This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n: 761999

EasyTV: Easing the access of Europeans with disabilities to converging media and content.

D3.1 Sign language capturing technology preliminary version

EasyTV Project

H2020. ICT-19-2017 Media and content convergence. – IA Innovation action.

Grant Agreement n°: 761999

Start date of project: 1 Oct. 2017

Duration: 30 months

Document. ref.:

Disclaimer

The EasyTV Consortium consists of the following partners:

	Partner Name	Short name	Country
1	Universidad Politécnica de Madrid	UPM	ES
2	Engineering Ingegneria Informatica S.P.A.	ENG	IT
3	Centre for Research and Technology Hellas/Information Technologies Institute	CERTH	GR
4	Mediavoice SRL	MV	IT
5	Universitat Autònoma Barcelona	UAB	ES
6	Corporació Catalana de Mitjans Audiovisuals SA	CCMA	ES
7	ARX.NET SA	ARX	GR
8	Fundación Confederación Nacional Sordos España para la supresión de barreras de comunicación	FCNSE	ES

PROGRAMME NAME:	H2020. ICT-19-2017 Media and content convergence. convergence. – IA Innovation action
PROJECT NUMBER:	761999
PROJECT TITLE:	EASYTV
RESPONSIBLE UNIT:	CERTH
INVOLVED UNITS:	CERTH, UPM
DOCUMENT NUMBER:	D3.1
DOCUMENT TITLE:	Sign language capturing technology preliminary version
WORK-PACKAGE:	WP3
DELIVERABLE TYPE:	R
CONTRACTUAL DATE OF DELIVERY:	20-09-2018
LAST UPDATE:	20/09/2018
DISTRIBUTION LEVEL:	PU

Distribution level:

PU = *Public,*

RE = *Restricted to a group of the specified Consortium,*

PP = *Restricted to other program participants (including Commission Services),*

CO = *Confidential, only for members of the LASIE Consortium (including the Commission Services)*

Document History

VERSION	DATE	STATUS	AUTHORS, REVIEWER	DESCRIPTION
v.0.1	21/08/2018	Draft	Kiriakos Stefanidis (CERTH)	Table of Contents definition and document structure
v.0.2	13/09/2018	Draft	María Poveda Villalón (UPM)	Added section 8.4
v.0.3	20/09/2018	Final	Kiriakos Stefanidis, Anargyros Chatzitofis, Kosmas Dimitropoulos and Petros Daras (CERTH)	Final version ready for review
v.0.4	25/09/2018	Final	Kiriakos Stefanidis, Kosmas Dimitropoulos (CERTH)	Final version after reviews

Definitions, Acronyms and Abbreviations

ACRONYMS / ABBREVIATIONS	DESCRIPTION
API	Application Programming Interface
GUI	Graphical User Interface
CNN	Convolutional Neural Network
DOF	Degrees of freedom
DoA	Description of Action
FPS	Frames Per Second

Table of Contents

List of Figures

List of Tables

Executive Summary

This document is the preliminary version of the deliverable D3.1 concerning the development of the EasyTV capturing technology. This technology will be used for capturing the motion of the signers, i.e., persons speaking Sign Language. The main points outlined in this deliverable are the following:

- A multi-phase architecture for the development of the EasyTV capturing technology has been designed in order to meet the requirements of the DoA.
- Research on the current state-of-the-art of hand tracking sensor technologies has been conducted and the optimal setup for the requirements of capturing Sign Language has been decided.
- A first implementation of motion data generation is provided. Deep network algorithms detect 2D keypoints for the hands, face and body of the signer and then 3D data is formed by merging the values from the corresponding depth frames.
- Different types of exported files and formats are discussed.

Finally, a first implementation of the EasyTV capturing module, i.e., motion capture application, is presented. In this initial version of the application, the different phases of the capturing process have been integrated to form the complete structure of the application, but contain parts that have not yet been implemented. These parts will be implemented in the following development steps and presented in the final version of the deliverable.

1. INTRODUCTION

Motion capture systems are technologies that are used for capturing the motion of a person. The result of this process is data that encodes motion information and is used to animate digital character models in 2D or 3D computer animation. Such technologies have been successfully used in cinema, video games, sports and military training. Recently, there have also been some efforts to use motion capture for the playback of signing avatars.

The EasyTV capturing technology aims to provide accurate motion capture for Sign Language translation tasks. These tasks will be defined by the moderator of the EasyTV crowdsourcing platform (T5.2) and distributed to Sign Language experts in order to fulfill them. Since speaking Sign Language involves not only hand gestures but also facial expressions, the technology has to be capable of capturing all necessary information for any sign and thus include an accurate hand tracking technology, as well as, detectors for the face and body of the signer. The results exported by this module include text, video, and most importantly, motion files. This data will be uploaded to the EasyTV crowdsourcing platform.

The present document is organized as follows:

- **Chapter 2** presents the architecture of the capturing technology conforming to the EasyTV DoA.
- **Chapter 3** gives an overview of the existing hand tracking technologies and proposes a capturing setup that fits the requirements and constraints of Sign Language.
- **Chapter 4** describes the acquisition phase where the signer speaks Sign Language on the capturing setup and images are recorded.
- **Chapter 5** presents an initial design for the annotation tool which will be used for annotating the captured videos with text.
- **Chapter 6** analyzes the detection phase where 2D keypoints are detected and 3D data is generated.
- **Chapter 7** gives an overview of the refinement process where errors in the motion data are eliminated and values are restored.
- **Chapter 8** discusses the data files that are exported by the capturing module, as well as, their utilization by the crowdsourcing platform and other EasyTV services.
- **Chapter 9** outlines the conclusions of this work.

2. ARCHITECTURE

This chapter outlines the aspects concerning the architecture of the EasyTV capturing technology. These include the interconnection with other EasyTV services (i.e., external architecture), as well as, the internal components that make up the capturing technology (i.e., internal architecture).

2.1. Input & output

The input and output of the capturing technology is defined in document D1.3 “First release of the EasyTV system architecture” which describes the overall EasyTV architecture and the interconnections between its modules and services. According to this document, the input of the EasyTV capturing technology is given by an expert signer, that is, a person who has excellent knowledge of Sign Language and is capable of correctly signing in front of the capturing sensor. The input acquired by the capturing software concerns different types of image data (i.e., RGB-D) that capture the requested sign or signs, but also, text annotations that describe the meaning of each sign. As it will be described latter in this document, these two actions are performed in a sequential manner with the acquisition of the image data performed first and the user annotation following next. The visual content is processed by detection algorithms in order to extract the necessary information to compose the motion information of the signing.

The output of the capturing module is a collection of files that are uploaded to the crowdsourcing platform. The uploaded data will be stored properly into repositories and be available for use by other EasyTV services and components.

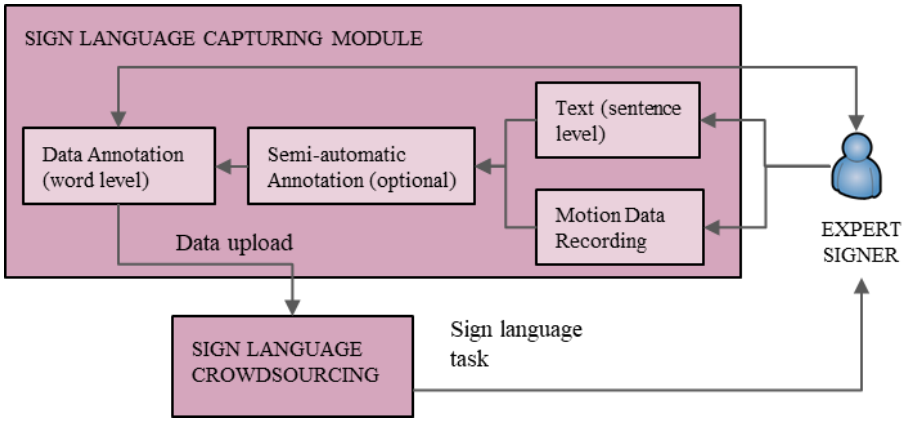

Figure 1: Schematic representation for the input and output of the capturing module.

A schematic representation for the input-output of the capturing module is given in Figure 1. In this figure, the crowdsourcing platform issues a task for the signer, the signer makes the requested signs and the capturing module records the images and extracts the motion data. As we can see, there are different types of annotation procedures: the expert signer initially annotates sentences but can also break the sentences into words in a later stage. Moreover, a semi-automatic annotation mechanism can be used for making the process easier for the signer by automatically recommending the right translation for a given sign. At the end of the procedure, the data is uploaded to the Sign Language crowdsourcing platform and stored into repositories. The crowsourcing platform can generate additional tasks to the same signer and the whole process is then repeated.

2.2. The phases of the capturing process

The EasyTV capturing technology consists of a number of phases that meet the requirements of the EasyTV project for Sign Language motion capture. These phases form a pipeline architecture

where the output of each phase is given as input to the next one. The architecture includes phases for the acquisition of images containing the signs, the annotation of each sign, the detection of keypoints for the hands, body and face, the generation of 3D data, the correction of motion data, and the formating of motion files. A graphical representation of this interconnection between the phases of the capturing process is given in Figure 2.

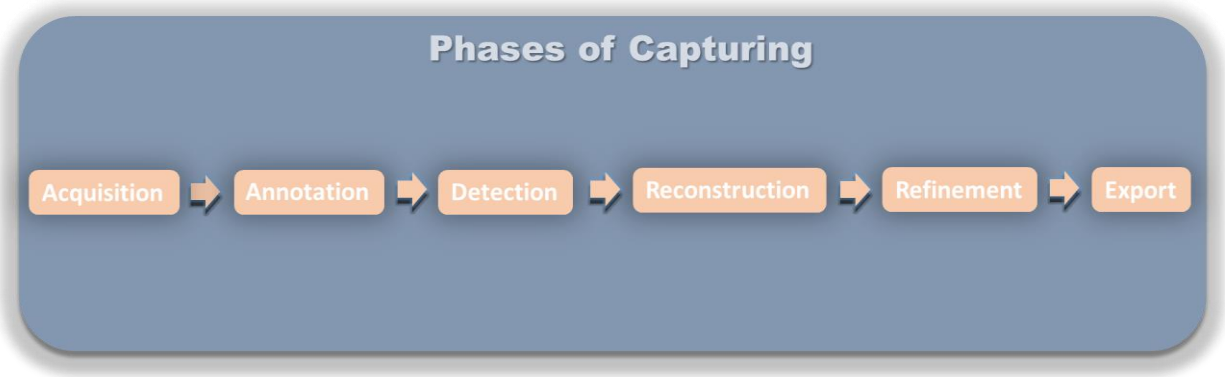

Figure 2: Pipeline diagram demonstrating the six phases implemented in the EasyTV capturing technology.

More specifically, a brief description for each phase is given below:

- **Acquisition:** The signer makes the requested signs and the capturing software acquires images from the sensor. There are two types of images captured by the sensor: those containing RGB information and others containing depth information. Both types though are necessary for the generation of 3D motion data. Also, color and depth videos can be generated in order to display the captured content to the user (e.g., admin or signer) in the annotation phase.
- **Annotation:** The admin or the signer reviews the captured content and annotates video. The annotation concerns the mapping of visual content to text (i.e., subtitles). This process can involve the annotation of the whole video with a single phrase or the trimming of the video, that is, the video is divided into segments with each segment representing a sign and annotated with text.
- **Detection:** Algorithms detect keypoints on frames. Keypoints are points of interest on the captured frames that identify the important visual content on the image. In our case, the important parts of the images are the hands, face and body of the signer. The keypoints detected can be either 2D or 3D keypoints. Currently, we use a detection procedure for 2D keypoints. The number of the keypoints and their position depends on the training of the detection algorithms. In the current state-of-the-art, such detection algorithms are usually deep neural networks that are trained on annotated images.
- **Reconstruction:** This is the method for generating 3D keypoints. Since our detected keypoints are two-dimensional, we also need to infer the depth information in order to compose 3D keypoints. The 3D keypoints are necessary for generating motion data that can be imported into the EasyTV realistic Sign Language avatar (T2.4).
- **Refinement:** Concerns a method for automatic error correction in the detection process. This phase is required due to noise in the detected data or mis-detections.
- **Export:** Data that is exported in proper file formats for compatibility with other EasyTV services. The most important files exported are the motion files supported by 3D animation software and avatar technologies.

A more detailed analysis of each phase is given in the following chapters of this document.

3. CAPTURING SETUP

This chapter provides a brief overview of the existing sensor technologies used for hand tracking, as well as, the overall setup that is considered to be suited for the purposes of EasyTV and the capturing process of Sign Language.

3.1. Sensor technologies for hand tracking

Hand tracking techniques can be divided into three major categories:

- Tracking with interface where inertia and optical motion capture (mocap) systems are used).
- Tracking without interface where just cameras are used.
- Other tracking techniques.

Figure 3 presents some examples of the most popular sensors from the first two categories.

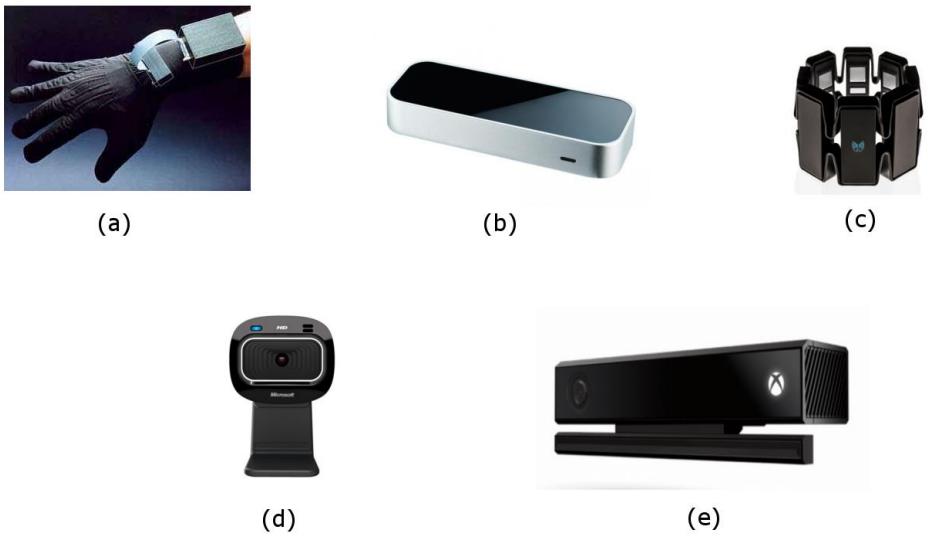

Figure 3: Hand tracking sensors: (a) Gloves, (b) Leap Motion, (c) Myo armband, (d) RGB camera, (e) Microsoft Kinect.

3.1.1 Hand tracking with interface

- **Inertial motion capture gloves**

Inertial motion capture systems are able to capture finger motions by sensing the rotation of each finger segment in 3D space. Applying these rotations to kinematic chain, the whole human hand can be tracked in real time without occlusion and through wireless communication.

Hand inertial motion capture systems, like for example Synertial mocap gloves [10], are using tiny IMU based sensors, located on each finger segment. For most precise capture, at least 16 sensors have to be used. There are also mocap gloves models with less sensors (13 / 7 sensors) for which the rest of the finger segments is interpolated (proximal segments) or extrapolated (distal segments). The sensors are typically inserted into textile gloves which makes the use of the sensors more comfortable.

Because the inertial sensors are capturing movements in all 3 directions, flexion, extensions and abduction can be captured for all fingers and thumb. The captured movements are then interpreted

by the software that accompanies the gloves, and gestures can then be categorized into useful information, such as to recognize Sign Language or other symbolic functions.

Expensive high-end wired gloves can also provide haptic feedback which is a simulation of the sense of touch. This allows a wired glove to be used also as an output device. Traditionally, wired gloves have only been available at a high cost. Besides sign recognition, wired gloves are often used in virtual reality environments and in robotic applications to mimic human hand movement by robots. Some examples include CyberGlove Systems [7], Manus VR [8], VR gluv [9], Synertial gloves [10].

Inertial sensors have two main disadvantages connected with finger tracking:

- Problem to capture absolute position of the hand in space.
- Problem with magnetic interference - metal materials use to interfere with sensors.

The second problem may be noticeable mainly because hands are often in contact with different things, often made of metal. The current generations of motion capture gloves are able to withstand unbelievable magnetic interference. The magnetic immunity depends on multiple factors: manufacturer, price range, and number of sensors used in mocap glove.

- **Optical motion capture systems and marker functionality**

Some of the optical systems, like Vicon [11] or ART [12], provide hand motion capture with the use of markers. In each hand there is a marker per each “operative” finger. Three high-resolution cameras are responsible for capturing each marker and measure its positions. The coordinates in 3D of the labels of these markers are produced in real time with other applications, providing that the markers are visible to the cameras.

Markers operate through interaction points, which are usually already set. Because of that, it is not necessary to follow each marker all the time; the multipointers can be treated in the same way when there is only one operating pointer. To detect such pointers through an interaction, ultrasound infrared sensors are enabled. In the case of bad illumination, motion blur, malformation of the marker or occlusion, the system allows following the object even though some markers may not be visible. Because the spatial relationships of all the markers are known, the positions of the markers that are not visible can be computed by using the markers that are known. There are several methods for marker detection like border marker and estimated marker methods.

3.1.2 Hand tracking without interface

At hand tracking without interface methods, image analysis techniques are applied to detect the hand part from images captured with RGB or RGB-D cameras. To detect hand gesture one must first segment the hand image from the background. The traditional method of using single RGB camera relies on skin colour for segmentation and is unreliable due to a wide variation in user skin colour. Depth cameras make gesture detection much more reliable because depth can be used for image segmentation. Shape analysis is then applied to the segmented foreground (hand) to detect and track fingertips and palm to recognize gesture.

In general, most commercial systems of this category perform the following steps:

- Point and pinch gesture recognition: taking into account the points of reference that are visible (fingertips).
- Pose estimation: a procedure which consists of identifying the position of the hands through the use of algorithms that compute the distances between positions.

3.1.3 Examples of modern mocap systems

- **HTC Vive**

Vive [13] is a virtual reality headset designed with SteamVR in mind which comes with two wireless

controllers that act as the virtual “arms” to track hand and arm movements in space in real time.

- **iMotion**

Created by a California-based startup Intellect Motion, iMotion [14] is a motion controller that creates haptic feedback by hand-and-arm interactions through a “virtual touchscreen”. The iMotion controller provides pinpoint accuracy in three-dimensional space by utilizing a device’s camera, such as those found in smartphones, tablets, gaming consoles, and PCs. By installing an extra software on a compatible device, the iMotion turns the user’s hands into a controller for on-screen objects in games, VR experiences, and even the simple navigation of a device’s homescreen.

- **Leap Motion**

Leap Motion [15] is a handsfree solution specifically designed for virtual reality. The Leap Motion for Virtual Reality Mount is a camera that is mounted onto VR headsets in order to give accurate depth tracking and motion sensing and enhance VR experiences by turning the user’s hands as the sole controller of anything on screen. Leap Motion provides a wide field of view of 135 degrees, almost-zero latency, pinpoint accuracy, and high robustness. It is still in beta stage, meaning its development is in progress. Some compatible VR devices for the mount are the Oculus Rift Development Kits 1 and 2.

- **Microsoft Handpose**

Developed by Microsoft, Handpose [16] is a cutting-edge hand tracking technology that uses the Kinect motion-sensing camera to detect articulated hand movements in space in real time. What sets apart Handpose from other haptic technologies in development is its high degree of robustness in tracking errors, and its fast data processing that compensates for these possible errors. The Kinect camera scan tracks hand movements within the camera’s range of view or focus, whether the hand is far or near the camera.

- **MindLeap**

During the recent Game Developers Conference in San Francisco, Swiss neurotechnology company MindMaze [17] unveiled its one-of-a-kind technology that benefits virtual reality. It’s called MindLeap, and is made by Swiss neurotechnology company MindMaze. It is a technology that reads neural signals to perform certain tasks. But besides decoding neural signals, it can also track hand movements using advanced tracking sensors. MindLeap was primarily developed for medical applications, but is also used in the VR section. It is still in continuous development as the MindLeap technology is being offered as an SDK for third-party developers to enhance certain experiences such as in gaming and VR.

- **Myo**

Developed by Thalmic Labs [18], Myo is an armband worn by the user to be used as an interaction device for many practical applications. By using electrical signals in the user’s muscles, Myo can interpret these signals as perceived input and uses it to trigger certain on-screen functions in a variety of interfaces. It is versatile such that it can be used to control computers, smartphones, and tablets. Myo is purely wireless, powered by Bluetooth to connect to third-party devices. As of now, it is currently on sale but the technology is currently being developed to work with other tech devices in the near future, including gaming consoles and VR gear.

- **Nimble Sense**

Nimble Sense [19] is a hand tracking technology developed by startup Nimble VR. It is actually a depth camera that uses depth information to track hand movement with a high level of accuracy and precision. It is suited as an accessory to the Oculus Rift VR headset, as it perfectly sits on top of the headset. With a low latency, it accurately tracks hand input using both a photonic mixer device and an eye-safe laser that effectively senses the location of the user’s hands in space in real time. An API makes this gathered data interpreted into input that translates to manipulations and interactions in the virtual reality interface.

3.2. Proposed setup

While the choice for the best hand tracking sensor is of central importance to reach an optimal setup for the capturing process of EasyTV, the hand tracking problem is considered to be an open research topic entailing several difficulties for its solution. In fact, in July 24, 2017, Oculus Chief Scientist Michael Abrash posted an article on occulus blog website entitled "*VR's Grand Challenge: Michael Abrash on the Future of Human Interaction*" in which he mentioned the challenges posed by the hand tracking problem [20]:

> *"Unfortunately, hands have about 25 degrees of freedom and lots of self-occlusion. Right now, retroreflector-covered gloves and lots of cameras are needed to get to this level of tracking quality"*

The status of the problem still remains the same up till now. Despite the apparent trade off between cost and quality, there are other practical issues one has to consider, such as, the interconnection between the devices and the computational requirements for processing the acquired data. Such matters prevent us from combining a large number of hand tracking sensors into the capturing setup of EasyTV.

The nature of Sign Language imposes additional constraints on the selection of the right sensor technology for the capturing process. More specifically, the signer must be able to make any movement he/she wants, as naturally as possible, without any disturbance from the capturing equipment. For example, wearing hand tracking gloves might disturb the signer and prohibit him/her from making certain types of gestures, such as, hand movements around the head, turning of the wrists, or splitting hands. This also holds true for the rest of the obtrusive hand tracking sensors like arm bands or even small finger trackers. Therefore, we consider that obtrusive tracking technologies are not suitable for capturing Sign Language motion.

Since we omitted the use of gloves and other obtrusive sensors, we turned our attention to markerless tracking technologies. As discussed in the previous section (3.1), markerless sensors have the advantage of being unobtrusive and thus allow any hand movements by the signer. For that reason, we continued our experiments by testing simple RGB cameras, as well as, more advanced RGB-D sensors like Microsoft Kinect v2.0 and Intel RealSense, which additionally capture depth information. We also tested setups consisting of multiple RGB-D sensors. An example of a multi-view capturing setup is shown in Figure 4.

Figure 4: Testing a multi-view setup in visual computing lab at CERTH.

Concerning the generation of 3D data, we have focused our attention primarily on two 3D reconstruction methods: multi-view 3D reconstruction and 3D reconstruction using depth alignment. Comparing these two methods we can find many differences in their requirements. More specifically, a multi-view setup (see Figure 4) requires multiple RGB sensors for capturing images from different angles, and for this reason it raises the cost of the equipment. Also, each sensor needs to be connected to a separate personal computer and this further raises the cost. In order to extract the depth information, this reconstruction method applies the technique of triangulation, in which a point is estimated in 3D space given its projections onto two, or more images [21]. This process is very computational demanding and usually done locally on the PCs performing the capturing. For that reason, the specifications for the PCs need to be high-end. Also, the quality of the 3D reconstruction is proportional to the number of sensors.

Another issue is synchronization. This means that the multiple views of each frame must be synchronized in order to achieve good results in triangulation. The problem here lies in that real synchronization can only be succeded through hardware synchronization, that is, via an external pulse generator signaling the start of the acquisition process for the imaging devices. Even the best software synchronization solutions do not produce perfectly accurate results. Last requirement for multi-view 3D reconstruction is calibration. All cameras must be calibrated in order to get the intrinsic and extrinsic parameters of each imaging device. This raises many practical issues in a capturing setup. If, for any reason, one camera is moved out of its initial position, the calibration process must be repeated. This is an important issue because the calibration process is non-trivial and must be done by a qualified engineer and not by simple users of the software.

On the contrary, a 3D reconstruction procedure based on aligned depth images is much more practical and has a lot less requirements. More specifically, it requires only one imaging sensor and one PC. This sensor though, has to be an RGB-D device in order to also provide depth information. Moreover, the PC does not need to be a high-end machine because there are no processing-demanding tasks for estimating the depth values. Also, there is no need for synchronization since we only acquire images from one viewpoint. Finally, we are alleviated from any calibration procedure. A summary of the comparison between the two reconstruction methods is given in Table 1.

Table 1: Requirements for 3D reconstruction methods.

Requirements	Multi-view 3D Reconstruction	3D Reconstruction using depth alignment
# sensors	>1	1
# PCs	1 per sensor.	1
Processing power	High-end PC	Minimum requirements
Synchronization	Yes	No
Calibration	Yes	No

To summarize, our research and experimental testing lead us to head towards a capturing setup that can be both accurate in terms of generating high quality 3D data, but also practical in the sense that it makes the process of capturing Sign Language easier. The proposed setup requires just a single RGB-D sensor and one PC or laptop running the EasyTV capturing application. Although the requirements for depth alignment and extraction are negligible as shown in Table 1, the processing requirements of the keypoint detection phase (see chapter 6) require the PC being a high-end machine. A final note about the setup is that lighting conditions in the room must be adequate.

4. SIGN ACQUISITION

This chapter discusses how the signer speaks Sign Language in front of the sensor, how images are acquired by the capturing module and video is generated.

4.1. Signer and admin

First of all, the admin prepares the setting for the recordings. A green panel can be optionally placed on the wall to facilitate the foreground-background separation during post-processing, the Kinect sensor is placed around 2 meters away from the spot of the signer, marked with a white line on the floor and the lighting is adjusted accordingly to achieve optimum conditions of capturing.

Before the start of each capturing session, the admin describes explicitly the recording procedure to the signer. There is a predefined sequence of expressions that the signer has to do, one after the other, in front of the Kinect sensor. The signer is instructed to stand at neutral position beside the white line marked on the floor (Figure 5). Once the admin gives the signal that the recording is starting, the signer should make the first expression of the sequence and then return to neutral position for 1 second. After that, the signer makes the next expression and returns to neutral position again. These steps are repeated until all expressions are collected and the recording session is completed.

Figure 5: Setting up the position of the signer.

4.2. Acquired frames and video generation

Color images are acquired via a RGB sensor. As we described in the previous sections, we chose RGB-D as our preferred sensor technology for capturing signs. One of the RGB-D sensors that we used for testing is Kinect. More specifically, the native resolution of its color frames is fixed at Full HD quality, i.e., 1920×1080 pixels captured at a frame rate of 30 fps. On the other hand, the resolution of the depth images is 512×424 also taken at 30 fps. While its SDK provides detectors that capture skeleton points for the body and the face, it doesn't include a keypoint detector for the hands. Also, the keypoints extracted may not be the preferable for Sign Language motion capture. For that reason we included thrid-party detectors that meet the requirements of the EasyTV project and provide accurate hand tracking. More details for the detection phase are given in chapter 6.

The alignment of the color and depth frames has to be taken into consideration. This means that color and depth frames can have the same resolution. In fact, there is the choice of mapping depth frames to color space or mapping color frames to depth space. As we explain later in section 6.3, the purpose for aligning color and depth frames is to extract for every RGB pixel the corresponding depth value, thus generating 3D data by using 2D keypoint detectors on RGB images. Figure 6 shows an example of RGB and depth frame alignment.

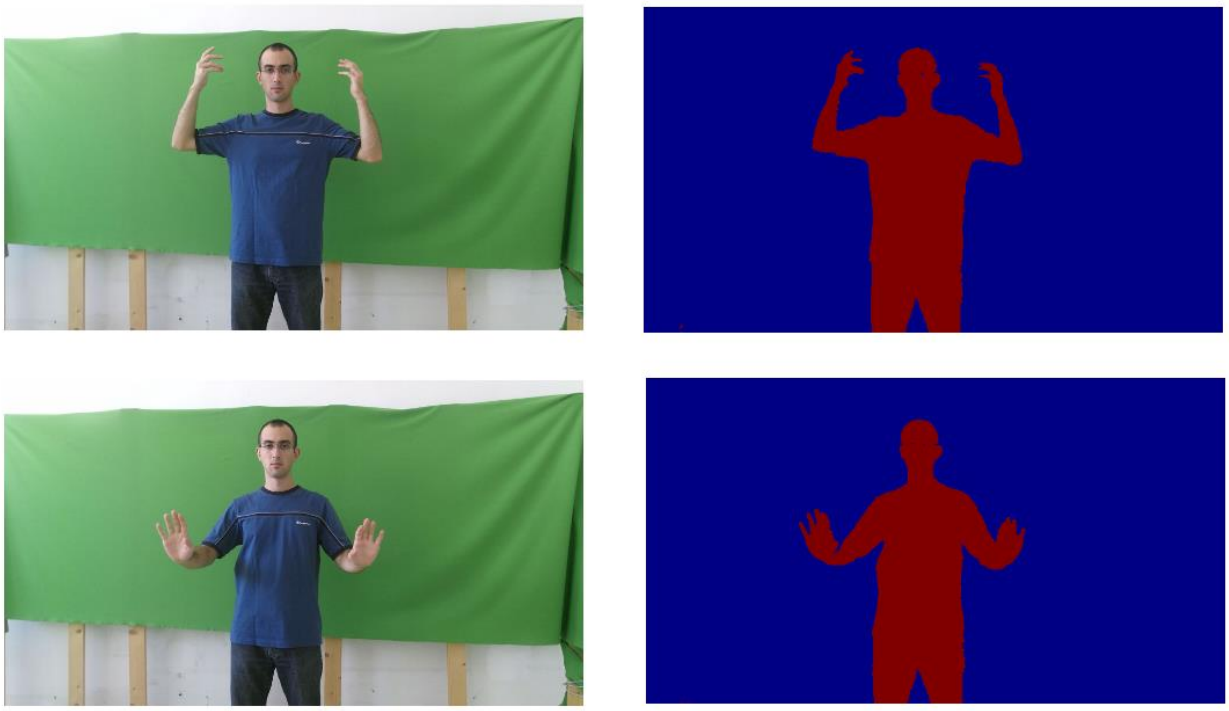

Figure 6: Capturing color and depth frames for Sign Language using the Microsoft Kinect v2.0 sensor. Note that the depth frames are totally aligned with the corresponding color frames.

When the acquisition process terminates and all RGB and depth frames have been collected, they are merged into videos. The purpose for generating videos of the recorded content is two-fold:

- The admin of the capturing process can review the recordings in the annotation phase.
- The moderator of the crowdsourcing platform can review the recordings in the validation phase.

The videos can be exported in a popular file format, such as, the mp4 or avi format. For the generation of the videos a practical solution is the *ffmpeg* software [22]. *ffmpeg* is free software meaning that it can be integrated with the EasyTV capturing technology. It is a command-line application for processing video or audio files, and is widely used for format transcoding, basic editing (trimming and concatenation), and video scaling. Its trimming capabilities make it an even more favorable solution because a separate tool will not be needed for the phase of video trimming (chapter 7).

5. USER ANNOTATION

When RGB and depth images are captured, the signer must annotate them in order to map the signs to their corresponding meaning. In the rest of this chapter, we consider that we have captured a video containing more than one sign, i.e., a video which corresponds to a phrase rather than a single word.

5.1. Isolating the signs

In the first step, the user must properly separate the different signs that are contained in the video. For that reason, the annotation tool must include:

a. A display screen for replaying the RGB video captured in the acquisition phase.
b. A slide bar and/or input fields for time, through which the user can specify the start time and the end time of different portions in the video.

Through this simple GUI, the user reviews the video and marks the time segments that enclose the individual signs. An example of such user interface for video trimming is shown in Figure 7.

5.2. Annotation with subtitles

Alongside the isolation of the signs, the user has to also specify the text corresponding to the meaning of each sign. Therefore, the graphical interface of the annotation tool must include a textbox for receiving the user's input, which must be correlated with each marked time region.

5.3. Synchronized video trimming

The last step of the annotation phase concerns the trimming of the video. The regions marked by the user are given to a video trimming software in order to cut the video into parts. Each part corresponds to an individual sign that the signer made in the acquisition phase. We have to note here that we capture both RGB and depth information so we must trim both streams and not ruin the correspondance between them. Therefore, the trimming process has to be synchronized among the two streams and result into a RGB and a depth video segment for each sign. One of the most practical solutions for realizing the trimming process is the *ffmpeg* software. We can use this software as a separate executable that accepts command line arguments. For this reason, this application can be more easily integrated to a graphical user interface for sign isolation and annotation.

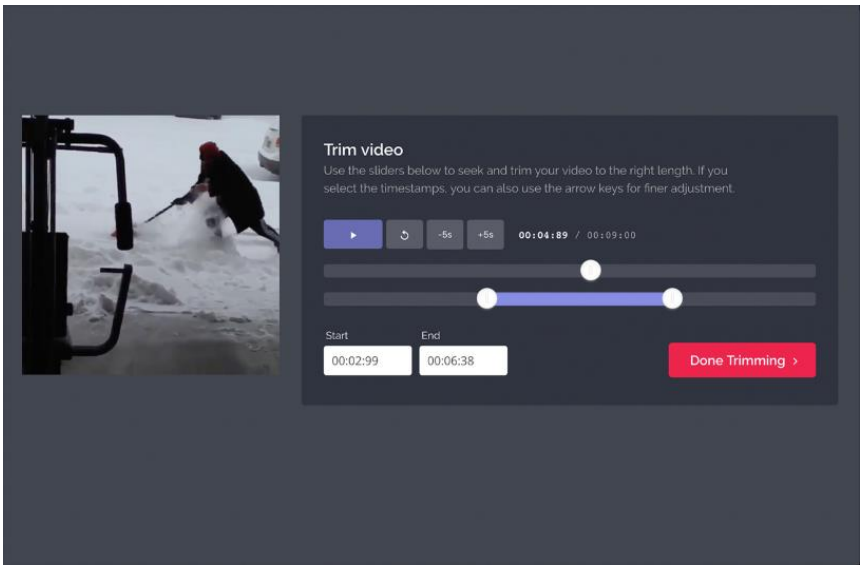

Figure 7: Example of a video trimming software.

5.4. Semi-automatic annotation tool

The purpose of the semi-automatic annotation tool is to enable the automatic detection and classification of signs in a signed sentence, thus assisting an annotator during his/her task. To this end, we propose a Sign Language Recognition (SLR) methodology that can accurately and robustly classify signs [6]. A block diagram of the proposed SLR methodology is presented in Figure 8.

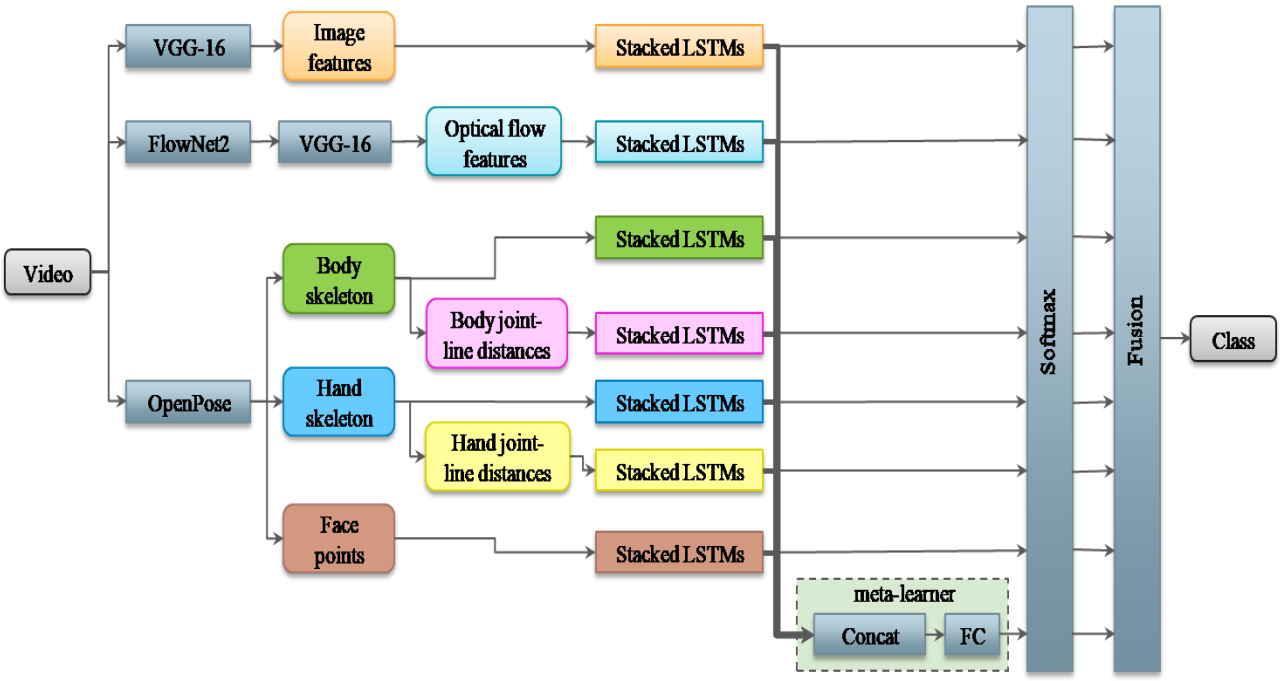

Figure 8: Proposed SLR methodology [6]. Each data stream is shown with a different colour.

The proposed methodology relies on the extraction of video (i.e., image and optical flow) and skeletal (i.e., body, hand and face) features from video sequences. Established deep learning methods are employed in order to extract discriminative features from video sequences, compute motion between consecutive frames (optical flow) and extract skeletal data from images. All these features form 7 streams that are individually processed by recurrent neural networks, called Long Short-Term Memory (LSTMs) in order to derive temporal dependencies among the features. An extra network, called meta-learner is also employed to merge the information from all streams and compute additional features. Finally, all the derived information is fused in order to compute the optimal class for each tested video sequence. For further information about this SLR methodology, we urge the read to refer to the deliverable D3.3 "*Remote control with gesture/gaze controls preliminary version*" that was produced in the framework of the EasyTV project.

6. KEYPOINT DETECTION

This chapter provides an analysis of the methods that were used to detect keypoints on the acquired images. In the current state of the capturing technology, we decided to use 2D keypoint detectors on RGB images and then infer the depth values in a following step.

6.1. Overview

There are many definitions for keypoints in literature. In fact, keypoints are usually referred to as interest points within an image. They are spatial locations, or points in the image that define what is interesting or what stands out in the image [1]. In other contexts though, they can be regarded as a set composed of (1) interest points, (2) corners, (3) edges or contours, and (4) larger features or regions such as blobs. The special characteristic about keypoints is that no matter how the image changes, i.e., whether the image rotates, shrinks/expands, is translated (i.e., if affine transformations are applied to the image) or is subject to distortion (i.e., a projective transformation or homography is applied to the image), one should be able to find the same keypoints in this modified image when comparing with the original image.

Keypoints are found by specific algorithms that process the image. Such types of algorithms that detect keypoints are called *detectors* and find application in a wide area of computer vision tasks, such as, image processing and analysis, object recognition and classification, and also, motion capturing. Some desirable properties of a keypoint detector are:

- Accurate localization.
- Invariance against shift, rotation, scale, brightness change.
- Robustness against noise, high repeatability.

6.2. Detecting 2D keypoints

Sign Language detection is a very demanding process requiring the motion capture of gestures, as well as, body movements and facial expressions. For that reason, we need to detect keypoints not only for the hands of the signer, but also for the body and the face. In order to accomplish that, three distinct keypoint detectors need to be integrated with the capturing module. In the current state of the module, we decided to use the *OpenPose* library [23] as our preferred solution for the implementation of the detection phase. Our research indicated that this is the state-of-the-art solution for unobtrusive hand detection, and hence, optimal for capturing the motion of Sign Language. *OpenPose* is also a practical fit for the EasyTV capturing technology because it only requires RGB images as input for detecting keypoints. Thus, the software can work with any camera available. Except for hand detection, it also includes robust keypoint detectors for the face and the body, making it a complete keypoint detection suite that doesn't require any extra third-party software to integrate with. In fact, *OpenPose* can detect a total of 135 keypoints on the acquired RGB images, which are sufficient for accurately capturing the motion of the signer.

Each keypoint detector in *OpenPose* is a deep neural network and, more specifically, a special type of CNN called *Convolutional Pose Machine* (CPN). CPNs have the ability to learn long-range dependencies among images and multi-part cues, and also, inherit a modular sequential design. These features combine with the advantages afforded by convolutional architectures, thus making the networks capable of learning feature representations for both image and spatial context directly from data. In the first stage, the convolutional pose machine predicts part beliefs from only local image evidence, while the convolutional layers in the subsequent stage allow the classifier to freely combine contextual information by picking the most predictive features. More comprehensive information about the architecture of a CPN can be found in [2].

In order to avoid the problem of annotating databases for hand detection, the training process of a CPN is done using a technique called *Multiview Bootstrapping*. While a thorough analysis of this method is provided in [3], we will also mention some important aspects of it for the completeness of

presentation. Multiview bootstrapping is an approach that allows the generation of large annotated datasets using a weak initial detector. More specifically, the weak detector is trained on a small annotated dataset in order to detect subsets of keypoints in the so called “good views” which are the views for a certain frame achieving the highest scores as evaluated by a heuristic scoring policy. A robust 3D triangulation procedure is then used to filter out incorrect detections. Images where severe occlusions exists are then labeled by reprojecting the triangulated 3D hand joints. The inclusion of the newly generated annotations in the training set, iteratively improves the detector, and thus, in each iteration we obtain more and more accurate detections. With this approach we generate geometrically consistent hand keypoint annotations using constraints from the multiple views as an external source of supervision. In this way, we can label images that are difficult or impossible to annotate due to occlusion.

More formally, if $d(I)$ is a keypoint detector on an image I, its output would consist of a set of tuples of position vectors x_p and confidence values c_p, with each tuple corresponding to a certain keypoint,

$$d(I) \mapsto \{(x_p, c_p)\ for\ p \in [1 \dots P]\}$$

An initial training set T_0 consists of annotated images with a predefined set of N_0 keypoints, i.e.,

$$T_0 \coloneqq \{(I^f, \{y_p^f\})\ for\ p \in [1 \dots N_0]\}$$

We train an initial detector d_0 by using stochastic gradient descent on dataset T_0,

$$d_0 \leftarrow train(T_0)$$

This detector is then used to produce a new dataset of labeled images, namely T_1. The bootstrap technique suggests that an improved detector can be generated if we construct a new training set by concatenating datasets T_0 and T_1,

$$d_1 \leftarrow train(T_0 \cup T_1)$$

This procedure continues until we find a keypoint detector that achieves high accuracy results on a given test set.

As we mentioned above, the detectors are trained on datasets containing only “good views”. This is an important point to note because the inclusion of erroneously labeled frames in the training dataset will lead the iterative process to failure. While the selection of the valid frames could be realised by using uniform temporal subsampling on the videos, another technique is used which segments the video into windows of W frames (e.g., W=15 or W=30) and picks only the best frame from each window. By “best” we mean the frame that has the maximum sum of detection confidences for the different views. The mathematical formula with which scores are given to frames, is:

$$score(\{X_p^f\}) = \sum_{p \in [1 \dots P]} \sum_{u \in I_p^f} c_p^u$$

Finally, the frames of the training set must be checked for errors. This validation is done manually by visual inspection of the top 100 frames in the training set.

As we mentioned earlier, there is a detector for each part of the signer's body. The hand detector finds 21 keypoints on each hand of the signer, that is, a total of $2 \times 21 = 42$ keypoints are detected (Figure 9). These include keypoints on the tips of the fingers, other articulation points on the arches, as well as, one point on each wrisp of the signer.

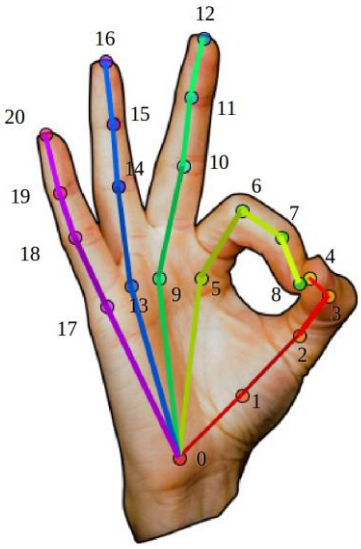

Figure 9: Hand keypoints.

The body detector detects 18 keypoints (Figure 10). Concerning the arms of the signer, keypoints for the wrisps, the elbows and the shoulders are detected. Keypoints for the hips, the knees and the feet are detected for legs. Also, keypoints for the neck and the head of the signer are detected.

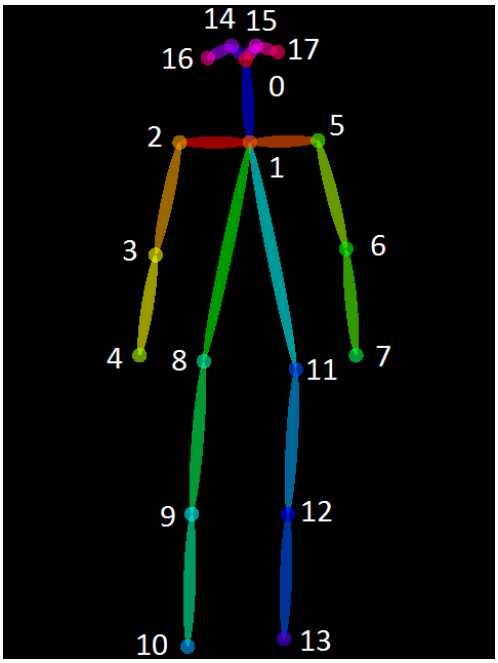

Figure 10: Body keypoints.

Finally, the face detector detects 70 keypoints (Figure 11). These keypoints include the contour of the face, the eye orbits and eyelids, two layers of points for the lips and also some points for the nose of the signer.

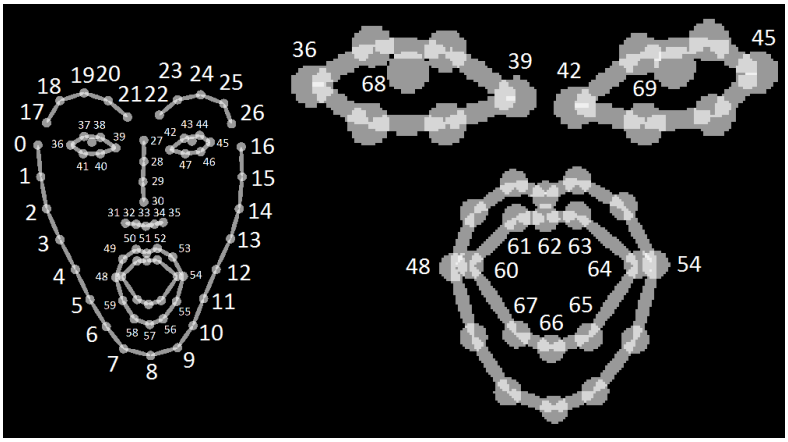

Figure 11: Facial keypoints.

In order to retrieve all the keypoints that are necessary for the accurate capturing of Sign Language motion, all three detectors must be executed upon each acquired frame. Considering that these detectors are deep neural networks and computations of just one deep network are heavy even in its inference phase, it is evident that the process of the overall keypoint detection is a very computational demanding task if done in parallel. Therefore, there is the option of executing the detectors in a sequencial manner, one after the other, or executing only two detectors in parallel. For example, we can execute the hand and body detectors first, and then the face detector by its own. As an example, we see in Figure 12 the hand and body detectors running in parallel and finding keypoints on a signer.

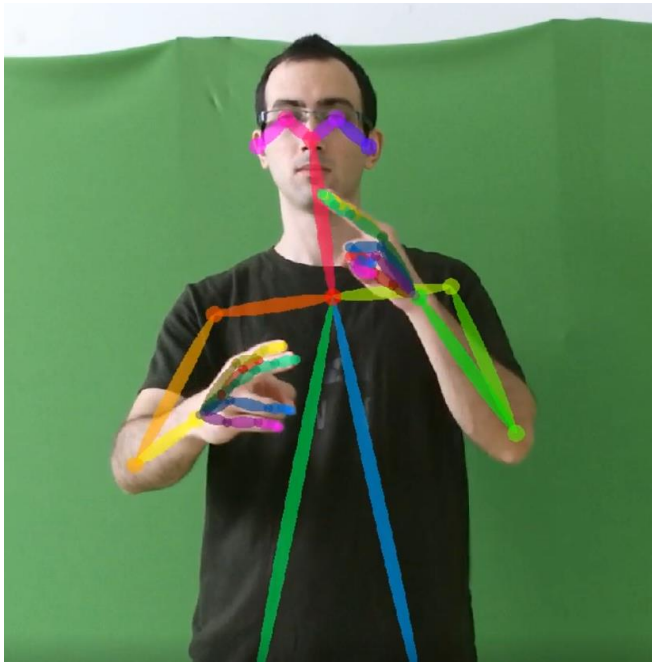

Figure 12: 2D keypoints detected for hands and upper body of a signer speaking Sign Language.

6.3. Generating 3D keypoints

In order to provide avatar playback, the keypoints extracted from the captured images have to be mapped to specific control points on the avatar. Using a 2D detection algorithm, we only get the values corresponding to the width and height of an image which is not appropriate for controlling a 3D avatar. We, therefore, need a technique to infer the depth dimension and, by using it, construct 3D keypoints.

As we mentioned earlier in section 3.2, it is more practical and reliable to use the aligned depth frames given by an RGB-D sensor over a multi-view 3D reconstruction solution. What we do is simply extracting the values from depth frames for each X and Y coordinate, and then merge all three values and store them in a single file.

Another issue that we have to consider is that in case the avatar has fixed control points, the generated 3D keypoints from the detection phase need to map the positions of those control points. For example, if an avatar has a fixed control point on the hip, then the detection algorithm needs also to find that specific keypoint on the acquired images. If a mis-alignment exists between the avatar control points and the detector keypoints, then we either have to re-train the detection algorithm or to perform mathematical transformations in order to re-position the skeleton points properly.

Figure 13 shows how the 3D keypoints look if we draw them using OpenGL. A signer made Sign Language gestures and images were captured using an RGB-D sensor. In the detection phase, we initially found hand and body keypoints from RGB data using detection algorithms, and then, the depth values from the aligned depth frames were extracted to form 3D data. The result shows that in the given frame all keypoints were accurately detected without being affected by occlusions or noise. For this demonstration though, we did not use the face detector so no facial keypoints are show in the picture.

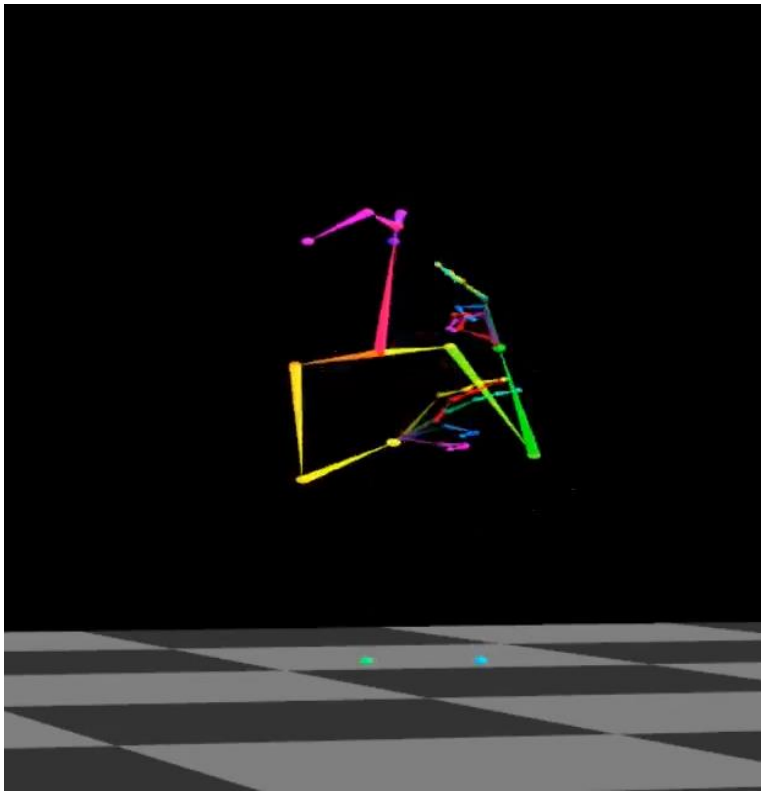

Figure 13: 3D keypoints drawn in OpenGL for Sign Language, formed by merging the 2D keypoints shown in Figure 11 with the corresponding values from the aligned depth frames.

7. MOTION REFINEMENT

In this chapter we describe the types of errors that can be encountered in the detection process and the techniques that can be applied in order to correct them.

7.1. Errors in motion data

Data from the detection process can contain errors. This means that either some keypoints are not detected by the algorithms, i.e., missing keypoints, or the algorithms produce erroneous values for some keypoints, i.e., mis-detections. The first case can occur due to occlusions by other body parts while the signer makes the gestures. For example, the gestures made for a certain sign might involve hiding some fingers behind the hand. When the acquired images are given to the detection algorithms, the keypoints of the hidden parts would not be detected. The second case can be encountered in the presence of noise. Such noise can be generated due to sensor specifications, room lighting conditions, or even colors of the clothes that the signer wears.

Noisy data can have negative effects when propagated in formated motion data. As it will be explained later in chapter 8, most motion file formats follow a hierarchical structure under which the position of each body part is defined as an offset from the previous one. Thus, errors in joint detection will affect other joints in the hierarchy as well. It should be apparent that under such a scheme, even a single error occuring on a keypoint value might disturb the skeletal structure and motion greatly, and produce unwanted results when the motion file is imported for avatar playback.

7.2. Keypoint reconstruction

The problems described in the previous paragraph can be solved by applying methods for missing marker reconstruction (due to literature terminology, in this section a marker refers to a keypoint not a sensing device). Although the problem of the missing keypoints consists an active research area, its solutions have been used for recovering human mocap data. Figure 14 demonstrates how such a refinement process can restore missing keypoints (see [4]).

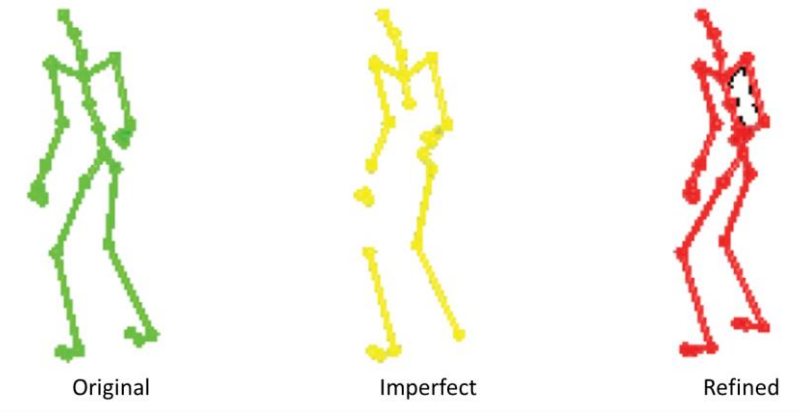

Figure 14: Green color: An ideal keypoint detection, Yellow color: An imperfect detection with missing keypoints, Red color: Keypoints after a motion refinement process.

A number of methods have been used in literature to solve the missing marker problem. Some of them include: the traditional interpolation which constructs new data points within a sequence of known data points, matrix factorization where the hierarchy of the body is used to break the motion into blocks, dictionary learning which aims at finding a sparse representation of the input data, Kalman smoothing which can predict values in a lower dimensional space, and a more recent neural network approach which is based on non-linear correlations within data.

8. EXPORT

This chapter discusses the files that are exported by the capturing technology, their internal format, and the EasyTV services that use them.

8.1. Types of files

The EasyTV capturing module will finally output a number of different file types. These files are essential to other EasyTV services and modules. Examples are the realistic 3D avatar and the multilingual ontology. The files are initially uploaded to the crowdsourcing platform and then stored into repositories in order to be accessible by these services and modules. Below are some types of files that are exported by the capturing module.

- Motion files

These files contain the motion data generated in the detection phase of this module, as described in chapter 6. The selection of the right file format affects the simplicity in calculations when creating the file and the availability of solutions when importing it for avatar playback. Considering the importance of these two aforementioned points, we devote the next section to this topic.

- Video files

These files concern videos containing the frames acquired by the RGB-D sensor. There are two types of videos exported: RBG and depth videos. File format can be either .avi or .mp4 (since .avi formats are larger than .mp4). They can be used in the validation phase of the crowdsourcing process by the moderator of the crowdsourcing platform, as well as, in search options for database content retrieval.

- Description file

This file is a sum-up of all the exported files. More specifically, it describes the mapping between motion files, video files, and user annotations which ascribe the meaning of each sign. Currently, we decided to use XML as the preferred format for these files due to its simplicity and also its straightforward API inclusion by most programming languages.

8.2. Choosing the right file format for motion data

Among all types of files generated by the EasyTV capturing module, the most important is the one containing the motion data. This data has been generated in the detection phase as described in chapter 6. Because this file will be used by the EasyTV realistic avatar for the playback of the signer's gestures and expressions, the choice of its format is essential.

There are many file formats for storing motion capture data. Amongst the most famous ones are the Biovision's Hierarchical format (BVH), as well as, the more modern and proprietary FBX format by Autodesk. Other formats include C3D, V/VSK, and ASF/AMC. A complete list with motion formats is given in [24]. These formats are supported by most modern motion capture technologies and 3D animation platforms. This makes them an attractive solution to intergate with the EasyTV realistic avatar.

As an example for explaining the structure of a motion capture file format, we will briefly describe the BVH format as presented in [5]. In this format, the data is arranged hierarchically, with every joint position depending on the previous one. More specifically, the hierarchical section of the file starts with the keyword HIERARCHY which is followed by the keyword ROOT and the name of the bone that is the root of the skeletal hierarchy. For the definition of each bone, the line delimited by the keyword OFFSET refers to the translation of the origin of the bone with respect to its parent's origin along the x, y and z-axis, respectively. The offset is also used for implicitly defining the length and direction of the parent's bone. There is a problem, however, when determining the length and direction of bones that have multiple children. In that case, a good choice for determining the length of the bone is to use the first child's offset definition in order to infer the parental bone

information and then treat the offset data for other child nodes simply as offset values.

The other line concerning the bone's definition, starts with the word CHANNELS and is followed by a value that defines the DOFs for the current bone. The channel data is given in the motion section at the end of the file. We have to note here that the order with which each channel is seen in the hierarchy section of the file, exactly matches the order of the data in the motion section of the file. For example, the motion section of the file contains information for the channels of the root bone in the order defined in the hierarchy, followed by the channel data for it's first child, followed by the channel data for that child, and so on, through the hierarchy. Also, the order concatenating the Euler angles when creating the bone's rotation matrix needs to follow the order defined in the CHANNEL section. The Euler order refers to each bone, therefore different orders for different bones could possibly affect the correct looking of an animation. After the OFFSET and CHANNEL lines, the next non-nested lines in the bone definition are used to define child items, starting with the keyword JOINT. However, in the case of end-effectors, a special tag is used, i.e., "End Site", which encapsulates an OFFSET triple that is used to infer the bone's length and orientation.

After the definition of the skeletal hierarchy, another section withith the BVH file contains the motion information. This includes: the number of frames in the animation, frame rate and the channel data. The number of frames and frame rate are represented by two numerical values, namely, a positive decimal integer and a positive float, respectively. The rest of the file contains the channel data for each bone in the order that they are seen in the hierarchy definition. That is, each line of float values represents an animation frame. An example demonstrating the internal structure of a BVH file is given in Figure 15.

```
HIERARCHY
ROOT hip
{
  OFFSET 0 0 0
  CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
  JOINT abdomen
  {
    OFFSET 0 20.6881 -0.73152
    CHANNELS 3 Zrotation Xrotation Yrotation
    JOINT chest
    {
      ...
            OFFSET 4.14528 8.04674 8.04672
            CHANNELS 3 Zrotation Xrotation Yrotation
            End Site
            {
              OFFSET 1 0 0
            }
          }
          JOINT rightEye
          {
            OFFSET -3.6576 8.04674 8.04672
            CHANNELS 3 Zrotation Xrotation Yrotation
            End Site
            {
              OFFSET 1 0 0
            }
          }

      }
      JOINT rCollar
      {
        OFFSET -2.68224 19.2634 -4.8768
        CHANNELS 3 Zrotation Xrotation Yrotation
        JOINT rShldr
        {
          ...
        }
      }
      ...
    }
    ...
}
MOTION
Frames: 2752
Frame Time: 0.00833333
53.6842 83.8008 -93.0874 0.0 0.0 0.0 -2.04814 -0.0011253 -0.0554687 -0.56432
0.00679362 -0.0976938 -4.42258e-14 -7.95139e-16 -1.2424e-16 -1.32679e-13
-6.36111e-15 -3.71168e-16 -0.0 0.0 0.0 -0.0 0.0 0.0 -2.2117e-14 -2.44598e-17
9.93923e-17 -4.58949e-13 -6.4605e-16 -1.78906e-15 -7.9495e-12 2.36931e-12
3.81667e-14 -9.11375e-11 -2.48058e-12 -3.53042e-13 -2.14952e-10 -2.41345e-13
-6.45653e-13 -3.71218e-10 -9.11011e-11 2.28935e-12 -3.75305e-13 -2.14928e-10
```

Figure 15: An example BVH file.

Another popular format for storing motion data is the JSON format. This format is similar to XML, making it even more practical for integration and usage. Such file format is supported by UNITY which is one of the most famous game engines existing. With UNITY, one can develop realistic 3D avatars capable of importing JSON files for motion playback. One such example of JSON file is presented in Figure 16. More information concerning avatar playback is provided in section 8.5.

```
{
   "keypoints" : [
      {
         "Left_Hand" : [
            {
               "X" : "-72.7222",
               "Y" : "11.0613",
               "Z" : "-17.6048",
               "keypoint_id" : "1"
            },
            {
               "X" : "-72.2793",
               "Y" : "12.1006",
               "Z" : "-17.4841",
               "keypoint_id" : "3"
            },
            {
               "X" : "-72.2954",
               "Y" : "12.605",
               "Z" : "-16.8757",
               "keypoint_id" : "4"
            },

            ...

         ],
         "timestamp" : 1537268535254
      },
      {
         "Right_Hand" : [
            {
               "X" : "-67.6528",
               "Y" : "10.8201",
               "Z" : "-18.8874",
               "keypoint_id" : "1"
            },
            {
               "X" : "-67.9415",
               "Y" : "11.6766",
               "Z" : "-19.1585",
               "keypoint_id" : "2"
            },
            {
               "X" : "-68.0413",
               "Y" : "12.2864",
               "Z" : "-19.1948",
               "keypoint_id" : "3"
            },

            ...

         ],
         "timestamp" : 1537268535254
      },
      {
         "Body" : [
            {
               "X" : "-70.3623",
               "Y" : "22.6794",
               "Z" : "-7.5761",
               "keypoint_id" : "0"
            },
            {
               "X" : "-70.3019",
               "Y" : "15.5137",
               "Z" : "-9.54713",
               "keypoint_id" : "1"
            },
```


Figure 16: An example JSON file for motion capture data.

8.3. Uploading the files to the crowdsourcing platform

When the files are exported by the capturing module, they must be uploaded to the EasyTV crowdsourcing platform in order to be available to other EasyTV services and modules. The exported files are the result of a crowdsourcing task that the user has completed. The crowdsourcing platform contains a special form for uploading the files. This form is part of the UI that concerns the task that is appointed to this user. The different file types make it necessary to import one compressed folder instead of a number of different files. After this folder is uploaded, the crowdsourcing platform should decompress it, extract the files and proceed to the validation process. If the moderator validates the content of the files and accepts them as being the correct answer to the given task, the files are stored into the proper repositories. In the opposite case, that is, if the moderator rejects the motion capturing for the given task, the user has to repeat the task or the task may be assigned to a different user.

8.4. Input to multilingual ontology

The capturing module also sends information (via the crowdsourcing platform) of the recorded video to the multilingual ontology for the enrichment process being developed in the Task 3.2. A first proposal of an exchange JSON format is presented in Figure 17. The JSON file must contain information about the language of the recorded video, the oral language sentence that represents the video and the sentence composed by the concatenation of signs, which cannot be the same as oral language. Moreover, information about each video segment is provided with the start and end in the video and their meaning. In addition, an URL where the video is stored should be provided.

The JSON file will be processed by the easyTV-annotator library to enrich the multilingual sing language ontology (see document D3.2 Enriched multilingual ontology with signs in different languages preliminary version). The library processes the natural language sentence to retrieve linguistic aspects such as the tokens, the part of speech of the words, their lemmas and whether or not there is a named entity. This information is needed to find the suitable concepts in the ontology that represent each of the words that compose the sentence. Then, the class of the ontology is associated with the video segment received in the JSON file.

```
{
    "video": {
        "url": "http://.....",
        "nls": "the tree elephants",
        "sls": "elephant the three" ,
        "duration": "00:50" ,
        "language": "es" ,
                "segmnents": [  {
                "order": "1",
                "start": "00:00",
                "end" : "00:30",
                "content" : "elephant" ,
                }
                {
                "order": "2",
                "start": "00:31",
                "end" : "00:40",
                "content" : "the" ,
                }
                {
                "order": "3",
                "start": "00:41",
                "end" : "00:50",
                "content" : "three" ,
                }]
        }
}
```

Figure 17: An example of a JSON format for describing Sign Language content.

8.5. Avatar playback

The visualization of the recorded signs will take place via a humanoid avatar. Adobe Fuse CC is the 3D computer graphics software that was chosen to create the avatar at least at this preliminary stage of development as it is easy to design a desirable character following some basic guidelines, especially regarding the quality of signs visibility, i.e., black shirt. The main advantage of Fuse is that the user can choose and modify character components, such as body parts or clothes, in real-time. Figure 18 illustrates the avatar creation process in Fuse.

When the design is completed, the character is imported in Mixamo, a 3D software engine related to Fuse, in order to use the Rigging service. Rigging is a method to create a skeleton for a 3D model by constructing a series of bones so it can be animated and move. Each bone has a three-dimensional transformation (which includes its position, scale and orientation), and an optional parent bone and therefore they form a hierarchy. So, moving a shoulder-bone will move the rest of the hand too. The bones are connected with each other through joints. The rigging allows a 3D model to be animated in an articulated manner. An articulation is a rotation/translation of a joint which moves a connected bone. On the other hand, the pose is a set of joint articulations which results in positioning the articulated body. In the case of an avatar, the rigging is the skeleton that ties in to the human posture. Mixamo's technologies use machine learning methods to automate the steps of the character animation process, including 3D modeling to rigging and 3D animation.

Another advantage of the software in question is a basic facial animation with blended shapes, which can be used for lip sync. Generally facial expressions can be achieved either with bone transformation or blend shapes. In the first case, bones can be moved to specific directions and the combination of whole movements form an expression. Moreover, the method is based on controlling bone rotations when moving between different positions. On the other hand, blend shapes create the illusion that one shape changes geometrically into another in a natural-looking way. BlendShape is a technique of allowing a single mesh to deform in order to achieve numerous pre-defined shapes and any number of combinations of in-between these shapes. Again, the combination of blending shapes can animate an avatar's mouth to open or smile. The final result of this procedure is a rigged 3D model with blend shapes, ready to be imported in Unity 3D as Mixamo supports the specific platform.

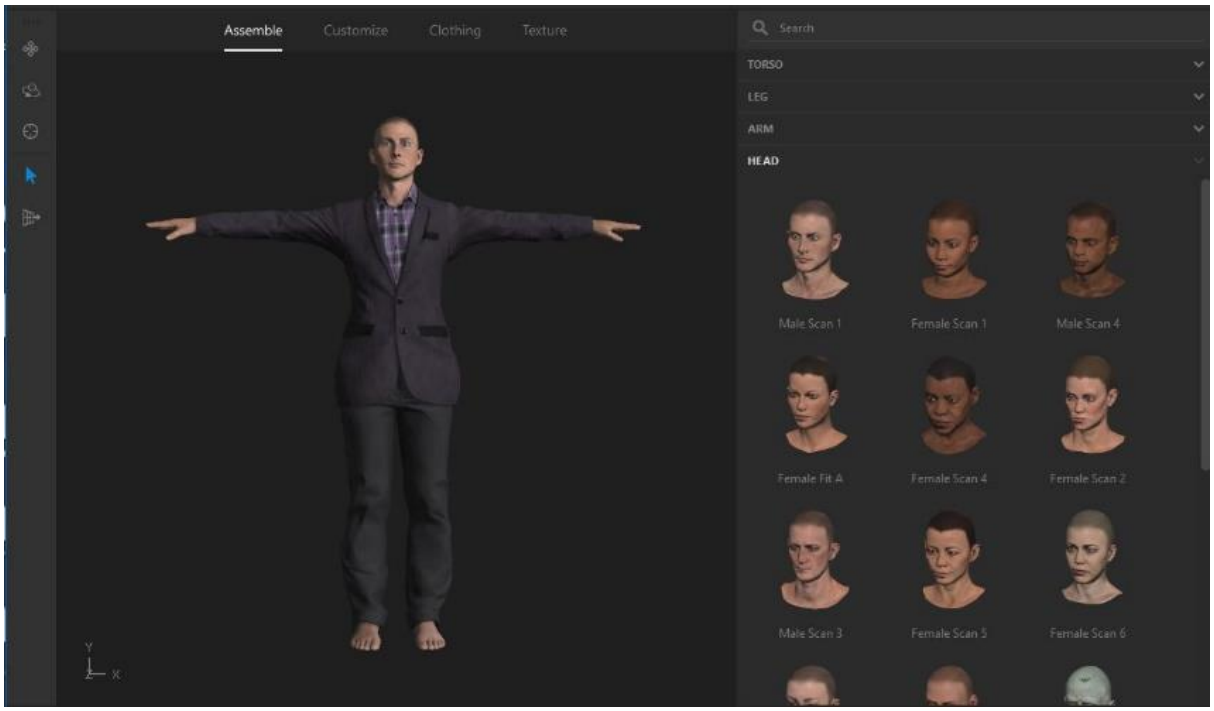

Figure 18: Adobe Fuse CC 3D Model.

The rigged 3D avatar is imported into Unity3D as an object, ready to be modified. Unity3D is a cross-platform game engine used to create both three-dimensional and two-dimensional games, as well as, simulations for desktops and laptops, home consoles, smart TVs, and mobile devices. Inside the editor of Unity3D the avatar will be modified in order to reproduce the recorded signs. Regarding avatar's rigging, few changes need to be made first in order to give a humanoid aspect to the character motion, i.e., definition of a humanoid animation type in the rig option. Files with positional data in JSON format, as mentioned before, are loaded into the avatar, pointing out target positions for each joint and specific timestamps. Figure 16 shows the specific format in which the data is written for Unity3D.

Inverse Kinematics (IK) method is used in order to make the coordinates of each point reach a target configuration. Forward kinematics uses the joint parameters to compute the configuration of the pose, whereas inverse kinematics reverses this calculation to determine the joint parameters that achieve a desired configuration, having as main objective to reach the desirable position. The synchronization of frames per second between recorded data and reproduction is very important for a normalized movement. Even if the sync is perfect, an issue may occur regarding the physics, resulting in image jitter. In order to offer a smoother image and avoid jittering, interpolation must be applied between each different position. The whole process mentioned above consists of a group of C# scripts useful for the control for loading data in correct manner, smooth motion and synchronization.

Despite the usage of JSON files, the main objective is to match each recorded keypoint of the body detector (as described in section 6.2) to each joint in the rigged body of the avatar. Generally, a rigged body in 3D modeling has a root joint which is the base of the structure. Usually for the humanoid avatars, the center of the hips constitutes the root joint. However, as it can be seen on Figure 9, there is not that kind of keypoint. Moreover there are more keypoints recorded in the head which do not exist in the avatar. As a consequence, additional effort must be applied in order to avoid wrong reproduction of the recorded signs. An example of a signing 3D avatar is shown in Figure 19.

Figure 19: Signer avatar interprets the word "name".

9. THE EASYTV SIGNER3D APPLICATION

This chapter presents a first implementation of the EasyTV capturing technology.

9.1. Overview

The Signer3D application is the EasyTV motion capture technology. It is a desktop application that implements all six phases described in this document. It connects to an RGB-D sensor and produces the data required for the other EasyTV services.

In the following, a brief description about the Signer3D application is presented. This description includes an initial design of the graphical user interface, and an outline of the first stages of the capturing phases that have been implemented. A more detailed description of the software will be given in the final version of the deliverable, that is, "*D3.7 Sign language capturing technology final version*".

9.2. User interface

The Signer3D application consists of a user-friendly interface (GUI) that allows the admin of the motion capture process to easily capture signs and extract the motion data.

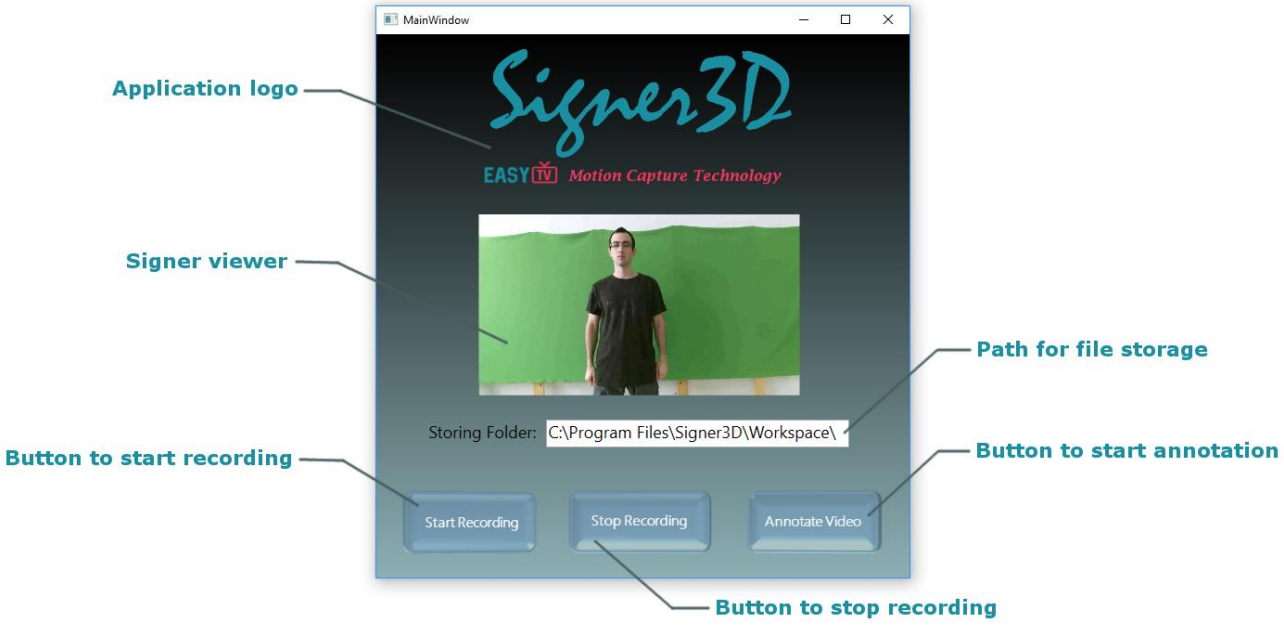

Figure 20: A preliminary version of the *Signer3D* main user interface.

As we can see in Figure 20, the main window of the application includes:

- Application logo.
- Signer viewer: A display screen for visualizing the signer while recording. This is important so that the admin can see if the signer is in the right position and his moves do not exceed the sensor's viewing area. Also, the effects that the lighting conditions of the room may have on the sensor can be observed.
- Path for file storage: A textbox for defining the folder where data will be written into. This data concerns both the acquired images and the final output of the application.
- Button to start recording: The admin starts the image acquisition phase.

- Button to stop recording: The admin stops acquistion and files are stored in the specified folder.
- Button to start annotation: This button proceeds to the annotation phase where the admin has to annotate the captured video.

9.3. Implementing the motion capturing phases

The Signer3D application implements the phases of capturing as described in the previous chapters of this document. In this, initial version of the application, the current state of the phases' implementation is as follows:

- *Acquisition*: This phase is completed. Both RGB and depth images are acquired by an RGB-D sensor and stored as .jpg files in the folder specified by the user in the main window of the application. Also, the alignment between RGB and depth frames has been accomplished and frames are stored at the same resolution.
- *Annotation*: User annotation is provided only for the whole video. Isolation of signs and synchronized trimming of the videos are not yet supported.
- *Detection*: In this phase, we analyze RGB data and use 2D keypoint detectors to extract keypoints for the hands, face and body of the signer.
- *Reconstruction*: For the detection process that we followed, the 3D reconstruction phase has been implemented. 3D data is formed by exploiting the aligned depth frames.
- *Refinement*: This phase will be available in the final version of the application.
- *Export*: The application currently exports the captured RGB and depth video. Motion data is also exported in JSON format that can be imported by UNITY as described in section 8.2. The description file is not yet supported.

9.4. First use: creating a database for Greek Sign Language

The first version of the Signer3D application was used for the creation of a database for Greek Sign Language. The database consisted of just RGB and depth images and was created with the contribution of the Centre of Greek Sign Language [25] located in Thessaloniki, Greece. The task for every expert signer was to perform a set of dialogs, i.e., questions and answers that are commonly encountered in Greek public services. Figure 21 shows some photos from that capturing process.

Figure 21: Creating a database for Greek Sign Language using the preliminary version of *Signer3D*.

10. CONCLUSIONS

This document presented the initial design, research and development steps for the implementation of the EasyTV capturing technology. As it is apparent throughout the document, the requirements for unobtrusive motion capture of Sign Language exclude marker-based sensors from our candidate list of hand tracking technologies and constrain the available solutions for reaching an optimal setup for accurate motion capturing. These constraints made us head towards the choice of a markerless solution composed by just one RGB-D sensor. Moreover, the structure of the EasyTV project, as outlined in the DoA, lead us to define a multi-phase architecture for the development of the capturing module; an architecture that also fits the markerless setup and aims to provide accurate and robust motion capture results for Sign Language. At the heart of our multi-phase architecture lies the keypoint detection phase, where motion analysis algorithms process the acquired frames and find specific points of interest for the hands, face, and body of the signer. While, obviously, a better training procedure of a detector leads to better detection accuracy, it is also crucial to consider the fluctuating effects that the quality of 3D data causes on realistic avatar performance. For that reason, we augmented our architecture by adding an extra post-processing step: the motion refinement phase. We believe that by developing a robust refinement algorithm tailored to 3D data for Sign Language, we can reach the accuracy of state-of-the-art motion capture systems while being oriented towards Sign Language.

In the final version of this deliverable, a more detailed and thorough analysis of the Signer3D application will be presented. The next steps towards the completion of the application concern the implementation of the missing parts of the motion capturing phases and also, the testing of the final application. In particular, the annotation phase will be extended to support video trimming capabilities so that a video can be segmented into separate sign recordings and be annotated accordingly. Moreover, we aim to provide enhanced versions of the detection algorithms, especially for hand detection. Hand pose estimation algorithms are a very active research area, and thus, we will try to elaborate on results like those produced in [26]. In accordance with T2.4, we will discuss the requirements and constraints that avatar technologies impose on keypoint positions and define keypoints that accurately map to avatar control points. For the refinement phase, we aim to implement a technique that will produce robust results when 3D data are imported for avatar playback. Finally, concerning the exported file formats, we will support description files that aggregate all information of other exported files and provide mappings between text, video and motion files.

11. REFERENCES

[1]. S. Krig, “Interest point detector and feature descriptor survey,” in Computer Vision Metrics. Springer, 2014, pp. 217–282.

[2]. S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In CVPR, 2016.

[3]. T. Simon, H. Joo, I. Matthews, and Y. Sheikh, “Hand keypoint detection in single images using multiview bootstrapping,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017.

[4]. Zhao Wang, Shuang Liu, Rongqiang Qian, Tao Jiang, Xiaosong Yang, and Jian J Zhang. 2016. Human motion data refinement unitizing structural sparsity and spatial-temporal information. In IEEE Int. Conference on Signal Processing.

[5]. M. Maddock and S. Maddock, “Motion capture file formats explained,” Department of Computer Science, University of Sheffield, 2001.

[6]. D. Konstantinidis, K. Dimitropoulos and P. Daras, “A deep learning approach for analyzing video and skeletal features in sign language recognition,” IEEE International Conference on Imaging Systems and Techniques (IST), Krakow, Poland, October 2018.

[7]. http://www.cyberglovesystems.com/

[8]. https://manus-vr.com/

[9]. https://vrgluv.com/

[10]. https://www.synertial.com/

[11]. https://www.vicon.com/

[12]. https://ar-tracking.com/

[13]. http://www.htcvr.com

[14]. https://www.kickstarter.com/projects/576456616/imotion-3d-motion-controller-with-haptic-feedback

[15]. https://www.leapmotion.com

[16]. http://research.microsoft.com/en-us/projects/handpose/

[17]. http://www.mindmaze.ch

[18]. https://www.thalmic.com

[19]. http://nimblevr.com

[20]. https://www.oculus.com/blog/vrs-grand-challenge-michael-abrash-on-the-future-of-human-interaction/

[21]. https://en.wikipedia.org/wiki/Triangulation_(computer_vision)

[22]. https://www.ffmpeg.org/

[23]. https://github.com/CMU-Perceptual-Computing-Lab/openpose

[24]. https://en.wikipedia.org/wiki/List_of_motion_and_gesture_file_formats

[25]. https://www.keng.gr/

[26]. http://icvl.ee.ic.ac.uk/hands17/challenge/