Sign Language Translation Mobile Application and Open Communications Framework

Deliverable 5.3: Interactive Co-creation Web-based Platform for Learning from User Input

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 101017255

Project Information
Project Number: 101017255
Project Title: SignON: Sign Language Translation Mobile Application and Open Communications Framework
Funding Scheme: H2020 ICT-57-2020
Project Start Date: January 1st 2021

Deliverable Information
Title: Interactive Co-creation Web-based Platform for Learning from User Input
Work Package: WP 5 - Target Message Synthesis
Lead beneficiary: UPF
Due Date: 17/12/2021
Revision Number: V0.2
Authors: Víctor Ubieto, Pablo L. García, Josep Blat
Dissemination Level: Public
Deliverable Type: Demonstrator

Overview: This deliverable reports on the first prototype of a tool for capturing, editing and storing SL animations, that has been conceived and implemented by UPF-GTI. It presents the context of this deliverable and the work within the the SignON project and the SL synthesis tasks, points out at some significant aspects of the current state of the art in animating virtual signers, discusses background work

and presents key choices and aspects of the tool, as well as the initial expert evaluation, and the reformulation of the interface. It also indicates the near future plans.

Revision History

Version #	Implemented by	Revision Date	Description of changes
V0.1	Víctor Ubieto	30/11/2021	Version for internal review by project partners
V0.2	Víctor Ubieto	15/12/2021	Version after the review by partners

The SignON project has received funding from the European Union's Horizon 2020 Programme under Grant Agreement No. 101017255. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the SignON project or the European Commission. The European Commission is not liable for any use that may be made of the information contained therein.

Approval Procedure

Version #	Deliverable Name	Approved by	Institution	Approval Date
V0.1	D5.3	Aoife Brady Shaun O'Boyle	DCU	06/12/2021 09/12/2021
V0.1	D5.3	Andrea Cavallaro Marco Giovanelli	FINCONS	01/12/2021
V0.1	D5.3	Vincent Vandeghinste	INT	02/12/2021
V0.1	D5.3	Adrián Nuñez Marcos Olatz Perez de Viñaspre	UPV/EHU	02/12/2021
V0.1	D5.3	John J O'Flaherty	MAC	01/12/2021
V0.1	D5.3	Santiago Egea Gómez	UPF	03/12/2021
V0.1	D5.3	Irene Murtagh	TU Dublin	10/12/202x
V0.1	D5.3	Karim Dahdah	VRT	10/12/2021
V0.1	D5.3	Mathieu De Coster	UGent	06/12/2021
V0.1	D5.3	Jorn Rijckaert	VGTC	10/12/2021
V0.1	D5.3	Anthony Ventresque	NUID UCD	10/12/2021
V0.1	D5.3	Henk van den Heuvel, Louis ten Bosch	RU	01/12/2021
V0.1	D5.3	Catia Cucchiarini	TaalUnie (NTU)	10/12/2021

V0.1	D5.3	Tim Van de Cruys	KU Leuven	12/12/2021
V0.1	D5.3	Davy Van Landuyt	EUD	08/12/2021
V0.1	D5.3	Mirella De Sisto	TiU	02/12/2021

Acronyms

The following table provides definitions for acronyms and terms relevant to this document.

Acronym	Definition
SL	Sign Language
LSC	Llengua de Signes Catalana, Catalan Sign Language
MoCap	Motion Capture
BML	Behavior Markup Language
SiGML	Signing Gesture Markup Language
ML	Machine Learning
MF	Manual Feature
NMF	Non-Manual Feature
BVH	Biovision Hierarchy, a widely used character animation file format

Table of Contents

1. Introduction

This deliverable D5.3, *Interactive co-creation web-based platform for learning from user input* due in M12 of the project is framed within WP5 of the project, *Target Message Synthesis* and more specifically, within WP5.T2, *Developing an interactive system of learning from user generated signed content*. The most visible outcome of SignON will be a translation app, as illustrated in the following image. WP5 is in charge of synthesising the output, which can be text, speech or a Sign Language utterance mediated by a 3D virtual character (aka avatar). D5.3 is framed in the generation of the virtual signer (aka signing avatar).

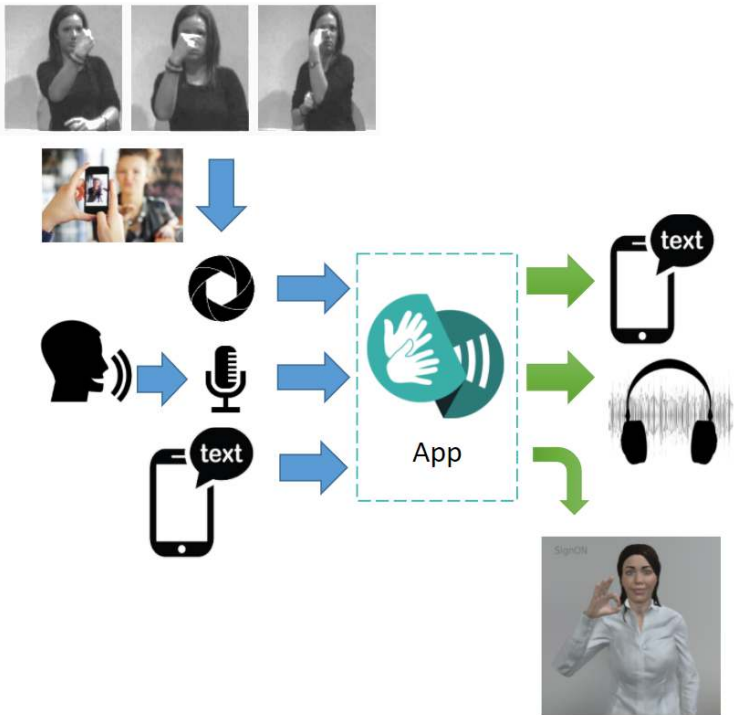

Fig 1: Diagram of the SignON app

Sign language (SL) synthesis translates any given textual or otherwise encoded representation of an intended output message into a SL representation that is generated through the movements of a 3D virtual character. It is worth recalling that our approach for the generation of the virtual signer uses a web based interactive 3D visualisation. We provide an example[1] of the intended output which shows an interactive visualisation of our current virtual character, EVA.

[1] (https://webglstudio.org/latest/player.html?url=fileserver%2Ffiles%2Fvictor%2Fprojects%2FEva_SSS.scene.json)

Fig 2: Still picture of the Virtual Signer

More specifically, as illustrated by the following image, our pipeline synthesis starts from a ***Sign_A*** (for more detail see D5.4, *Sign language-specific lexicon and structure (Sign_A)*) representation of the target message, which we translate to ***extended BML*** (*Behaviour Markup Language*, a specification used by researchers in conversational virtual characters), and render it via a BML *realiser*. More details about BML (and alternative descriptions of SLs) are provided in the companion deliverable D5.7. *A planner for translating from Sign_A to BML-based script*, due in M12 of the project, as is this deliverable.

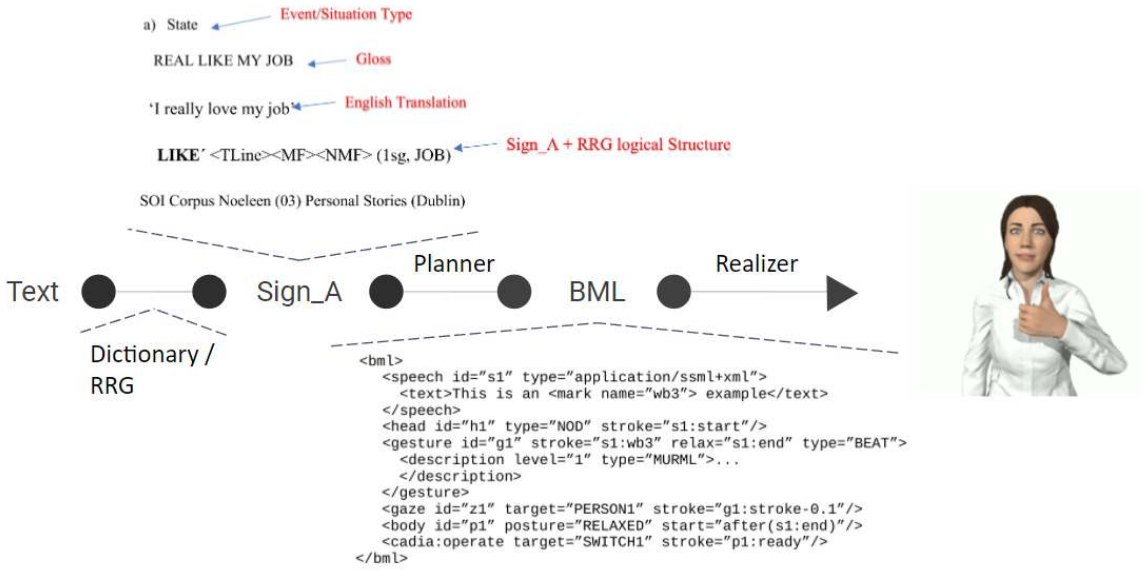

Fig 3: Diagram representing the SL synthesis pipeline

In plain terms, the task of synthesising the Virtual Signer can be broken down into three interrelated aspects:

- Creating a representative avatar
- Making the avatar visually realistic
- Applying natural and realistic animations

Broadly speaking, WP5.T1, *Co-designing a personalisable virtual animated signer*, is devoted to the first aspect, WP5.T5, *Real-time synthesis and delivery of target SL*, to the second aspect, while this deliverable is related to the third aspect. There is a background task for the generation of the virtual signer, discussed in the companion deliverable D5.7 already mentioned, WP5.T4, *Development of a planner for translating from Sign_A representation to BML-based script*.

WP5.T2, *Developing an interactive system of learning from user generated signed content* and this deliverable mainly addresses two of the challenges on the specific issue *Avatars & Computer Graphics* identified in Bragg et al.'s recent interdisciplinary survey and perspective on SL research and challenges[2], namely, realistic transitions and (scarcity of) public motion-capture datasets.

This deliverable reports on the **first prototype of a tool for capturing, editing and storing SL animations**; these animations will be used to drive the virtual character, in order to fulfill the target of producing natural and realistic SL animations. As it is a web based tool, it is available through a link: https://webglstudio.org/projects/signon/animations_editor/

2. Brief Pointers Towards the State of the Art

Naert et al.'s 2020 survey on animating signing avatars[3] 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* synthesis and *utterance* synthesis, as well as existing signing avatars. The paper focuses on the so-called *manual features* (MFs): hand configuration, hand placement, and hand orientation. Kacorri 2015[4] 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.

It is worth mentioning the work by Heloir and Nunnari 2016[5] which is inspirational for our system. The authors of the 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 next.

[2] Danielle Bragg, Oscar Koller, Mary Bellard, Larwan Berke, Patrick Boudreault, Annelies Braffort, Naomi Caselli, Matt Huenerfauth, Hernisa Kacorri, Tessa Verhoef, Christian Vogler, Meredith Ringel Morris: Sign Language Recognition, Generation, and Translation: An Interdisciplinary Perspective, *ASSETS '19*, October 28–30, 2019, Pittsburgh, PA, USA

[3] Lucie Naert, Caroline Larboulette, Sylvie Gibet: A survey on the animation of signing avatars: From sign representation to utterance synthesis, *Computers & Graphics*, **92**, 76-98, 2020, https://doi.org/10.1016/j.cag.2020.09.003

[4] Kacorri, Hernisa: Tr-2015001: A survey and critique of facial expression synthesis in sign language animation; 2015. CUNY Academic Works. https:// academicworks.cuny.edu/gc_cs_tr/403

[5] Heloir, A., Nunnari, F. Toward an intuitive sign language animation authoring system for the deaf. *Univ Access Inf Soc* **15**, 513–523 (2016). https://doi.org/10.1007/s10209-015-0409-0

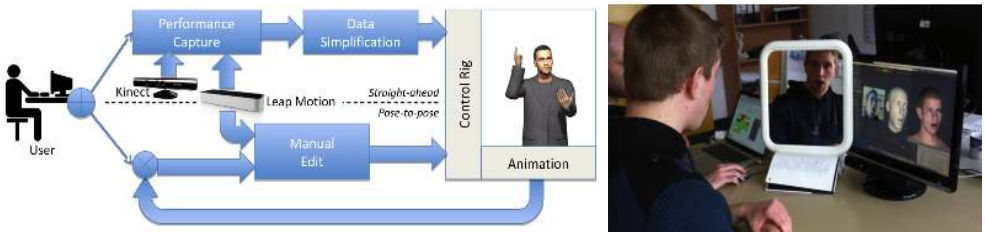

Fig 4: (left) Diagram of the authoring system proposed by Heloir-Nunnari; (right) photo of the system

Before discussing the different stages of our work, we should mention the two basic approaches with respect to animation: *procedural* (such as this example of animation based on SigML), where continuous motion is relatively easy to generate, but usually leads to robotic looking movements, so that realism needs to be enhanced, and *data driven* (such as this example based on Motion Capture, MoCap), which leads to movements that look more realistic, but requires a lot of manual capture of signs/animations. The tool we report on is intended for capturing animations, to be able to use a hybrid, procedural/data driven approach.

3. Preparatory and Related Work: MoCap Strategies

The system by Heloir and Nunnari mentioned above was based on capture using some input devices, Leap-Motion and Kinect-like, which were then relatively novel. As hand capture is very important for SLs, we have been exploring different possibilities.

Marker-based methods (see Figure 5, which normally use visual markers at key positions to be able to capture better movements) are highly intrusive which makes it difficult to perform signs correctly. This is actually a problem, since the main objective of doing Motion Capture is to retrieve natural animations. Additionally, errors usually appear in the capture process. These errors are manifested as noise and loss of the tracks. The noise can be produced by physical problems such as camera inaccuracy, illumination changes, camera movement, etc.; and the loss of tracks occurs when markers are occluded to the cameras.

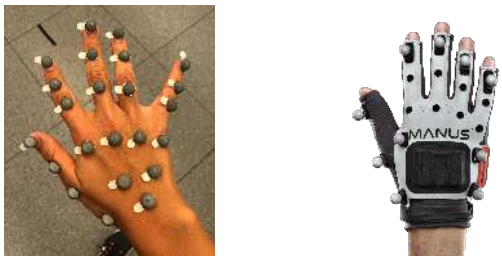

Fig 5: Marker systems

Sensor-based methods (which use sensors, visual or electromagnetic, to capture the movements, see the image at Fig 6) are based on devices which can only register actions within a limited range, many of them are planned to be placed as a point of view of the person. This is not a good option for Sign Language because signs need a large range of action, including relatively large displacements of hands and arms, and including the face. Additionally, the results obtained for some movements within the short range are also not good enough for our case, where our objective is to make the capture very close to the reality of the sign. In Figure 6, we can see a simple case that we tested, where the sensor cannot capture the correct shape of the hand.

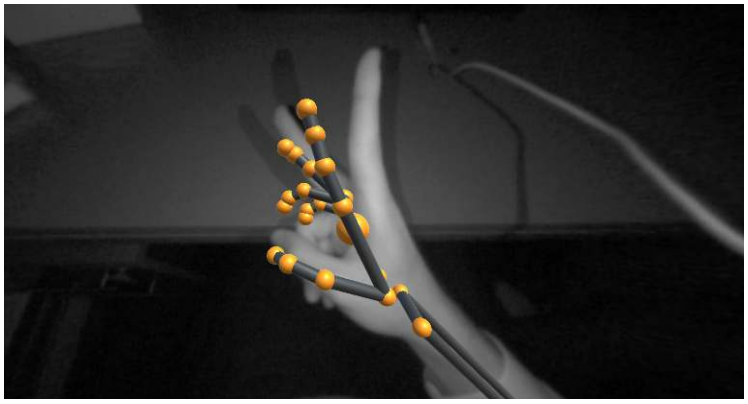

Fig 6: Sensor-based systems example using Ultraleap Stereo IR 170

Machine Learning methods have recently become very powerful as supported by the strong push from deep learning; movements are reconstructed from real models and for that reason the methods deal well with occlusions.

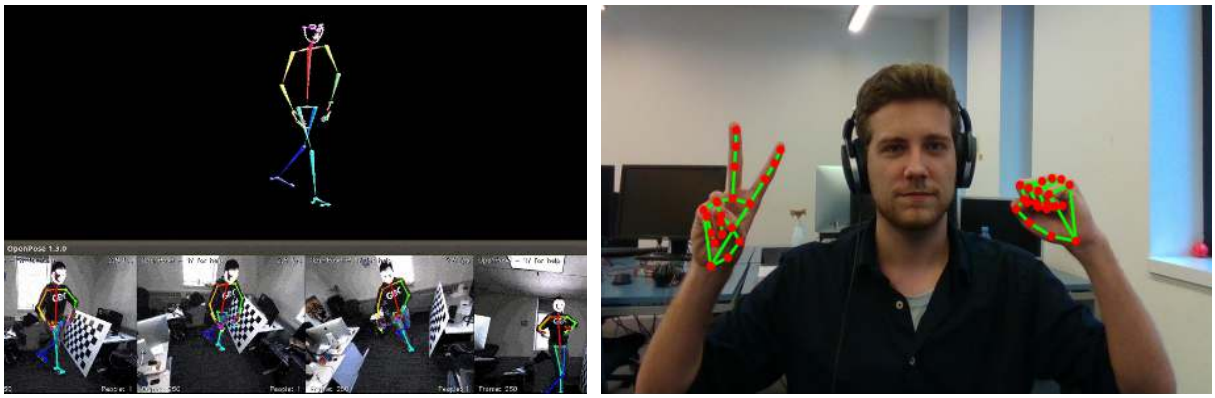

Fig 7: (left) OpenPose; (right) MediaPipe

UPF-GTI has been testing the different methods, and the brief discussion above summarises our main conclusions from our tests and the information available.

Additionally, UPF-GTI organised a session with a signer to collect signs using a high-end Motion Capture (MoCap) system available at UPF. The main goal of this activity was to experiment with an existing (high-cost) system as soon as possible, to identify as many issues as possible to inform the (low-cost) tool we were preparing in parallel, while at the same time see how well a high-cost system performs in order to compare results.

Fig 8: (left) Pictures of the MoCap Lab and systems used; (right) preparing the capture

The capture session could not take place in July 2021 as planned because of a new peak in the COVID-19 pandemic, and instead took place at the end of October. The setup was previously prepared with a non-signer who played mock-up gestures, and the actual capture was performed later with a 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, and it included details on the skeleton to be used (especially the hand, as shown below). The planning was inspired in the process described in Naert et al[6] and the work done in Matthias Schöder et al[7].

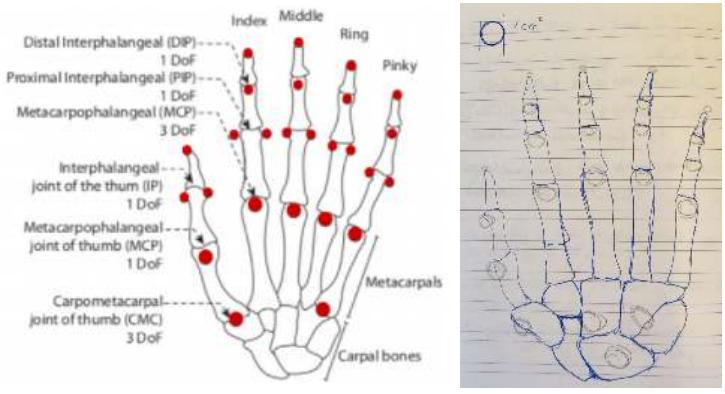

Fig 9: (left) Usual markers setup; (right) Our approach for the markers setup

[6] Naert, L., Larboulette, C. and Gibet, S., 2020, May. 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* (pp. 6008-6017).

[7] Matthias Schröder, Jonathan Maycock, and Mario Botsch. 2015. Reduced marker layouts for optical motion capture of hands. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games (MIG '15)*. Association for Computing Machinery, New York, NY, USA, 7–16.

The session uncovered quite a few issues, from signs especially difficult to capture, through improvements in the process of capture, or the complexity of the associated software, to the noisiness and errors of the capture (46 markers, which should have led to the same number of tracks, generated approximately 250 tracks). Some of these problems can be solved with post processing, but we are still assessing whether it is worthwhile compared to the results from other capture systems. Nevertheless, it was in general useful to the overall objectives of the project, and the work with the MoCap Lab of UPF might be continued.

4. First Functional Prototype

Connected with this previous work, a first functional prototype of a web tool to capture, edit and store SL animations has been developed. The functionalities are supported by different elements of the user interface (UI). A screenshot of the interface for **capturing** is shown below.

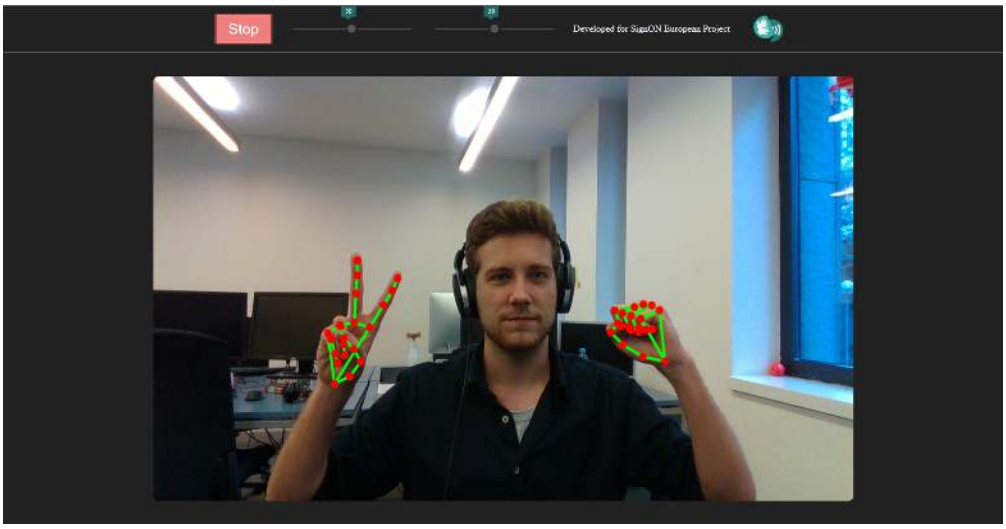

Fig 10: Screenshot of the capture interface

Capturing takes place through a standard webcam, and is based on MediaPipe[8], as hinted at in the screenshot. There is a *start* and *stop* button (and scheduling a short delay for starting the capture after clicking the capture button is possible). After clicking on *stop*, a confirmation/go back dialog appears. If confirmation is clicked, the next functionality, **editing**, is enacted through the interface shown below. However, for the first prototype, the capture has been performed offline using a Blender plugin to

[8] According to its own developers, "MediaPipe offers open source cross-platform, customizable ML solutions for live and streaming media". (https://mediapipe.dev/)

convert a video into an animation file in the widely used format BVH. The connection between the offline and the online modules will be integrated in the next version.

The following figure shows the Blender Interface with its main steps: 1) used plugin options; 2) indication of the video to load and convert to skeleton; 3) exporting the animation in BVH format.

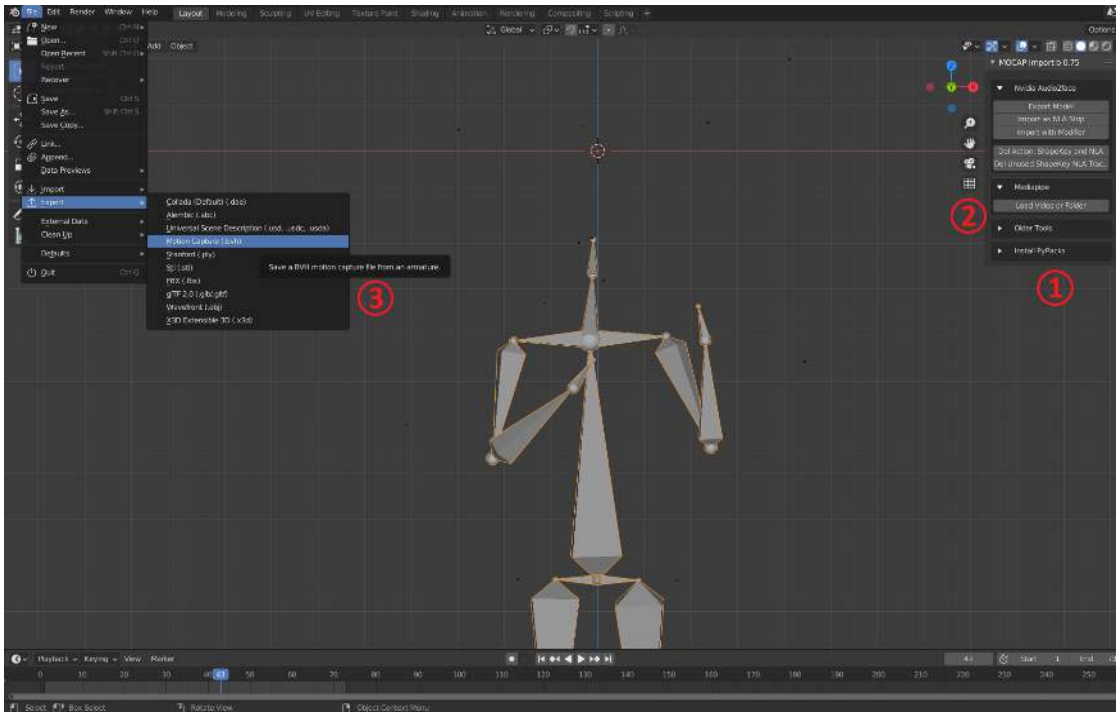

Fig 11: Screenshot of the offline capture through Blender

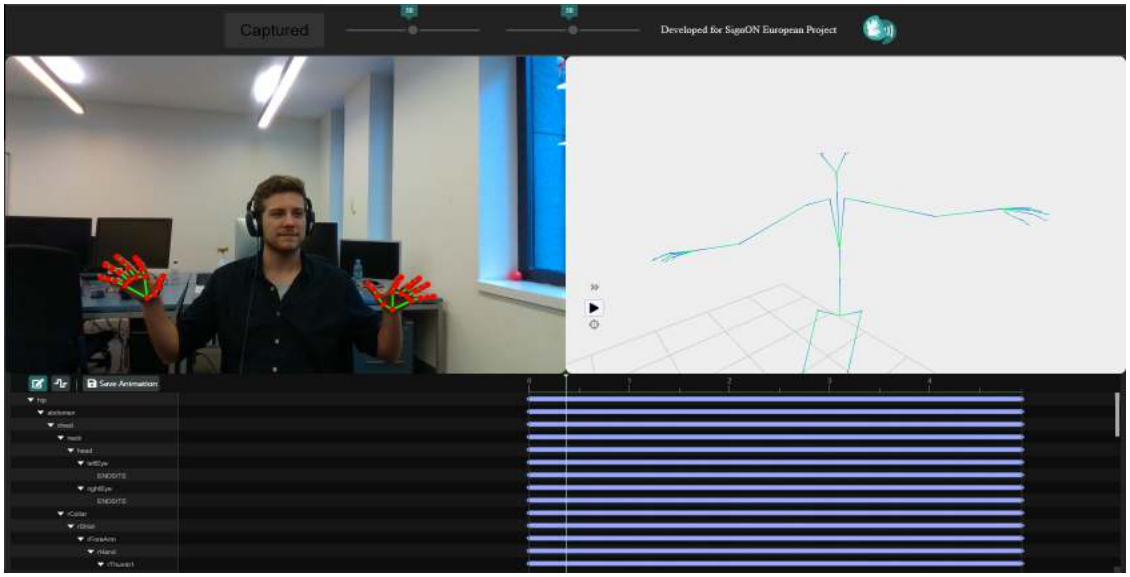

Fig 12: Screenshot of the editing interface

When we move to the editing station (see Figure 12), the window is divided into three different subwindows. The top left one replays the video captured, which appears synchronised with the one that shows the animated skeleton captured (possibly containing errors). The one on the bottom shows a timeline of the animation, with the different joints of the skeleton and a channel for each, so that they can be edited, in order to get the right animated sign. When the user is satisfied with the animation[9], a dialogue requesting confirmation to store the animation, as shown next, appears.

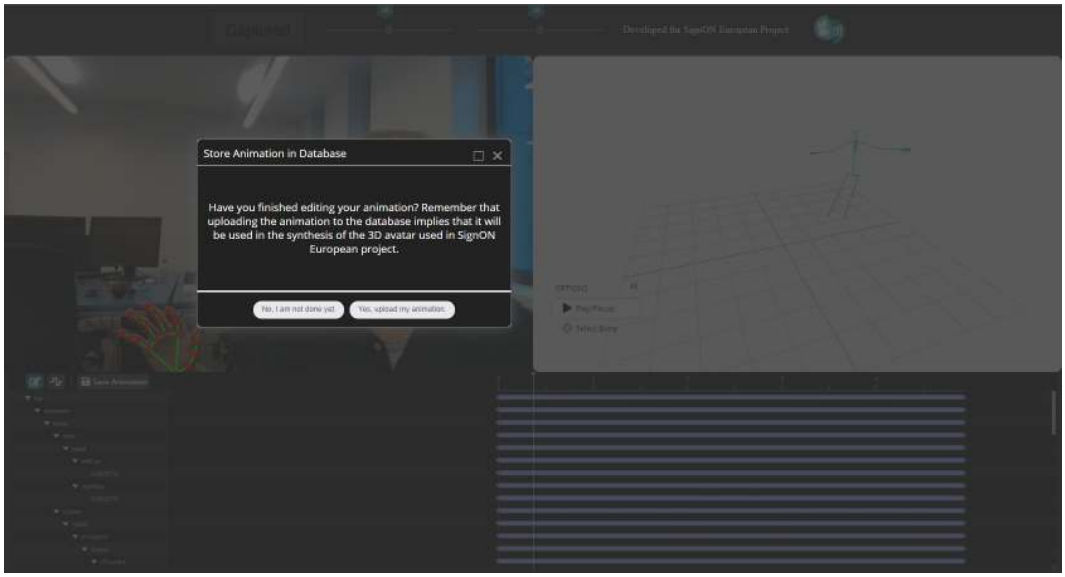

Fig 13: Screenshot of the storing interface

Animations are also stored in BVH format, which should allow their easy exchange and re-use.

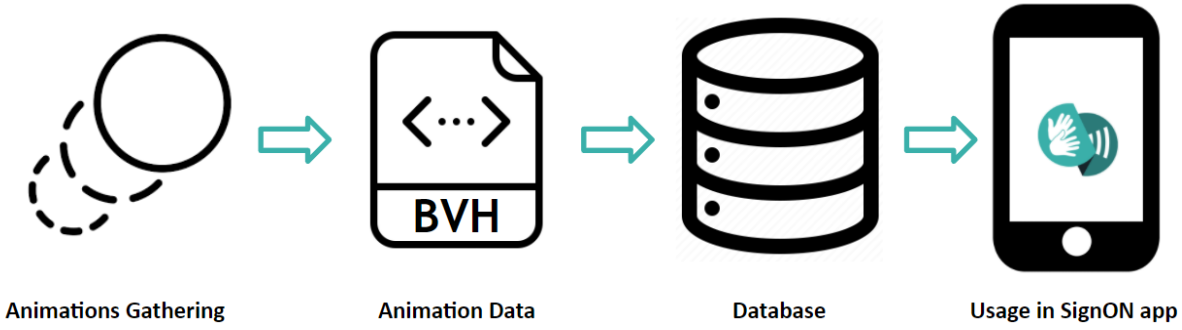

Fig 14: Diagram of the pipeline towards the app intended

Going into more detail about the data, MediaPipe is based on markers as positions in space, which give the locations of elements of the model. In order to turn this into data of a skeleton based animation, one

[9] The system might need moderation to ensure the quality of the animations uploaded in the database. This aspect will be explored in the final stages of the project, when enough animations and experience with capturing them has been collected, to be able to address this problem in a more appropriate way.

needs to take into account that 3D skeletal models consist of a hierarchy of bones, and animations are based on providing the rotations of the joints. Thus, to obtain the BVH data, we need to convert marker locations to a hierarchy of rotations. As indicated above, the BVH format is widely supported and allows for retargeting, i.e. the data can be adapted to different 3D models quite easily.

Fig 15: (left) MediaPipe skeleton; (right) BVH skeleton

Apart from Mediapipe tracking solutions, we used other modules/libraries to build this prototype. In the case of the 3D scene (right window from editing view, Figure 12) we make use of ThreeJS (https://github.com/mrdoob/three.js/), a library and web-oriented API to display animated 3D computer graphics using WebGL. It manages the 3D graphic engine, the load of the BVH data, and its use to animate the avatar. On the other hand, the timeline is taken from the web-based 3D scenes editor WebGLStudio (https://webglstudio.org/) from UPF-GTI. And lastly, the data storage is handled by our own file management library called LiteFileServer (https://github.com/jagenjo/litefilesystem.js).

5. Analysis/Evaluation of First Functional Prototype

The first functional prototype has allowed us to achieve several objectives:

- To provide a running proof of concept of the tool which should allow us to get a variety of natural animations of signs. This can then be used in the SignON app, which can be demonstrated to the partners of the project in order to support the user-centric nature of the project.

- To test the integrability of different components and systems resulting in a tool which can be easily available

The prototype still has some technical limitations. For instance, at the moment of writing this report we are fixing some bugs related to the visualisation of the obtained BVH. The prototype works in some browsers but not all, this will be fixed in future versions.

But a most important activity with the prototype has been to elicit expert evaluation/feedback, especially regarding the key editing activity. The complexity of this activity can be illustrated by the figure below, which is taken from the interface of one of the most often used software in animation editing (Blender).

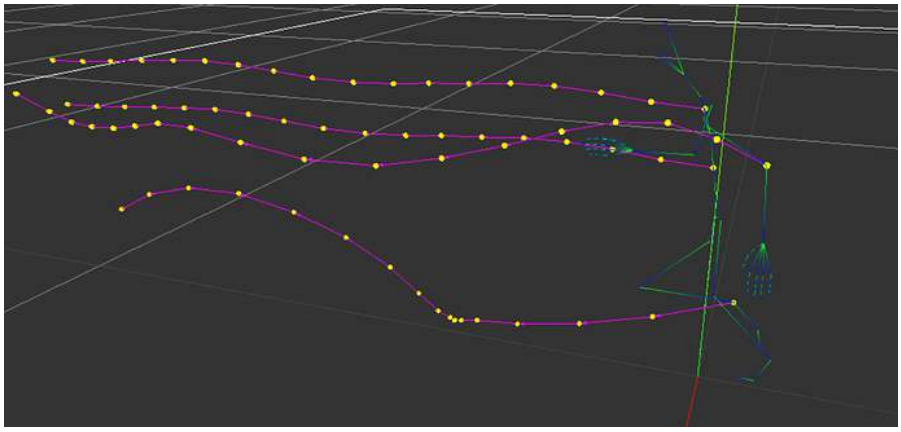

Fig 16: Exemplary editing interface of MoCap systems data (which shows the tracking of the different joints recorded by the system, which need manual editing)

The image shows the representation of the data captured by a high-end MoCap system for the different markers, at time steps. Data is usually noisy, sometimes also with substantial errors, and needs to be edited. On the other hand, it would be prohibitively costly to edit each frame, as there are lots of them, and thus this process should be carried out at key frames, which demands the task of previously identifying them.

The editing interface of the first prototype has similar problems as those illustrated by the image above, and those discussed in the previous paragraph. The expert evaluation has identified this key issue, and thus, we have already started to reformulate the interface, taking as a model previous work[10] done in UPF-GTI, and thus, the code will be re-usable.

[10] (https://tamats.com/projects/character_creator/)

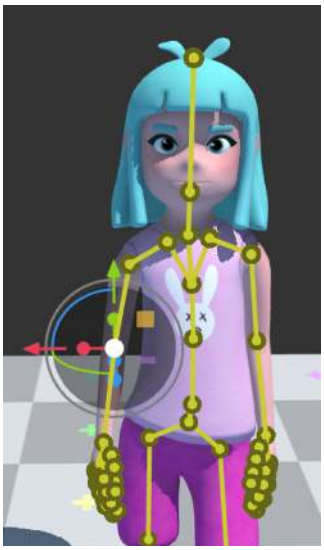

Fig 17: Model for reformulating the editing interface

This interface has the substantial advantage of allowing the user to work directly with the joints that need editing, and is already based in a rotation based animation approach.

6. Conclusions and Future Work

This document describes the first prototype of the tool for capturing, editing and storing SL animations conceived and implemented by UPF-GTI. We presented the context of this deliverable and the work within the SignON project and the SL synthesis tasks, pointed out some significant aspects of the current state of the art in animating virtual signers, discussed background work and presented key choices and aspects of the tool, as well as the initial expert evaluation, and the reformulation of the interface.

We intend to produce a second prototype that fixes the interface and technical issues in the next three months, and then have a second extensive evaluation with user partners.