
UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona


# Towards video alignment across cameras with sign language 2D poses

Bachelor degree Thesis
submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
by

Andrea Iturralde Amigó

In partial fulfillment
of the requirements for the degree in
**Telecommunications Technologies and Services Engineering**

Advisors: Xavier Giró Nieto, Laia Tarrés Benet
Barcelona, Date 11/10/2021

# Acknowledgements

I really want to thank my tutor for this project, Xavier Giro, for believing in me and counting on me when not many people did. He has always been very supportive and positive about even the slightest improvement and I am very grateful for that.

I would also like to thank Laia Tarrés, not only as a supportive PhD student but as a friend, for everything she has had to put up with and still being the best at it.

And last but not least, I would like to thank my dad, for being the motor of my motivation, the reason why I am doing this in the first place and the biggest support I could ask for. I can finally say it, I am an engineer like Papo.

# Abstract

This thesis degree is part of a project from the Image Group at UPC that is focused on sign language translation using deep learning technologies. This thesis builds on top of an existing database called How2Sign, that contains more than 83 hours of sign language translation videos.

This database has some textual annotations aligned to a front RGB camera. The same scenes are also captured by a side RGB and a front RGB-D cameras. These three cameras are not synchronized, so it is necessary to align the segments annotated on the RGB front camera to the other cameras. This thesis explores a solution based on the cross correlation operator.

Our work is to process the coordinates of the joints of the subject that appears in the videos, not from the point of view of image or video processing based on pixels.

The first part if this thesis is to investigate the properties of the cross-correlation function by locating short video segments of a long recording based on automatically extracted 2D human poses. The experiments studied the impact of adding noise.

The second part applied the cross-correlation to try to align two videos with the same scene, but recorded with different cameras from different points of view.

# Resumen

Esta tesis de final de grado forma parte de un proyecto del Grupo de Procesado de Imagen de la UPC enfocado a la detección de lenguaje de signos utilizando tecnologías relacionadas con deep learning. Este proyecto ya consta con una base de datos llamada *How2sign*, que contiene más de 83 horas de videos de traducción de lengua de signos.

Esta base de datos contiene anotaciones textuales alineadas a una cámara RGB frontal. Las mismas escenas también son capturadas por una RGB lateral y una RGB-D frontal. Estas tres cámaras no están sincronizadas, con lo cual es necesario alinear los segmentos anotados de la RGB frontal con las demás. En esta tesis se explora una primera solucion basada en la correlación cruzada.

Nuestro trabajo consiste en procesar los puntos de las coordenadas de las articulaciones del sujeto que aparece en los videos, no desde el punto de vista de procesado de imagen o video basado en píxeles.

La primera parte de esta tesis es investigar las propiedades de la función de correlación cruzada mediante la localización de segmentos cortos de vídeo de una grabación larga basada en la extracción automática de las poses en 2D. Los experimentos también estudian el impacto de añadir ruido.

La segunda aplica la correlación cruzada para intentar alinear dos videos con el mismo contenido, pero grabados con distintas cámaras desde distintos puntos de vista.

# Resum

Aquesta tesi de final de grau forma part d'un projecte del Grup de Processament d'Imatge de la UPC enfocat a la detecció de llenguatge de signes utilitzant tecnologies relacionades amb deep learning. Aquest projecte ja consta amb una base de dades anomenada *How2sign*, que conté més de 83 hores de vídeos de traducció de llenguatge de signes.

Aquesta base de dades conté anotacions textuals alineades a una càmera RGB frontal. Les mateixes escenes també són capturades per una RGB lateral i una RGB-D frontal. Aquestes tres càmeres no estan sincronitzades, amb la qual cosa és necessari alinear els segments anotats de la RGB frontal amb les altres. En aquesta tesi s'explora una primera solució basada en la correlació creuada.

El nostre treball consisteix a processar els punts de les coordenades de les articulacions de l'subjecte que apareix en els vídeos, no des del punt de vista de processament d'imatge o vídeo basat en píxels.

La primera part d'aquesta tesi és investigar les propietats de la funció de correlació creuada mitjançant la localització de segments curts de vídeo d'una gravació llarga basada en l'extracció automàtica de les poses en 2D. Els experiments també estudien l'impacte d'afegir soroll.

La segona aplica la correlació creuada per intentar alinear dos videos amb el mateix contingut, però gravats amb diferents càmeres des de diferents punts de vista.

# Contents

# List of Figures



# List of Tables

# Abbreviations

**ASL** American Sign Language

**DL** Deep Learning

**ETSETB** *Escola Técnica Superior d'Enginyeria de Barcelona*

**GPI** *Grup de Processat d'Imatge,* Image Processing Group

**RGB** Red Green Blue

**RGB-D** Red Green Blue + Depth

**SL** Sign Language

**UPC** *Universitat Politècnica de Catalunya*

# Revision history and approval record

| Revision | Date | Purpose |
|---|---|---|
| 0 | 20/09/2021 | Document creation |
| 1 | 24/09/2021 | Document revision |
| 2 | 27/09/2021 | Document revision |
| 3 | 04/10/2021 | Document revision |
| 4 | 08/10/2021 | Document revision |
| 5 | 09/10/2021 | Document revision |
| 6 | 10/10/2021 | Document revision |

DOCUMENT DISTRIBUTION LIST

| Name | e-mail |
|---|---|
| Andrea Iturralde | andrea.iturralde@estudiantat.upc.edu |
| Xavier Giró | xavier.giro@upc.edu |
| Laia Tarrés | laia.tarres@upc.edu |

| Written by: | | Reviewed and approved by: | |
|---|---|---|---|
| Date | 20/09/2021 | Date | 10/10/2021 |
| Name | Andrea Iturralde | Name | Xavier Giró |
| Position | Project Author | Position | Project Supervisor |

# 1 Introduction

## 1.1 Purpose

Sign Language (SL) is the main, and most times the only, form of communication of people with hearing problems. Nowadays, there are around 466 million people in the world with different degrees of hearing difficulties, of which the vast majority use SL to communicate. The limitation of not being able to fully or partly use spoken language has been addressed in different methods and technologies to improve their accessibility and try to make their lives easier.

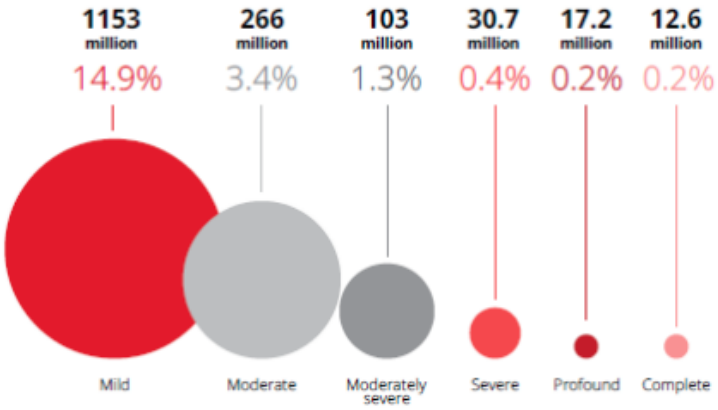


Figure 1: Classic motivation: Accessibility

A survey conducted in 2009 by the World Federation of the Deaf [1] revealed that more than 60% of the countries that answered, did not have access to professional sign language in education and health services. Also, professional sign language interpreters are even more scarce in underdeveloped countries.

For this reason, many people are marginalized and discriminated against by systems that do not provide them the right public services. This creates a huge communication barrier between people with hearing impairments and their daily lives activities.

The situation has worsened recently due to the COVID-19, where most people wear a mask most of the time they are talking, preventing lipreading, a skill that many deaf people have developed to understand non-SL speakers. In addition, due to COVID-19 restrictions, it is very complicated to have in-person interpreters in several fields like health or education, which makes it impossible for people to communicate in any way.

In the past 7 years or so, deep learning (DL) has grown exponentially and it is more present in our lives every day. It has been proved that DL is a very good solution to image recognition and processing problems, so why not try to adopt the principles that have been successful to try to solve language translation problems.

Interaction with computers and phones has been going into the direction of talking to

them. We interact with them mostly by voice, but why not start to interact with them by signs? Besides the clear problem about people with hearing difficulties not being able to communicate, there are a lot more advantages to begin communicating by signs. Sometimes when using speech, is not suitable for the situation, like background noise, the best solution is to sign.

So, why is DL the best solution for this? Mainly because in sign language there is not a 'lookup table' where we can just match a sign with a letter or word, and with the help of machine learning we can create a tool that automates the process and gives people with hearing problems a solution to the communication gap problem.

To help solve this problem, the UPC GPI has started the *Sign Language Recognition, Translation and Production* [2] project, to achieve bidirectionality between speech and sign language.

The first of the steps to achieve this goal was to create a dataset called How2Sign[3] [4].

The How2Sign dataset consists of a set of speech and transcriptions of instructional videos and their corresponding ASL translation videos and annotations. It is basically a collection of 83 hours of instructional videos that were translated to American sign language and recorded from different viewpoints including a depth sensor, the corresponding glosses and speech alignment thanks to the alignment to the how2 dataset It covers more than 35000 sentences and more than 16000 english words.

The instructional videos translated into ASL come from the existing How2 dataset [5] [6], a public multimodal and multilingual data set that provides us with the original spoken video and the english transcriptions uploaded by the users. Following the same splits from the How2, we selected a 60-hour subset from the training set and the complete validation and test sets to be recorded.

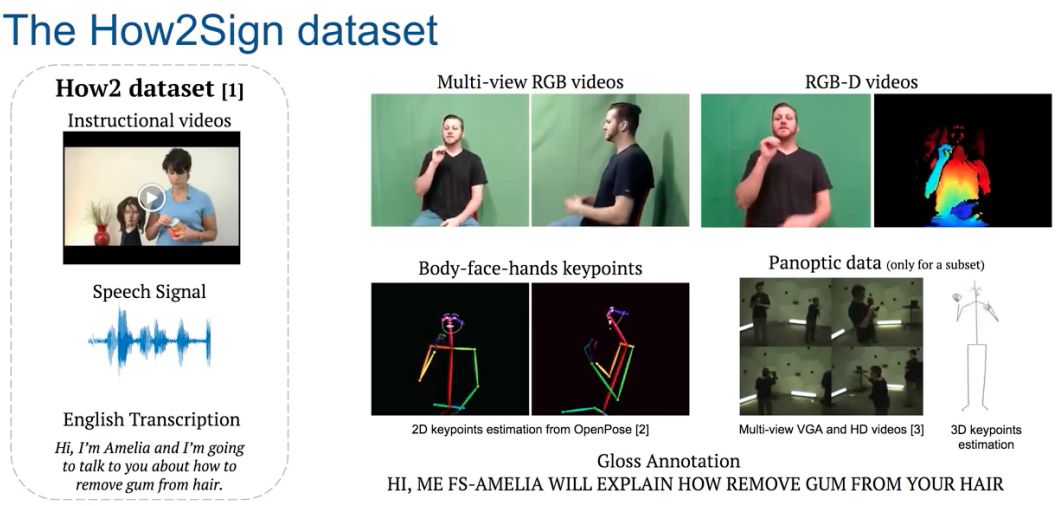


Figure 2: How2Sign dataset representation

To the best of our knowledge, How2Sign is the largest publicly available SL dataset across

languages in terms of vocabulary, as well as the largest ASL dataset in terms of video duration.

This thesis addresses the alignment of two video recordings, recorded from different cameras and points of view.

The first camera is frontal and records in RGB and the second camera is positioned slightly to the right of the subject and slightly further away and records in RGB-D. The reason for aligning them is that the RGB camera recordings contain glosses, which are the annotations corresponding to the signs being played back, and the RGB-D one does not, but the second camera contains information about the depth of the subject in the video, so it is much more useful for future operations. If we manage to align the recordings from both cameras, we will be able to align the textual transcriptions with the RGB-D camera, without the need of manual transcription.

## 1.2 Requirements and specifications

As mentioned before in the Purpose section, we dispose of videos recorded from different angles and with different cameras. The problem that appears right now is that only a group of videos, those recorded with the RGB camera and referred as *recordings* earlier, contain the English transcriptions, and our objective would be to be able to align them temporarily with the rest of the videos, referred as *videos* earlier, the ones recorded with the RGB-D camera, so we would not have to re-annotate the videos with english transcriptions, as it involves a lot of manual work.

The videos recorded with the RGB-D camera do not contain these annotations, but they are the most useful videos for 3D pose estimation, since they contain depth information, hence the alignment problem.

There is a misalignment between the RGB and depth sensors, so the alignment problem is not only temporary, but also spatial.

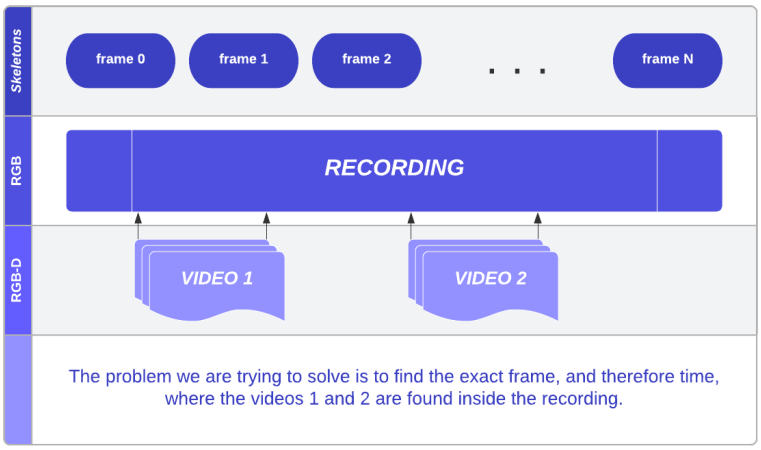


Figure 3: Video alignment model

The first part of this project consists in studying how the cross-correlation varies between frames of the same video, calculating it based on processing the coordinates of the joints of the subject that appears in it.

In this first part, it is also evaluated how the cross-correlation changes in the presence of noise, and how it improves or worsens depending on the frame in which it is executed and the body parts the thesis used in the processing.

The second part of my project consists of aligning a video clips, which from now on we will call *videos*, captured with an RGB-D camera, with a longer recording of the same scene, which from now on we will call *recording*, captured with an RGB camera.

Solving the alignment would allow training DL sign translation models that could benefit from RGB-D sensosrs.

The software specifications include programming the code in Python and then uploading it to Github together with the necessary documentation for further use for future work of the project.

## 1.3 Methods and procedures

The methods and procedures followed in this project, which will be detailed later on in the Methodology section, are depicted in Figure 4:

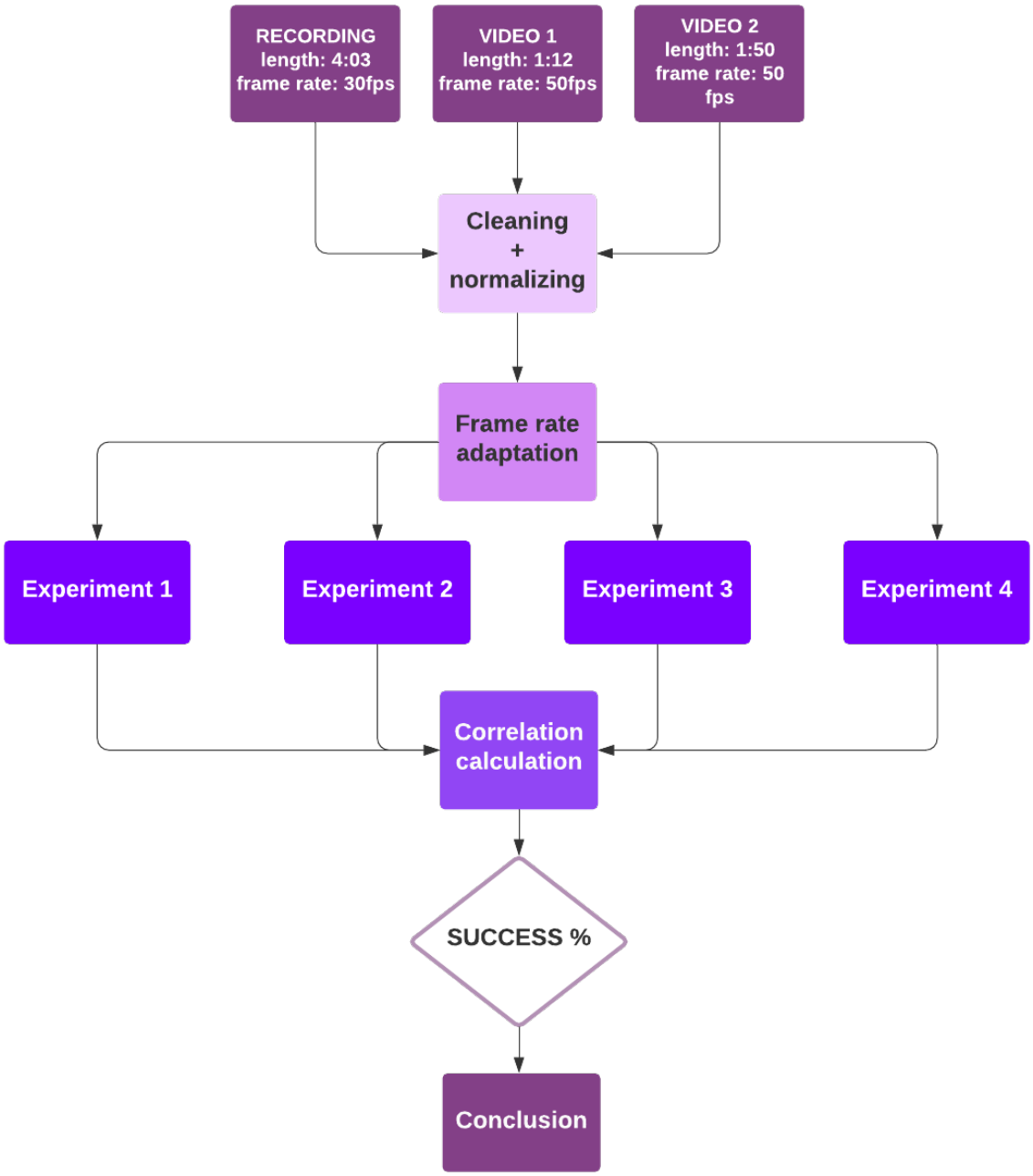


Figure 4: Methodology

As we can see in the figure above, the process to follow will be the following: to clean the body poses on the video and normalize their values, to adapt the frame rates of the three videos, to experiment with the different sets of poses, and finally to calculate the cross-correlation between the *videos* and the *recording*.

For the first part of the project, the video to put to test is video 1, the procedure to be followed will be as depicted above.

## 1.4 Work Plan - Gantt Diagram

The work plan of the whole project is the following:

1. WP1 Learning the framework of the project
   (a) T1.1 Sign up to Imatge UPC Slack
   (b) T1.2 Learn Python and Google Colab
   (c) T1.3 Understand the datasets of the Image Processing group in CALCULA
   (d) T1.4 Study options for video alignment
2. WP2 Test Openpose's running process
   (a) T2.1 Copy the videos and recordings
   (b) T2.2 Visualize the videos before processing
   (c) T2.3 Try to extract the poses from Openpose
   (d) T2.4 Find the pre-computed skeletons
   (e) T2.5 Try running Openpose with different initialization instructions
   (f) T2.6 Compare the results between the two
   (g) T2.7 Understand the JSONs generated by Openpose
   (h) T2.8 Watch the video with the skeletons on top
3. WP3 Adapt the *videos* and *recording*'s properties
   (a) T3.1 Check if the frame rate for both cameras is the same
   (b) T3.2 Adapt the frame rates
   (c) T3.3 Downsample de JSON files
   (d) T3.4 Remove points from skeleton and check output
   (e) T3.5 Plot the skeletons from the JSON files
   (f) T3.6 Change the rate of the poses in the output JSON file
4. WP4 Test autocorrelation on one video
   (a) T4.1 Run the auto-correlation on one video and extract conclusions
5. WP5 Clean, normalize, and correlate videos
   (a) T5.1 Clean up the poses
   (b) T5.2 Normalize the poses
   (c) T5.3 Translate H5 to JSON files
   (d) T5.4 Get the JSONs ready to run the whole process

(e) T5.5 Run the whole code and check results

As we can see in the work plan of the whole project, my contribution is a very small part of it. My main task is T1.4 Align and compress depth data with video segments.

My Gantt Diagram has been the following:

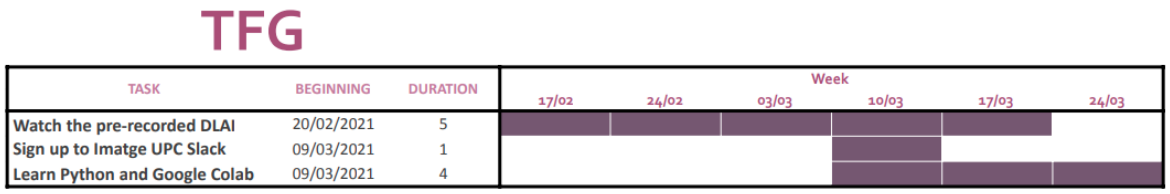


Figure 5: Gantt Diagram (part 1)

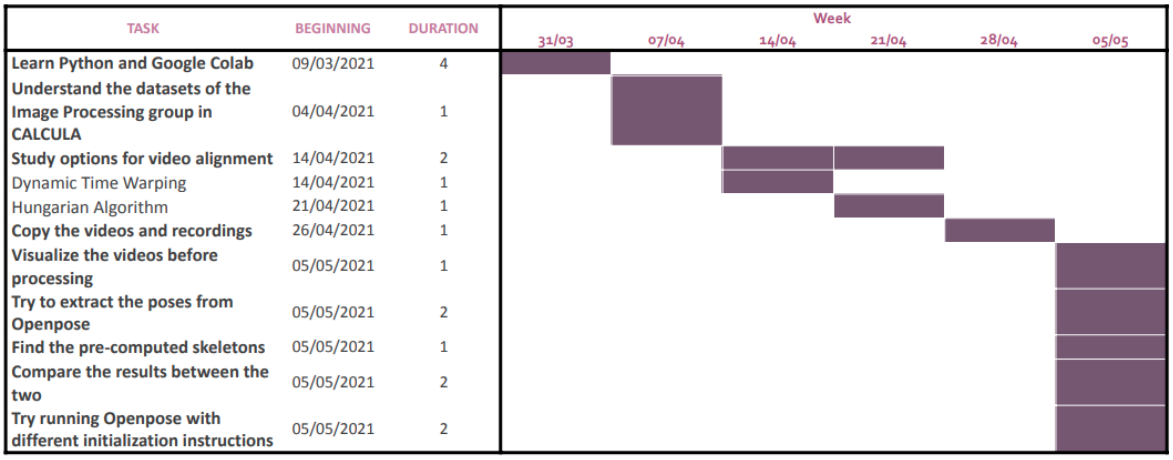


Figure 6: Gantt Diagram (part 2)

| TASK | BEGINNING | DURATION |
|---|---|---|
| Try to extract the poses from Openpose | 05/05/2021 | 2 |
| Find the pre-computed skeletons | 05/05/2021 | 1 |
| Compare the results between the two | 05/05/2021 | 2 |
| Try running Openpose with different initialization instructions | 05/05/2021 | 2 |
| Understand the json file properties | 12/05/2021 | 1 |
| Understand the jsons generated by Openpose | 12/05/2021 | 1 |
| Watch the video with the skeletons on top | 12/05/2021 | 1 |
| Check if the frame rate for both cameras is the same | 20/05/2021 | 1 |
| Try to access Calcula from Colab | 26/05/2021 | 1 |
| Reach Jordi and understand his code | 26/05/2021 | 3 |
| Adapt the frame rates | 26/05/2021 | 1 |
| Downsample de json files | 26/05/2021 | 2 |
| Remove points from skeleton and check output | 26/05/2021 | 2 |
| Plot the skeletons from the json files | 26/05/2021 | 2 |
| Change the rate of the poses in the output json file | 03/06/2021 | 2 |
| Run the auto-correlation on one video and extract conclusions | 03/06/2021 | 1 |
| Reach Laia and understand her code | 20/06/2021 | 7 |


Figure 7: Gantt Diagram (part 3)

| TASK | BEGINNING | DURATION |
|---|---|---|
| Reach Laia and understand her code | 20/06/2021 | 7 |
| Study other students' code and slides | 01/07/2021 | 3 |
| Explore the potential of MediaPipe | 01/07/2021 | 2 |
| Clean up the poses | 30/07/2021 | 7 |

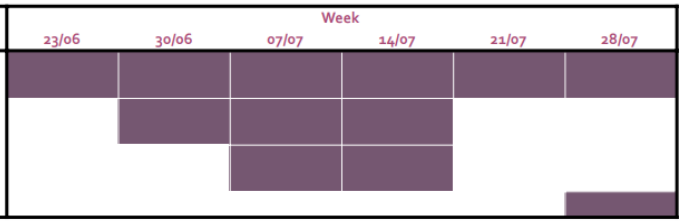


Figure 8: Gantt Diagram (part 4)

| TASK | BEGINNING | DURATION |
|---|---|---|
| Clean up the poses | 30/07/2021 | 7 |
| Normalize the poses | 03/09/2021 | 3 |
| Translate H5 to json files | 03/09/2021 | 4 |
| Reach the same results as Jordi | 10/09/2021 | 1 |

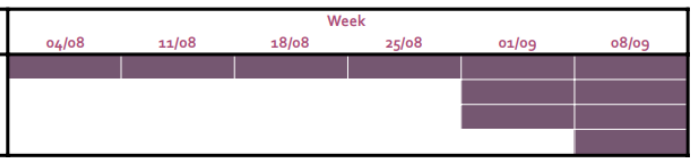


Figure 9: Gantt Diagram (part 5)

| TASK | BEGINNING | DURATION |
|---|---|---|
| Normalize the poses | 03/09/2021 | 3 |
| Translate H5 to json files | 03/09/2021 | 4 |
| Reach the same results as Jordi | 10/09/2021 | 1 |
| Get the jsons ready to try the whole process | 16/09/2021 | 1 |
| Run the whole code and check results | 16/09/2021 | 2 |
| Evaluate results | 30/09/2021 | 1 |
| Modify points of the skeletons and see diferences | 30/09/2021 | 1 |
| Decide the best combination | 30/09/2021 | 1 |
| Conclude the project | 07/10/2021 | 1 |
| Upload the project | 11/10/2021 | 1 |
| Present the project | 23/10/2021 | 1 |

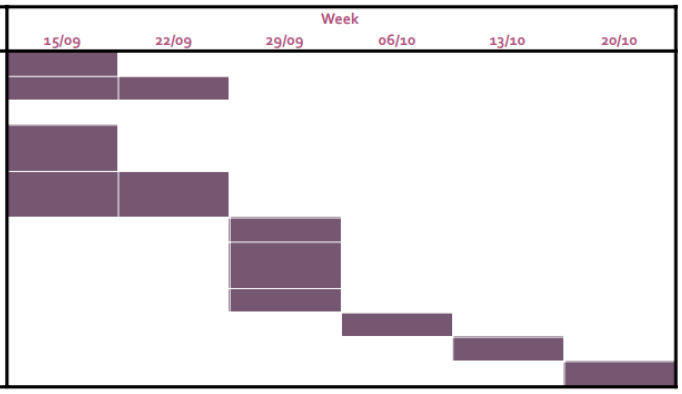


Figure 10: Gantt Diagram (part 6)

## 1.5 Incidents

At the beginning of the project we expected that the submission would take place in June. Due to various problems, we decided that it would be best to extend the delivery to October.

The main root of these problems was the installation and knowledge of the server CALCULA, to which I must connect every time I want to work on the project, since all the datasets that I use and all the tools and programs I run are there. CALCULA is a hard server to connect to due to the lack of documentation and, especially at the beginning, I had a hard time locating myself.

After these unexpected problems with CALCULA, I had to learn to program in Python which, as easy as it could sound since I know other programming languages, was not, because the coding environment (image and video processing) is not straightforward. Also, I had never programmed in Python before because there is not a subject on ETSETB's study plan about this programming language.

Of course, time always plays a vital role in any type of project, and in this project, it was not different. Between outside-school-work and the other subjects at the university, it was difficult for me to find the time to really get serious about the project, and also to coordinate with the tutor and the rest of the team that works in this project. A couple of PhD students and a Master student communicated with me since they work on the same project, doing different functions, and it was also difficult to coordinate the schedules of the four of us.

Finally, we decided that the best thing was to deliver and present in October in order to reach the objectives of my TFG.

# 2 State of the art

## 2.1 UCF

The alignment of videos recorded from different cameras has already been treated by The University of Central Florida, Orlando[7]. Their objective was to identify the egocentric camera holder in the top-view video, to identify the humans visible in the content of the egocentric video, within the content of the top-view video and to temporally align the two videos.

They proposed a unified framework to jointly solve all three problems. They also evaluated the efficacy of the proposed approach on a publicly available dataset containing a variety of videos recorded in different scenarios.

Their experiments show that solving these problems jointly improves the performance in each individual task, as the knowledge about each task can assist in solving the other two.

## 2.2 UV

The Universidad de Vigo[8], has studied this problem too, and it has also been addressed from the perspective of sign language detection in videos recorded from misaligned cameras. In their case, both cameras (Nikon and Kinect) are both frontal, as in the How2Sign case. The Kinect's internal alignment between its RGB and Depth sensor is not perfect, because the depth frame construction is not periodic and sometimes it misses some frames.

The main problem they found is the alignment with the Nikon, with which they record at a different frame rate, and that does not either shoot or stop at the same time, creating sometimes even a whole second of delay. Precisely for this reason, their way of attacking the problem has been to use DTW. Once they adjust beginning and end, the warping is very local (it is almost a perfect diagonal, except for when they lose a frame).

They also tested the cross-correlation in two ways: with the RGB sequences co-registered between Nikon and KinectRGB and with the KP sequences extracted with Openpose also in RGB. The latter worked better because the co-registration was not well tuned.

# 3 Methodology

## 3.1 Development

Before explaining the algorithms used, we introduce some of the framework which information is needed to understand the final code.

### 3.1.1 CALCULA

CALCULA is a computational service of a set of UPC research departments, including the GPI, managed through a queuing and load balancing system.

Thanks to this server, it is possible to access very large datasets, programs that require a lot of gpu, and data that is stored there for the use of students.

CALCULA has helped me to be able to access the videos that I have been treating, to be able to execute OpenPose on them and to be able to do all the corresponding processing on them.

### 3.1.2 Python - Jupyter

The programming language that we have used for the project has been Python, specifically from the Jupyter interface.

I accessed Jupyter from CALCULA, and had the possibility to view each change or modification applied to the videos, since it is not possible to do this from the computer terminal.

It has also been very useful since I have used some of the code of other people involved in the project and they were also in Python.

What I have dealt with most in the project, have been JSON files, since they are the ones generated by OpenPose when it analyzes the positions of the joints of each frame.

### 3.1.3 Dataset

These are the following characteristics of each one of the three videos, which should be known to understand the future changes about them:

| | Recording | Video 1 | Video 2 |
|---|---|---|---|
| Duration | 4:03 | 1:12 | 1:50 |
| Frame rate | 30 | 50 | 50 |
| Total number of frames | 7290 | 3625 | 5521 |

Table 1: Characteristics of each video

## 3.2 2D Pose extraction

OpenPose [9] [10] [11] [12] [13] is a real-time multi-person system to jointly detect human body, hand, facial, and foot KPs (in total 135 KPs) on single images.

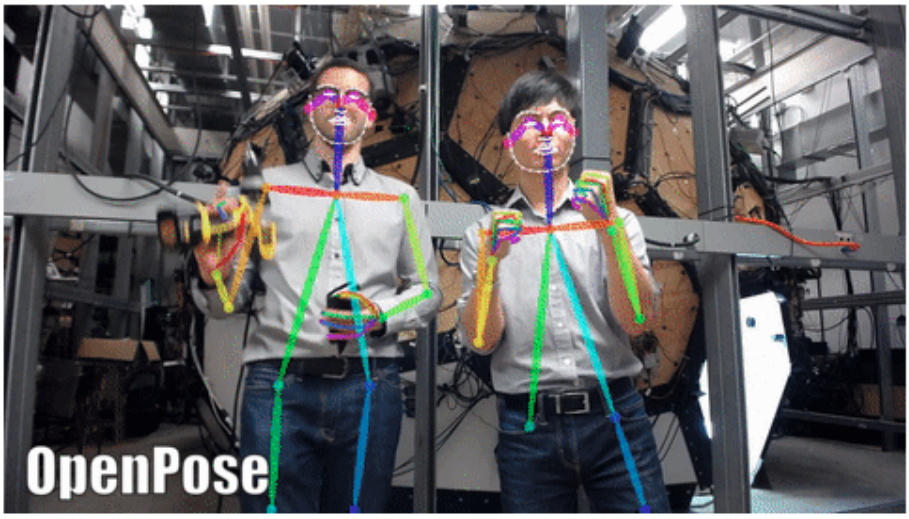


Figure 11: OpenPose's joints detection

OpenPose has different implementations, depending on the initialization of the program. the one I have used is the following:

1. D real-time multi-person KP detection: 15, 18 or 25-KP body/foot KP estimation, including 6 foot KPs. Runtime invariant to the number of detected people.
2. 2x21-KP hand KP estimation. Runtime depends on the number of detected people.
3. 70-KP face KP estimation. Runtime depends on the number of detected people.

OpenPose can work with variants depending on the input and the desired output, but I have made it work this way: Input videos: long recording, 2 short videos. On the running command, asking OpenPose to detect face, pose (torso) and both hands. Output: video with the skeletons of the translator, created by the points of all of the joints OpenPose has detected, and joined by a line following the body shape.

OpenPose generates a JSON file for each frame of the input video with the corresponding joints on the selected positions when initializing the program. It is also possible to visualize the whole video with the joints and lines of the translator's body.

If we visualize a JSON file after having processed it with OpenPose, the visualization is the following:

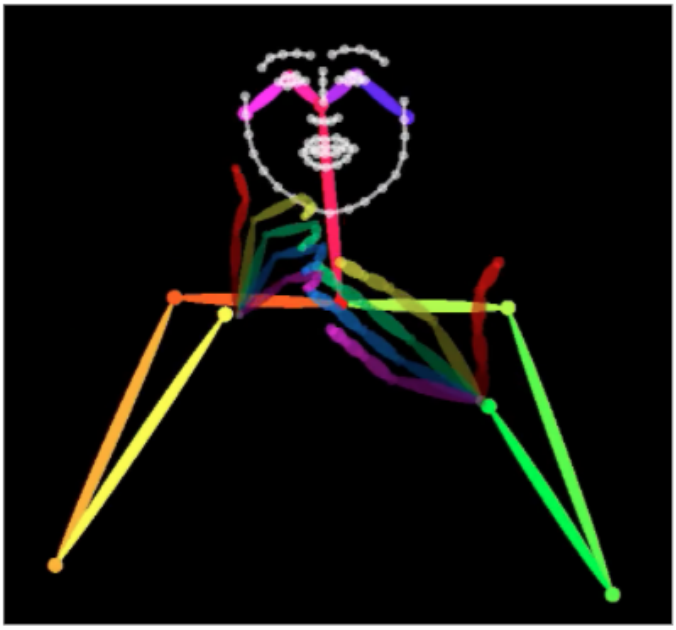

Figure 12: Frame from OpenPose after running it on the recording

To be visually clear, the KPs are the following:

**Face KPs**

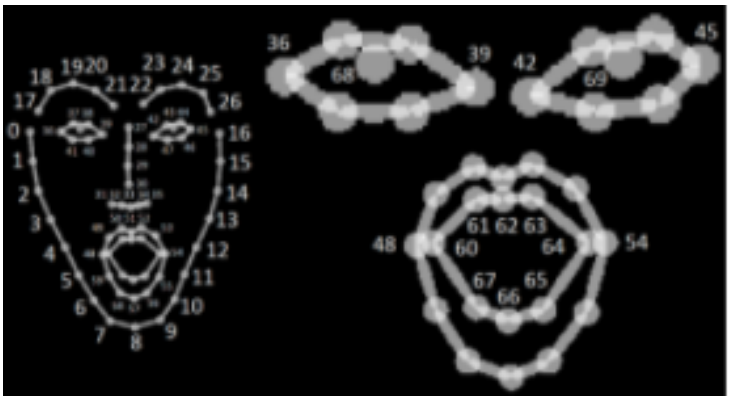


Figure 13: Pose KPs

**Pose KPs**

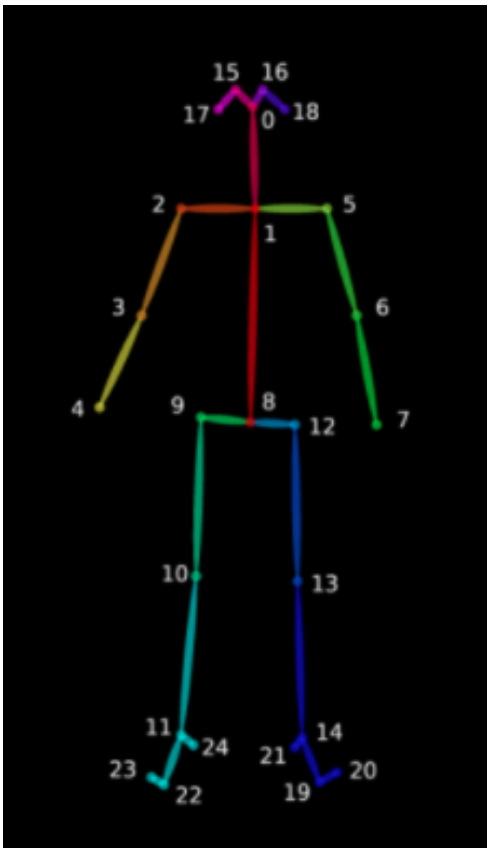


Figure 14: Pose KPs representation

### Hands KPs

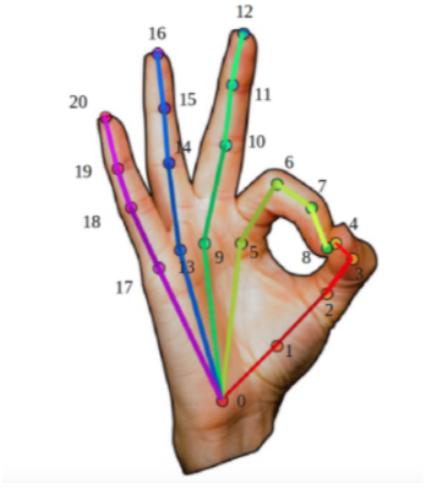


Figure 15: Output KP example from the right hand

A solution based on DTW has been explored, but we wanted to try a simpler solution,

since the duration of the two sequences is the same in both cameras, so we do not see why we should allow warping.

In the following sections, I will explain each step of my project, without getting into detail about the code.

The first step is to process both the *recording* and the *videos* through OpenPose. OpenPose separates each one of the videos in frames, it extracts the key-points (KPs) of the joints of each frame and saves them in JSON files, in the form of [x, y, c] for each KP.

So, for example, since each hand has 21 KPs, for one single frame there would be 63 KP for one hand, 21 for the X axis, 21 for the Y axis and another 21 for the C measure, which is the confidence of OpenPose for each joint.

The KPs that we are interested in are the 2D KPs of the face, the torso and both hands. This makes a total of 25*3 for the face, + 70*3 for the torso and 21*3*2 for the hands, equaling 411 KPs.

## 3.3 Pose smoothing and normalization

After extracting the KPs of every single frame from the 3 videos, we clean and normalize said KPs. We do this because, in the end, OpenPose is not perfect and it sometimes fails detecting the coordinates of each joint.

We apply the normalization[14], which measures the maximum and a minimum coordinates to scale the poses with different sizes in the same value range. Latter, the preprocessing calculates the mean of each joint, removes the poses that have a very different value from the mean, and finally interpolates its value with the neighbouring frames in time.

## 3.4 Video alignment with cross-correlation

### 3.4.1 Intra-camera

After running OpenPose on the videos and filtering them, we compute the cross-correlation of the *video* segment to locate within the *recording*.

This operation is based on the Python library *spicy* [15], specifically the function *signal*. This function is capable of calculating the cross correlation between two N-dimensional signals.

In this case, we used video 1, since it is the shortest and made tests and experiments lighter in execution time.

Firstly, we store each set of KP (pose, face, left hand and right hand) separately, since we will use a different set for each experiment. Then, we calculate the autocorrelation between the frames, for each body part and for all of them together.

In order to assess the robustness of the cross-correlation for the alignment task, we experimented by adding random noise to the value of the KPs. From here we assessed the cases

in which it is negligible, the KP that are most affected, and how the correlation worsens as the noise increases.

### 3.4.2 Inter-camera

In this part we used the three videos, the *recording* from the RGB camera, the *video1* and the *video2* from the RGB-D camera.

In this case, we also needed to and reduce the frame rate of the recording and the videos. This is done to reduce computation time and homogenize the frame rate for the three videos. The final characteristics of the videos are the following:

| | Recording | Video 1 | Video 2 |
|---|---|---|---|
| Duration | 4:03 | 1:12 | 1:50 |
| Frame rate | 10 | 10 | 10 |
| Total frame | 2440 | 726 | 1104 |

Table 2: New characteristics of each video

Now that we have adjusted the frame rates, we compute the cross-correlation [16] between each of the videos with the recording align them.

To check and validate the accuracy, different experiments have been done, changing the input KP before the correlation calculation to compare different results.

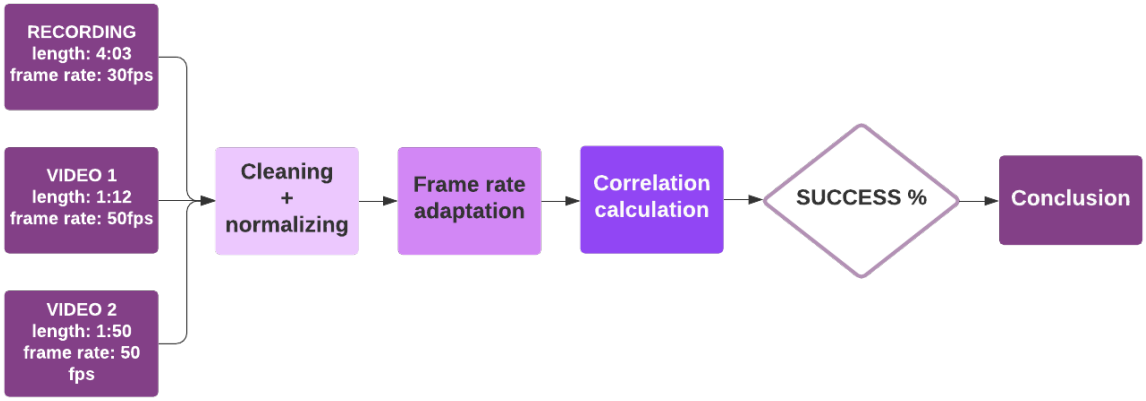


Figure 16: Block diagram of the project methodology

The normalization has been done using three different methods: a Gaussian normalization, a linear normalization based on a straight line (min-max normalization), and a normalization using a projective transformation matrix.

# 4 Experiments and results

## 4.1 Experiments

In this part of the project, all of the experiments performed will be exposed, with every keypoint combination, in the order they have been carried out and the results that have been obtained from each iteration.

The metrics used for analyzing the success of the whole project, is accuracy. The experiment is executed many times, the number of hits is counted, and finally a relationship between correct guess and execution is calculated in percentage.

A hit is a correct detection of the crossed-correlation, meaning the output frame sequence from the *recording* matches the input sequence of frames of the *video*.

The higher the accuracy percentage is, the better the code is working.

### 4.1.1 PART I: Alignment of video clips from the same camera

This first part, both the part that contains noise and the part that does not, has been very easy to evaluate, since the frame in which the autocorrelation was applied was known, and the way to check if it was the correct one was simply by looking if the output frame number, that is, of maximum correlation, was the input frame number.

In this part of the project, it has been decided to evaluate the cross-correlation with the accuracy measure, we have to repeat with N random segments and averaging the results.

The accuracy is a % relationship between the times the correlation is correct over the total times the correlation is calculated.

**Experiment 1: Comparing results between different sets of KPs**

**All KPs:**

The result is perfect, as expected. When plotting the cross-correlation there is a clear peak at the initial timestamp of the segment.

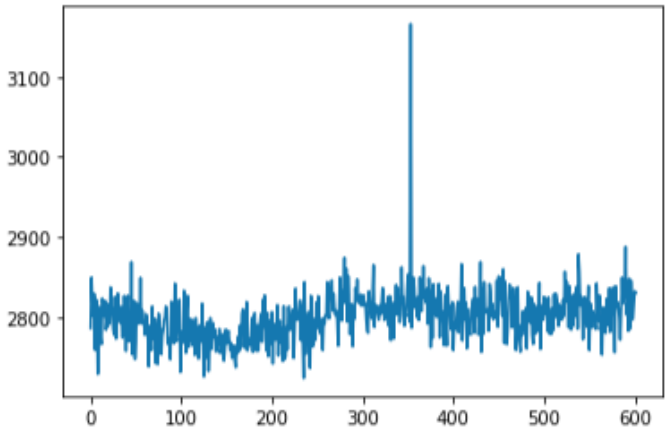


Figure 17: Perfect result with all KPs and no noise

In this example, the start point of the segment is the frame 253, which is easily found and has the highest cross-correlation value.

**Pose KPs only**

After having considered all available KPs, we study the impact of each body part in the alignment. We first start by considering the body KPs only.

The results are no longer perfect. Depending on the clip, the accuracy can vary between 11 and 33 percent, providing a mean of 22% accuracy, depending simply on the signs that the translator performs. In the frames in which the subject is still, either because it is not translating anything at that moment, or because its arms are very parallel, the cross-correlation fails.

On the other hand, in the frames in which he is clearly reproducing signs with his hands, the cross-correlation usually gets it right. This is due to the movement of the elbows and shoulders, which is much more distinctive than when the subject is at rest.

Correct autocorrelation:

Figure 18: Correct pose autocorrelation calculation

Missed autocorrelation:

Figure 19: Wrong pose autocorrelation calculation

### Face KPs only

Regarding to the face KPs, the result is not only not good, it is the worst possible case. After 5000 iterations, the only accuracy obtained has been 0%. As much as the interpreters may express themselves with facial expressions, they are too similar to each other, and the autocorrelation never works.

### Hands KPs only

Finally, as for the hand KP, the result does get better. It is never 100% accurate, but the values are much higher than the pose KP, going from 53% up to 70%, resulting in a mean of 61.5%.

This actually makes a lot of sense, as the hand KP are the most representative ones for sign language detection, the ones that change the most, and therefore the easiest to detect and pair.

The biggest cause of the correlation failing are the frames where the hands are really close together. Since Openpose is not perfect, the KP get mixed up and the correlation fails.

Missed autocorrelation:

Figure 20: Wrong hand autocorrelation calculation

### Experiment 2: Robustness to noise

The impact of noisy pose detecions was assessed in the case of all body KPs.

The KPs extracted from the frames are provided by OpenPose in a range between 0 and 1 and with many representative decimals, so we introduced a Gaussian random noise of

random generated between $10^{-10}$ and 0, and to gradually increase a power of 10 for each experiment, to see how the correlation worsens.

The number of tests to calculate the accuracy has been 5000 attempts, in which a random frame is chosen, which is be the initial frame of the sub-sequence, a sub-sequence made of 20 frames, and to this sequence is added, individually to all the KP of each frame, the said Gaussian noise. The noise added to each KP is the same between KP, but different between frames.

| | $10^{-10}$ | $10^{-9}$ | $10^{-8}$ | $10^{-7}$ | $10^{-6}$ | $10^{-5}$ | $10^{-4}$ |
|---|---|---|---|---|---|---|---|
| Accuracy (%) | 55 | 49.9 | 48 | 46.2 | 31.8 | 4.2 | 0 |

Table 3: Accuracy-noise ratio

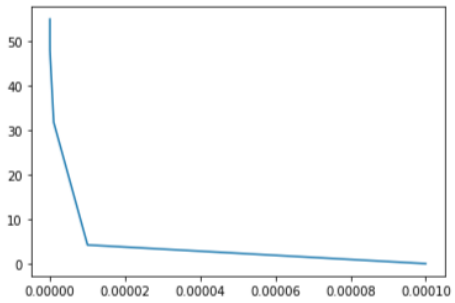


Figure 21: Plot of the robustness to noise

### 4.1.2 PART II: Alignment of video clips from the different cameras

In this second part, the videos were processed, cleaned and normalized with three different methods, the frame rate was reduced to be the same, and finally the cross-correlation was calculated.

Gaussian normalization:

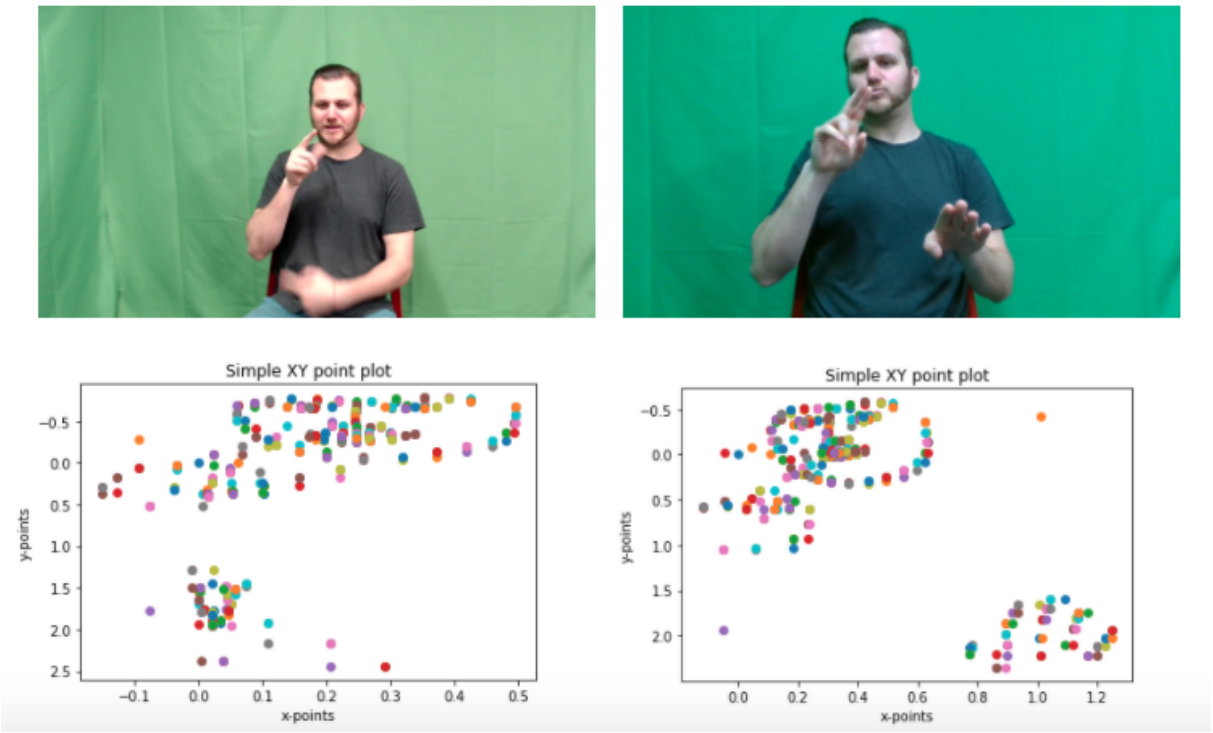


Figure 22: Frame and plot for Gaussian correlation result

In this particular case, comparing the frames and the plot, we could believe that the cross-correlation is close to successfully work, but when we check the graph, we can see that it is just luck.

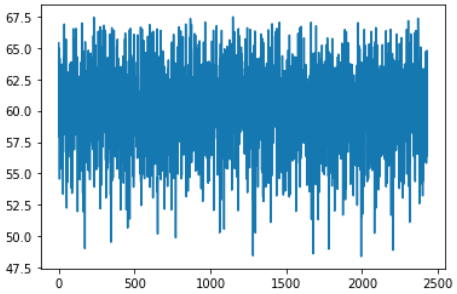

Figure 23: Graph for Gaussian correlation result

Linear normalization:

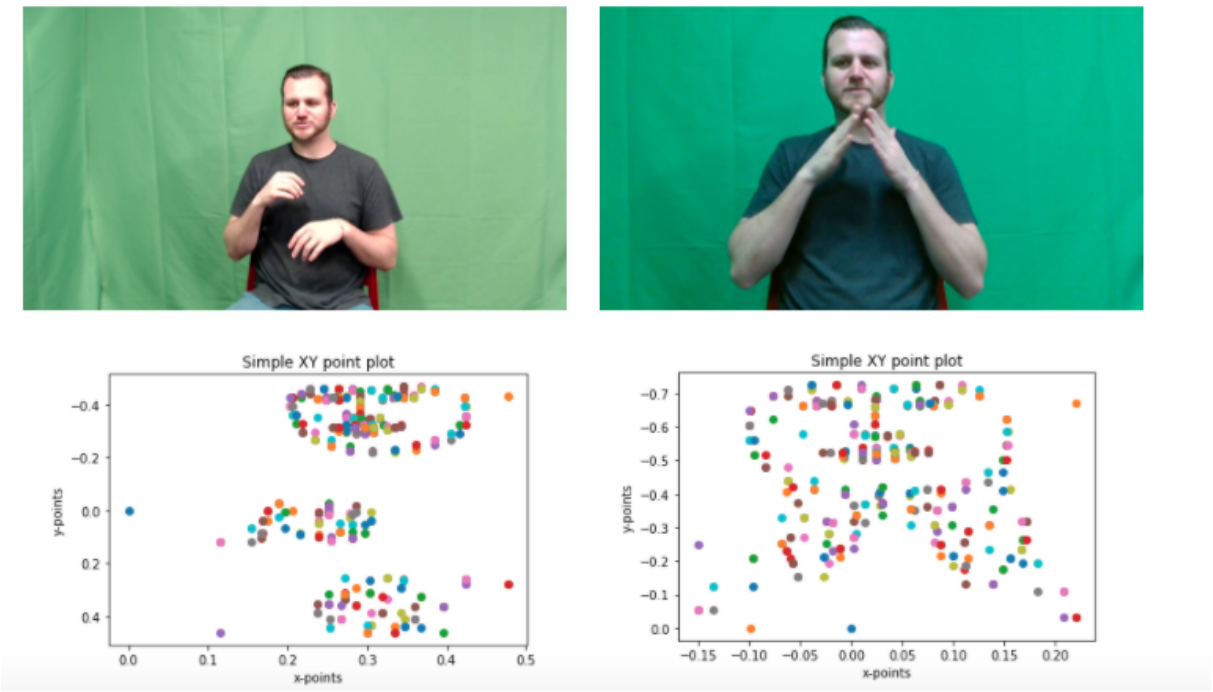


Figure 24: Frame and plot for linear correlation result

Matrix case: This matrix has been calculated from the coordinates of the points 0, 2 and 5, represented in Figure 16, of the KP of the *video* pose, which represent both shoulders and the head. Those three pairs of coordinates are the pairs of points x1,y1, x2,y2 y x3,y3 in the following matrix.

$$\begin{pmatrix} a1 & a3 & a5 \\ a2 & a4 & a6 \end{pmatrix} * \begin{pmatrix} x1 & x2 & x3 \\ y1 & y2 & y3 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} x4 & x5 & x6 \\ y4 & y5 & y6 \end{pmatrix}$$

For the pair of points x4,y4, x5,y5 and x6,y6 the same KP of the pose have been chosen, but from the frame corresponding to the *recording*. This frame has been approximated visually and "cheating", since the final objective is to relate the frames of the *videos* with those of the *recording* without previously knowing where they align.

Finally, matrix A is obtained by clearing the equation and isolating it. This is the matrix used in the normalization, which multiplies all the points of the videos.

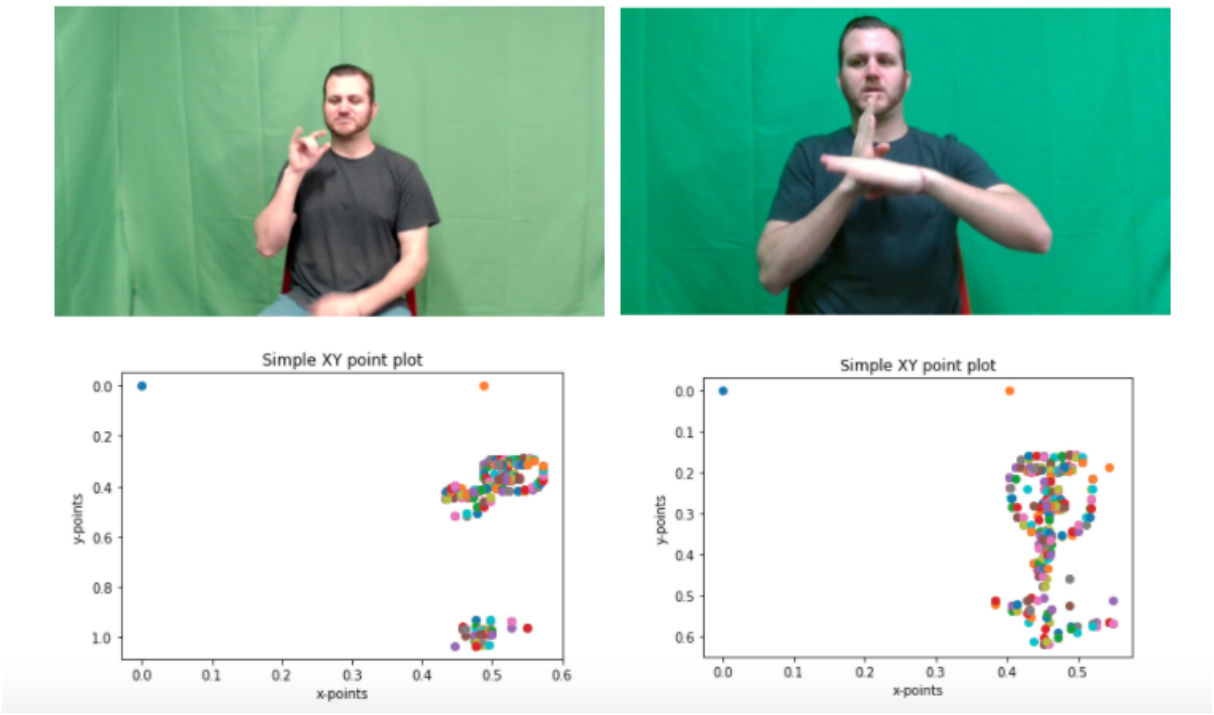


Figure 25: Frame and plot for matrix correlation result

As we can see, the results are not satisfactory. It does not make sense the fact that the highest correlation for the frame set is the one with this beginning.

In this case, it has also been tested all the time with all the KP except those of the face, since, as we have seen in part 1, it is the best of the combinations.

# 5 Conclusions

In the first part of the project, we tested the cross-correlation in a dummy setup of a long sequence of filtered and normalized 2D body poses, which loses accuracy when noise is introduced.

When we used sequence from different cameras, we observed that cross-correlation may work when the poses to be aligned are formed by clearly differentiated points. In this case, given the fact that OpenPose is not perfect, there is noise in the frames of all videos. This entails that, at the time of alignment, the points are not where they should be, and since they all have very similar values between frames the results are not reliable.

We would still need a method to normalize the points between them so that in most frames. We should also find a way to take advantage of the confidence level provided by Openpose for each pair of coordinates of the KP, since some of them are very low, and we could discard those measurements.

# 6 Budget

This project has been developed using the resources provided by the GPI of UPC thus, the only costs of this project, comes from the salary of the executors and the time spent in it.

I consider that my position is the equivalent of a junior engineer, while the position of my professor supervisor, Xavi Giro, as well as the one of the Phd student supervising me, Laia Tarres, corresponds to a senior engineer.

I will consider that the total duration of the project was 35 weeks, as depicted in the Gantt diagram in figures 5 to 10.

| | Amount | Cost (€/h) | Time (h/week) | Weeks | Total |
|---|---|---|---|---|---|
| Junior Engineer | 1 | 10 | 25 | 35 | 8750 |
| Senior Engineer | 2 | 20 | 2 | 35 | 2800 |
| Google Cloud computation | 1 | 0,95 | 4 | 35 | 133 |
| | | | | Total | 11683 |

Table 4: Budget

# 7 Future work

Regarding this project, I would like to continue working with this team in the master's degree, either as a subject in Introduction to research or in the Final Master Thesis.

Taking into account that the results of the correlation calculations after applying different normalizations are not good, I think that the cross-correlation method to solve problems such as misalignment between cameras is not correct.

I think that these normalizations should be reviewed in detail and probably modified so that they are not a small section of the code, but are deeply based on them. Also the possibility of using 3D coordinates should be explored to see if the results improve.

I consider this to be a very interesting project, which touches on topics that I like a lot, such as video and image processing or machine and deep learning, and I also consider its purpose to be very beautiful. Nowadays everything is all bad news and I would love to be a long-term participant in a project that helps a discriminated minority to better integrate into society.

# References

[1] World Health Organization. World report on hearing.

[2] Grup de Processat d'Imatge de la UPC. Sign language recognition, translation and production.

[3] Grup de Processat d'Imatge de la UPC. How2sign-dataset, github.

[4] Amanda Duarte; Shruti Palaskar; Lucas Ventura; Deepti Ghadiyaram; Kenneth DeHaan; Florian Metze; Jordi Torres; Xavier Giro i Nieto. How2sign-dataset.

[5] Ozan Caglayan. How2-dataset, github.

[6] Ramon Sanabri;a Ozan Caglayan; Shruti Palaskar; Desmond Elliott; Loïc Barrault; Lucia Specia; Florian Metze. How2: A large-scale dataset for multimodal language understanding.

[7] Shervin Ardeshir; Ali Borji. Integrating egocentric videos in top-view surveillance videos: Joint identification and temporal alignment.

[8] Laura Docío-Fernández; José Luis Alba-Castro; Soledad Torres-Guijarro; Eduardo Rodríguez-Banga; Manuel Rey-Area; Ania Pérez-P. A multi-source database for spanish sign language recognition.

[9] OpenPose team. openpose-dataset, github.

[10] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[11] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR*, 2017.

[12] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[13] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, 2016.

[14] Jan Zelinka; Jakub Kanis. Neural sign language synthesis: Words are our glosses.

[15] SciPy.org. Scipy developers.

[16] Jordi Lopez Serrano. Introduction to research: Video alignment.