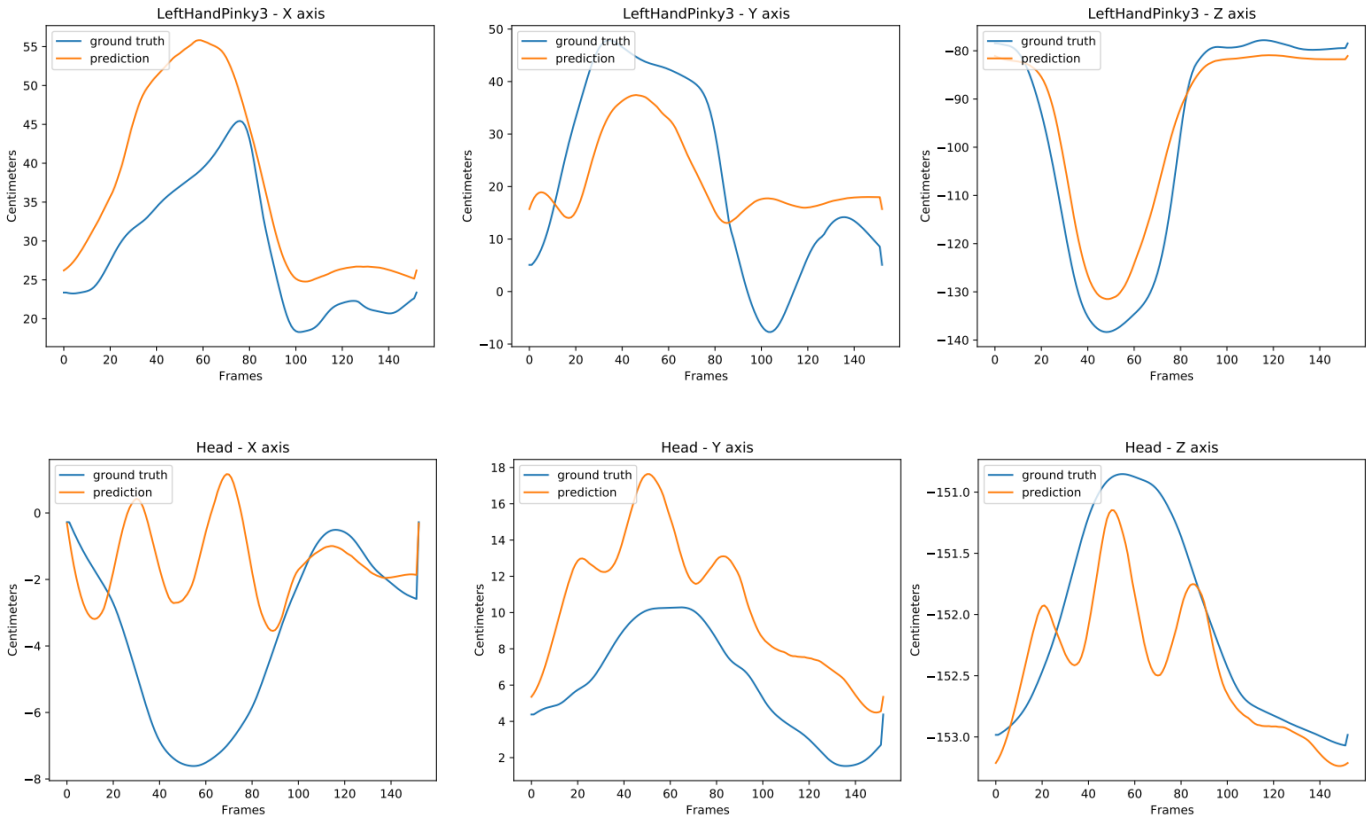



Fig. 26: Comparison of inferences (orange curves) to their target (blue curves), where the joint and axis are noted above. We examine the curves of the joints with the largest absolute distance (in centimeters) along all the axis.

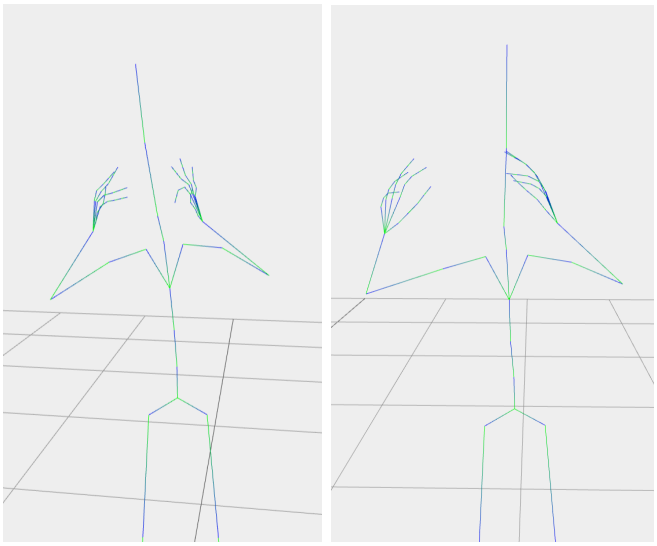


Fig. 27: Comparison frames between poses: (left) ground truth and (right) prediction. Failing at estimating correctly the positions of fingers.

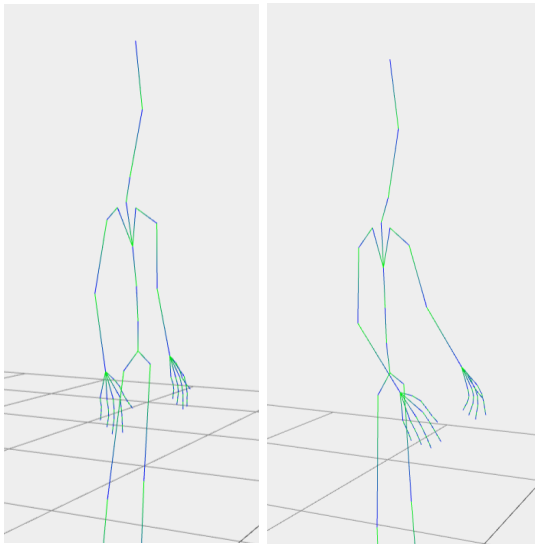


Fig. 28: Comparison frames between poses: (left) ground truth and (right) prediction. Failing at estimating correctly the depth of the hands.

## V. Conclusions and Future Work

In this work, we have proposed solutions to a number of challenges addressing the synthesis of animation for sign language. At this stage, it seems convenient to recall the research challenges identified by Bragg et al. [3] with respect to **Avatars and Computer Graphics** which we mentioned in the Introduction, and discuss them after having presented the results of this thesis. The challenges are: *Uncanny Valley*, *Realistic Transitions*, *Modeling Modulations*, *Finding Model Holes* and the *Public Motion-Capture Datasets*.

We have presented a methodology for creating *Public Motion-Capture Datasets* with a novel approach on the information used, where the identity of the contributors can be preserved, as only the animations need to be saved. Our approach is accessible and free-to-use, which allows each user customise it depending on its intended use. Indeed, the dataset created is generated automatically by means of a tool, which is publicly available. Researchers can customise the tool to generate a dataset appropriate for their research objectives. Additionally, the proposed use of quaternions does not substitute but complements already existing datasets, therefore it can be used to add a new information layer to their data. It also facilitates the implementation of the results in actual animations, which is not always straightforward when dealing with 3D positions depending on the skeleton of the virtual avatar.

We next used the dataset to train a neural network model based on quaternions estimation, introducing a novel approach for the problem of animation synthesis. This work has been evaluated and compared with state of the art papers that tackle the same problem with another approach. Although our results do not seem to match the quality of theirs yet, first steps towards the refining of the model were presented along with several experiments. In fact, the estimated animations preserve most of the movement of the ground truth data, which indicates that it can be improved with further work along the lines proposed.

Finally, an end-to-end system was also presented, consisting on a available tool to estimate animations from videos. When high quality animations are achieved, this should help in addressing another issue, namely, *Realistic Transitions* of the animations, as the technique is data based. If and when enough SL data are collected, it might be possible to address the aspects *Modeling Modulations* and *Finding Model Holes*. The *Uncanny Valley* aspect, which we are addressing within the SignON European project mentioned in the Introduction, would require a separate and lengthy discussion. On one hand, it is very important that the research heavily counts with the participation of signers [33], and understand well the Uncanny Valley concept[9].

At this stage, on the basis of the work carried out, we intend to put into practice several strategies, that we believe should have a positive impact on the results. First, the addition of a step for estimating depth information could improve the estimation of the rotation axis of the quaternions. Second, the evaluation on quaternions seems to clearly support that more work is needed to understand better which data has more effect at the training process. Finally, we plan to introduce the NMFs in our end-to-end system, as they are features that are essential in the synthesis of animations for sign language. In the end, it is the evaluation by signers of the SL animations which will decide the usefulness of our contributions, and for this evaluation both MFs and NMFs need to be included.

## Acknowledgment

*I would like to thank everyone who has been involved in this project, both directly and indirectly. Firstly, I could not have got this far without the knowledge and kindness of Josep, who is always there to teach me and help me improve in every possible way. I would like to thank my family, who are always there to love me unconditionally no matter what I choose to do or the path I follow in my life. I would also like to thank Coloma and Gloria, who agreed to guide me and help me throughout the project despite the difficulties it posed. And finally, I would like to thank the people close to me, Pablo, Eva, Arnau, Jaume, Mar, Alex, Jon, David, the GTI group and many other people that gave me encouragement and strength to keep working every day and make my life better. Without any of them, this would not have been possible.*

## References

[1] H. Kacorri, "Data-driven synthesis and evaluation of syntactic facial expressions in american sign language animation," Ph.D. dissertation, The Graduate Center, City University of New York, 03 2016.

[2] L. C. Quandt, A. Willis, M. Schwenk, K. Weeks, and R. Ferster, "Attitudes toward signing avatars vary depending on hearing status, age of signed language acquisition, and avatar type," *Frontiers in Psychology*, vol. 13, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fpsyg.2022.730917

[3] D. Bragg, O. Koller, M. Bellard, L. Berke, P. Boudreault, A. Braffort, N. Caselli, M. Huenerfauth, H. Kacorri, T. Verhoef, C. Vogler, and M. Ringel Morris, "Sign language recognition, generation, and translation: An interdisciplinary perspective," in *The 21st International ACM SIGACCESS Conference on Computers and Accessibility*, ser. ASSETS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 16–31. [Online]. Available: https://doi.org/10.1145/3308561.3353774

[4] H. Brock, F. Law, K. Nakadai, and Y. Nagashima, "Learning three-dimensional skeleton data from sign language video," *ACM Trans. Intell. Syst. Technol.*, vol. 11, no. 3, apr 2020. [Online]. Available: https://doi.org/10.1145/3377552

[5] I. Grishchenko, V. Bazarevsky, A. Zanfir, E. G. Bazavan, M. Zanfir, R. Yee, K. Raveendran, M. Zhdanovich, M. Grundmann, and C. Sminchisescu, "Blazepose ghum holistic: Real-time 3d human landmarks and pose estimation," 2022. [Online]. Available: https://arxiv.org/abs/2206.11678

[6] I. Murtagh, "A linguistically motivated computational framework for irish sign language," Ph.D. dissertation, Trinity College Dublin. School of Linguistic Speech & Comm Sci. C.L.C.S., 2019.

[9]The Uncanny Valley is a not completely precise concept related to the difficulty of creating characters which are realistic but do not look "creepy". The most important factor driving the effect seems to be the mismatch(es) the user can perceive, that impact on how the users unconsciously respond. See the 2015 review [34], or [35].

[7] L. Naert, C. Larboulette, and S. Gibet, "A survey on the animation of signing avatars: From sign representation to utterance synthesis," *Computers & Graphics*, vol. 92, pp. 76–98, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0097849320301370

[8] ——, "Motion synthesis and editing for the generation of new sign language content," *Machine Translation*, vol. 35, no. 3, pp. 405–430, 2021. [Online]. Available: https://doi.org/10.1007/s10590-021-09268-y

[9] H. Kacorri, "A survey and critique of facial expression synthesis in sign language animation," City University of New York, https://academicworks.cuny.edu/gc_cs_tr/403, Academic Works Tr-2015001, 2015.

[10] W. C. Stokoe Jr, "Sign language structure: An outline of the visual communication systems of the american deaf," *Journal of deaf studies and deaf education*, vol. 10, no. 1, pp. 3–37, 2005.

[11] R. Battison and E. Baird, *Lexical Borrowing in American Sign Language*. Linstok Press, 1978. [Online]. Available: https://books.google.es/books?id=7-O7AAAAIAAJ

[12] W. C. Stokoe, D. C. Casterline, and C. G. Croneberg, *A dictionary of American Sign Language on linguistic principles*. Linstok Press, 1976.

[13] S. Prillwitz and H. Z. für Deutsche Gebärdensprache und Kommunikation Gehörloser, *Hamnosys: Version 2.0; hamburg notation system for sign languages; an introductory guide*. Signum-Verlag, 1989.

[14] V. Sutton, *Sign writing for everyday use*. Sutton Movement Writing Press, 1981.

[15] J. Glauert and R. Elliott, "Extending the sigml notation; a progress report," in *Proceedings of the Second International Workshop on Sign Sanguage Translation and Avatar Technology (SLTAT)*, 2011.

[16] M. Kipp, A. Heloir, and Q. Nguyen, "Sign language avatars: Animation and comprehensibility," in *Intelligent Virtual Agents*, H. H. Vilhjálmsson, S. Kopp, S. Marsella, and K. R. Thórisson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 113–126.

[17] L. Naert, C. Larboulette, and S. Gibet, "LSF-ANIMAL: A motion capture corpus in French Sign Language designed for the animation of signing avatars," in *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020, pp. 6008–6017. [Online]. Available: https://aclanthology.org/2020.lrec-1.736

[18] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C. Chang, M. G. Yong, J. Lee, W. Chang, W. Hua, M. Georg, and M. Grundmann, "Mediapipe: A framework for building perception pipelines," *CoRR*, vol. abs/1906.08172, 2019. [Online]. Available: http://arxiv.org/abs/1906.08172

[19] Z. Cao, G. Hidalgo, T. Simon, S. Wei, and Y. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 43, no. 01, pp. 172–186, jan 2021.

[20] A. Heloir and F. Nunnari, "Toward an intuitive sign language animation authoring system for the deaf," *Universal Access in the Information Society*, vol. 15, no. 4, pp. 513–523, 2016. [Online]. Available: https://doi.org/10.1007/s10209-015-0409-0

[21] H. Brock and K. Nakadai, "Deep JSLC: A multimodal corpus collection for data-driven generation of Japanese Sign Language expressions," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), May 2018. [Online]. Available: https://aclanthology.org/L18-1670

[22] B. Dariush, M. Gienger, B. Jian, C. Goerick, and K. Fujimura, "Whole body humanoid control from human motion descriptors," in *2008 IEEE International Conference on Robotics and Automation*, 06 2008, pp. 2677 – 2684.

[23] "Biovision wikipedia," https://en.wikipedia.org/wiki/Biovision_Hierarchy, accessed: 2022-09-01.

[24] B. Dariush, M. Gienger, B. Jian, C. Goerick, and K. Fujimura, "Whole body humanoid control from human motion descriptors," in *In Proceedings of IEEE International Conference on Robotics and Automation (ICRA '08*, 2008, pp. 2677–2684.

[25] "Mixamo webpage," https://www.mixamo.com, accessed: 2022-09-01.

[26] "Hearai, mediapipe vs openpose," https://www.hearai.pl/post/14-openpose/, accessed: 2022-09-01.

[27] "Blender webpage," https://www.blender.org, accessed: 2022-09-01.

[28] "Mediapipe's holistics webpage," https://google.github.io/mediapipe/solutions/holistic, accessed: 2022-09-01.

[29] "Threejs github repository," https://github.com/mrdoob/three.js, accessed: 2022-09-01.

[30] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures," *Analytical Chemistry*, vol. 36, pp. 1627–1639, 1964.

[31] ——, "Smoothing and differentiation of data by simplified least squares procedures." *Analytical Chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964. [Online]. Available: https://doi.org/10.1021/ac60214a047

[32] C. K. Larive and J. V. Sweedler, "Celebrating the 75th anniversary of the acs division of analytical chemistry: A special collection of the most highly cited analytical chemistry papers published between 1938 and 2012," *Analytical Chemistry*, vol. 85, no. 9, pp. 4201–4202, 2013, pMID: 23647149. [Online]. Available: https://doi.org/10.1021/ac401048d

[33] V. Krausneker and S. Schügerl, "Best practice protocol on the use of sign language avatars," University of Vienna, https://avatar-bestpractice.univie.ac.at/en/english/, Tech. Rep., 2021.

[34] J. Kätsyri, K. Förger, M. Mäkäräinen, and T. Takala, "A review of empirical evidence on different uncanny valley hypotheses: support for perceptual mismatch as one road to the valley of eeriness," *Frontiers in Psychology*, vol. 6, 2015. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fpsyg.2015.00390

[35] M. Shin, S. J. Kim, and F. Biocca, "The uncanny valley: No need for any further judgments when an avatar looks eerie," *Computers in Human Behavior*, vol. 94, pp. 100–109, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0747563219300251