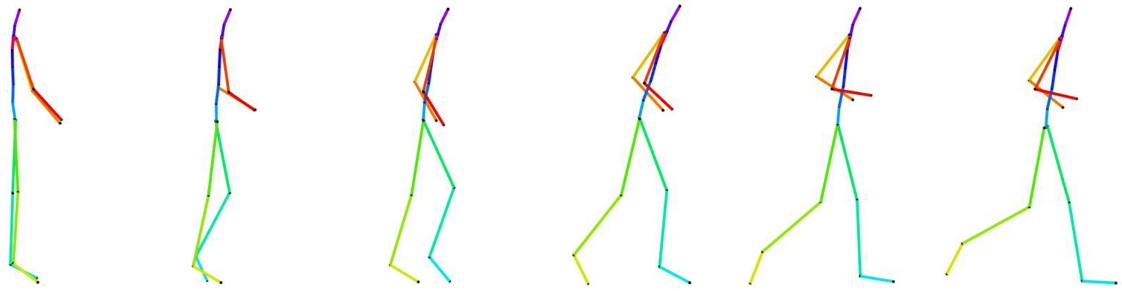
Final Degree Project

Grau en Enginyeria en Tecnologies Industrials (GETI)

Human Motion Dataset in the Wild (MAT)

REPORT

June, 2019



Author: Gabriel Coll Ribes

Director: Francesc Moreno Noguer

Codirector: Mariella Dimiccoli

Rapporteur: Maria Alberich Carramiñana

Escola Tècnica Superior d'Enginyeria Industrial de Barcelona

Summary

The popularity of wearable cameras is steadily increasing, both for entertainment and productivity purposes. Understanding the footage and inferring the body pose of the camera wearer can be of great importance in many fields, like medicine or robotics. It could help in monitoring rehabilitating patients at home from the hospital, in determining the acts of a law enforcement agent from the body camera or, in robotics, simplifying imitation learning based on video input or robot to worker coordination by estimating the posture of the operator.

In this project, we aim to build a human motion dataset acquired indoors and outdoors using a GoPro and MVN Awinda, a movement tracking system based on inertial sensors that provide the 3D human pose. Next, the dataset has been used to train a Deep Neural Network to classify a sequence of frames in the task that it's being performed (walking, running, ...). Finally, a model for estimating the 3D pose of the camera wearer at each frame is proposed based on the same structure than the first network.

The dataset is made of 300,000 frames, captured from seven different people, each one performing 5-6 tasks in different scenarios, both indoors and outdoors. Each sequence of video frames has a synchronized sequence of 3D poses associated to it. Every 3D pose is composed of 23 segments.

The vast majority of the code created and used in this project can be found in the GitHub repository for the project: https://github.com/BielColl/Human-Motion-Dataset-in-the-Wild-MAT-. Due to its size, the dataset built is not posted.

Contents

List of Figures

List of Tables

Preface

Project's origins

I got in touch with the *Insitut of Robòtica i Informàtica Industrial* (IRII) in September 2018, in search of a *Final Degree Project* in the field of robotics and computer vision, fields that had intrigued me for a while then. From the options that were given to me, one interested me especially. That one, the one I'm exposing in this report, included Machine Learning, which I had over-mysticised as highly complex and impossible for me to handle. The possibility to learn about it is the reason I choose this project among the other ones.

This project starts from the goal of building a human motion dataset using the recently acquired motion tracking system by the IRII, a dataset bigger than the one built in this project. However, because of the lack of experience in using this equipment, a set of technical difficulties were needed to be solved, such as video and 3D pose data synchronization or data management. Therefore, this project serves as a starting point from which a larger dataset could be build.

Previous requirements

For recollecting the data, one needs to know how to use efficiently and correctly the MVN Awinda hardware, its software and the GoPro camera. For processing and using the data, programming skills in *Python* and *MATLAB* were needed. Basic theory in Machine Learning was required, being crucial the knowledge about diverse network architectures (Fully Connected Networks, Convolutional Neural Networks and Recurrent Neural Networks) and some deep learning techniques, like Transfer Learning.

1 Introduction

1.1 Objectives

The main objective of this project is to build a human motion dataset using MVN Awinda and a GoPro which could be used for Deep Learning applications. For checking it's viability for such purposes, the dataset will be used to train a neural network to classify sequences of egocentric video frames. If creating such a dataset is plausible, this project could work as a predecessor of a bigger and more ambitious dataset.

1.2 Project scope

The dataset built in this project has been created to test the viability of using the MVN Awinda to create a larger dataset of human motion captures for Machine Learning applications. Therefore, the final data collected is not thought to be published or distributed. Similarly, the machine learning approaches applied in this project are intended to be a proof of the usefulness of the dataset built.

Once the viability of the dataset created is proven, this project can provide the basis for a future project of building a larger and more complete dataset of human motion captures and egocentric images in the wild.

1.3 Problem definition

In order to build the dataset, a protocol will have to be defined in order to work in an efficient way and avoiding as many mistakes as possible. The recording, synchronization and post-processing of the data will have to be clearly defined and fully working.

The built dataset will be used for training a LSTM based network. Several features will be extracted from the frames which will be concatenated in a single feature vector and work as an input for an LSTM. The objective will be to classify a sequence of frames in the task that is being performed of a set of 5 different tasks. This can be a considered a simple problem and the network should be able to obtain good results. Then, an additional model will be presented for future works, where the objective will be to infer the camera wearer 3D pose at each frame of a sequence. The pose will be obtained by classifying the frame in a set of 500 different poses, obtained by clustering all the poses of the dataset.

1.4 State of the art

1.4.1 Predicting human pose from an egocentric video

Third person pose estimation: Estimating the human pose from images in a third person viewpoint have been deeply studied. In these works, the person is entirely visible in the image, mak-

ing the challenge different from the one faced in this project. In first person viewpoint, most of the body is never visible and other parts rarely are (e.g hands, elbows or feet). Therefore, most of these approaches are not applicable for our scenario.

Recent works as DeepPose [1] uses Convolutional Neural Networks (CNN) to obtain the 3D poses from images using a direct regression approach. Others uses CNN to compute confidence maps in the image for each joint, showing the probability to find a certain joint in an areas of the image. In [2] they combined this approach with part affinity fields (PAF's) which takes into account how the joints connect with each other, improving the final results. In one hand, PAF's are not useful for this project, since only a small set of joints will be rarely seen on screen. In the other hand, confidence maps can help to detect if a certain joint is shown on screen or not, giving useful information about the invisible pose of the camera wearer. For example, only certain poses of the arms will end up with the hands in front of the camera.

Egocentric video analysis: Many Deep Neural Networks works have approached video analysis in numerous ways, like combining CNN with Recurrent Neural Networks as used in [3] or processing stacks of video frames with 3D CNN like in [4].

First-person body pose from video: Infering the first-person body pose from an egocentric video is, indeed, challenging, since most of the time the body of the camera wearer is not on screen. Most approaches extract information from the camera motion during the video (obtained extracting a sequence of homographies between adjacent frames) and from the scene itself, using feature extractors based on CNN's. Then, some works use this information to build a regression model to estimate the 3D Pose [5] and others combine it with recurrent neural networks [3], but in both approaches the previous frames and poses are taken into account. A recent approach [6] proposes to predict the 3D pose from egocentric video using imitation learning to learn a control policy for pose prediction. Also, it's combined with physics simulation, which is usually overlook in other works.

While the approach used in [3] provides good results, it relies in interactions of the camera wearer with other persons that are in sight of the camera. Due to this fact, if videos with no interaction with more people are used, the model may fall short in information. Instead, in this project the joints of the camera wearer will be looked for in the frame. This can provide useful information about the overall pose without needing person to person interaction.

1.4.2 Datasets

For all previous approaches based on Machine Learning algorithms, video and human pose datasets are required. It is common for those working in one of these approaches to build their own dataset, as they are able to make it fit their needs, but many others base their work purely in building a more ambitious dataset.

First person video only datasets: First person videos have gained popularity in the recent years, thank to smaller and more affordable devices, and they provide a different insight in people activities in relation to more common third person videos. The number of egocentric videos datasets is considerably smaller than third person ones, but its growing steadily. In the set of available datasets for egocentric videos there are those based in object interaction tasks [7,8,9,10] and others based more in person interaction tasks [3,11].

Egocentric video and human pose datasets: Due to it's affordability, Kinect cameras are vastly used [3,5] for recording the human pose. In [3] they also used a Panoptic Studio dome to record some of the sequences used. Other ways to capture human motion can rely in visual approaches based on passive or active markers. What most of these datasets have in common is that they use a chest-mounted camera instead of a head-mounted one (as will be used in this project), as it provides a more stable image. While this provides good results, it's not well borne out by reality, as lots of egocentric videos are recorded using head-mounted wearable cameras.

2 Equipment

2.1 MVN Awinda

XSens [1] MVN Awinda is an inertial motion capture system that enables the user to record full body human motion. It's based on wireless inertial sensors placed onto the subject of study and connected to a computer using the Awinda Station. All inertial data is processed with the algorithms built in the MVN Awinda software, MVN Analyze.

2.1.1 Straps and inertial sensors

To be able to record, MVN Awinda uses 17 inertial sensors placed in different places of the body (see Table 2.1 [2] and Figure 2.1). The pelvis and extremities sensors are secured using FabriFoam Velcro straps. The shoulders and the sternum ones are placed onto the velcro of the MVN shirt. The hands ones are secured using gloves with velcro, the feet ones can be placed with foot pads or secured using the shoelaces and the head one is secured using a head band. Before placing all the sensors, it's important to turn all them on using the button on the side. When active, the sensor LED starts blinking.

	Sensor name	**Position**
1	Pelvis	Flat on sacrum
2	Sternum	Flat, in the middle of the chest
3	Head	Any comfortable position
4	RightShoulder	Right scapula
5	RightUpperArm	Lateral side above right elbow
6	RightForeArm	Lateral and flat side of the right wrist
7	RightHand	Backside of right hand
8	LeftShoulder	Left scapula
9	LeftUpperArm	Lateral side above left elbow
10	LeftForeArm	Lateral and flat side of the left wrist
11	LeftHand	Backside of left hand
12	RightUpperLeg	External lateral side above right knee
13	RightLowerLeg	Flat on the shin bone (medial surface of the right tibia)
14	RightFoot	Middle of bridge of right foot
15	LeftUpperLeg	External lateral side above left knee
16	LeftLowerLeg	Flat on the shin bone (medial surface of the left tibia)
17	LeftFoot	Middle of bridge of left foot

Table 2.1: List of inertial sensors and their position

[1] https://www.xsens.com/
[2] Information extracted from [12]

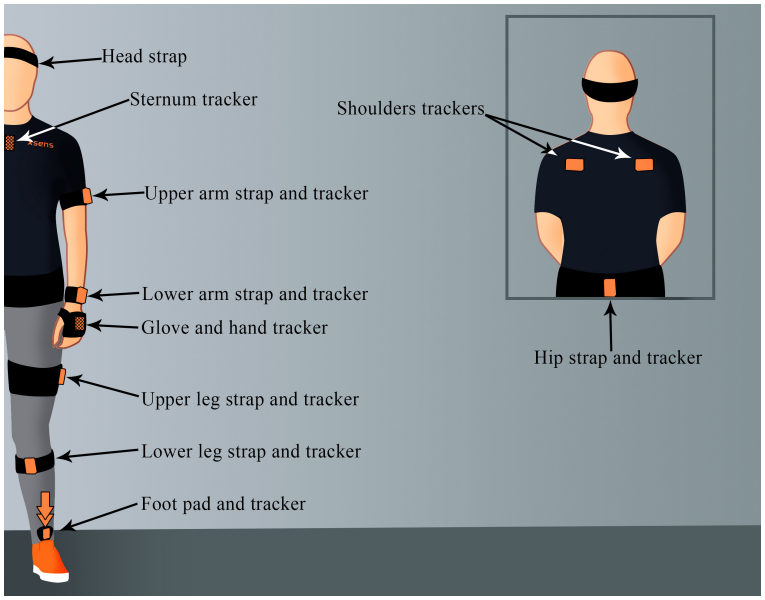


Figure 2.1: Representation of the inertial sensors of MVN Awinda

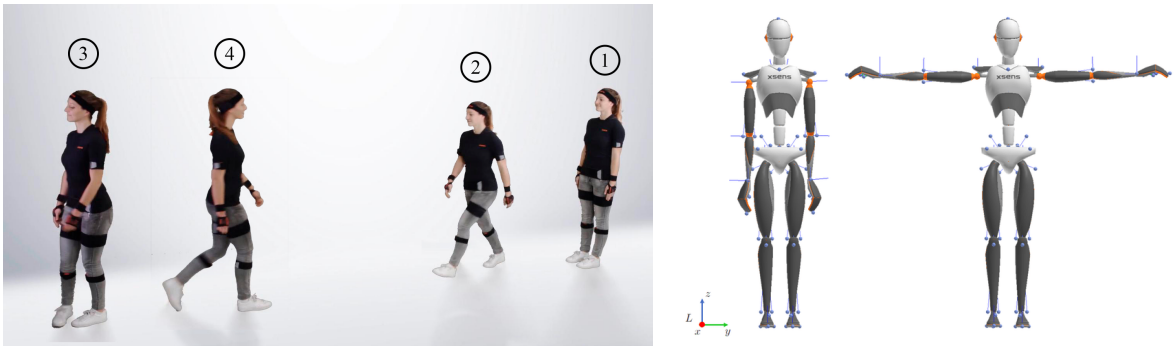
2.1.2 Data acquisition

When all the sensors are placed onto the subject, one must calibrate them with MVN Analyze. With the MVN Awinda Station connected to the computer and MVN Analyze opened, one would need to create a new recording session. The software will require physical measurements of the real subject to be able to estimate the joints position with the inertial data provided. If the goal of the recording is only capture human motion and exact segment length (maximum error of 3-5 cm in certain segments) it's unrequired, one would simply need to measure the height and the foot length (including the shoes if wearing), as the software will estimate the rest. If more accuracy is needed, one can get all the measurements required. The list of all the measurements that one can introduce in MVN Analyze is described in Table 2.2.

Measure	Definition
Body height	Ground to top of head when standing upright
Foot size	Length of feet or length of shoes if wearing shoes
Arm span	Top of right fingers to top of left fingers in T-pose
Ankle height	Ground to distal tip of lateral malleolus
Hip height	Ground to most lateral bony prominence of greater trochanter
Hip width	RIght to left anterior sup. iliac spine
Knee height	Ground to lateral epicondyle on the femoral bone
Shoulder width	Right to left distal tip of acromion (acromial angle)
Shoulder height	Ground to C7 spinal process
Extra sole height	Additional thickness of soles below normal shoe sole height. Use for stilts, platform soles, etc. This measurements is only necessary if specially thick soles are used

Table 2.2: Measurements that one can introduce in MVN Analyze to define the segments length

To calibrate the sensors, the person wearing them will have to follow a predefined sequence of movements. During calibration the subject will stand in a N-Pose, walk a few seconds, turn around 180 degrees, return to the origin, turn around 180 degrees again and remain in the same pose as before (see Figure 2.2a for a visual guideline). N-Pose (see Figure 2.2b)) is an alternative human pose to the more classic T-Pose and its recommended since the N-pose relates closely to a natural standing position.



(a) Calibration steps

(b) N-Pose and T-Pose [13]

Figure 2.2: Calibration information

2.1.3 Data visualization

MVN Analyze allows the user to do a live preview the 3D pose obtained by the inertial data of the sensors. It's important to keep in mind that the 3D pose has not been optimized yet, as it's a fast approximation made for live previewing. Therefore, minimal accuracy errors should be expected to be deleted after processing the data. Bigger accuracy errors should be noticed (e.g. the hands aren't properly touching when clapping), as they would be a warning sign that re-calibration is needed.

Before and after recording, one can check the data in *MVN Analyze* plotting one or more parameters of any segment, joint or sensors. In Figure 2.3 an example of a plotting made in MVN Analyze is shown. The plot is showing the position of the pelvis during a back and forth walking. While the Y and Z coordinates remain relatively constant, the X one increases and decreases with time.

2.1.4 Processing and exporting

As mentioned before, the data obtained with MVN Awinda needs to be processed to obtain more accurate poses. There are three orders of processing: no processing, *Normal Processing* and *HD Processing*. Each one provides better results than the previous one, but takes more time to compute. For reference, a 2 minute long sequence of data is processed in *Normal quality* in 5 min but can take up to 30-60 minutes to be processed in *HD quality*. Exporting unprocessed data is unrecommended.

After processing the data, one can export it to be used in third party software. All recordings are saved as MVN, which is the native file for MVN Analyze and contains all measured inertial

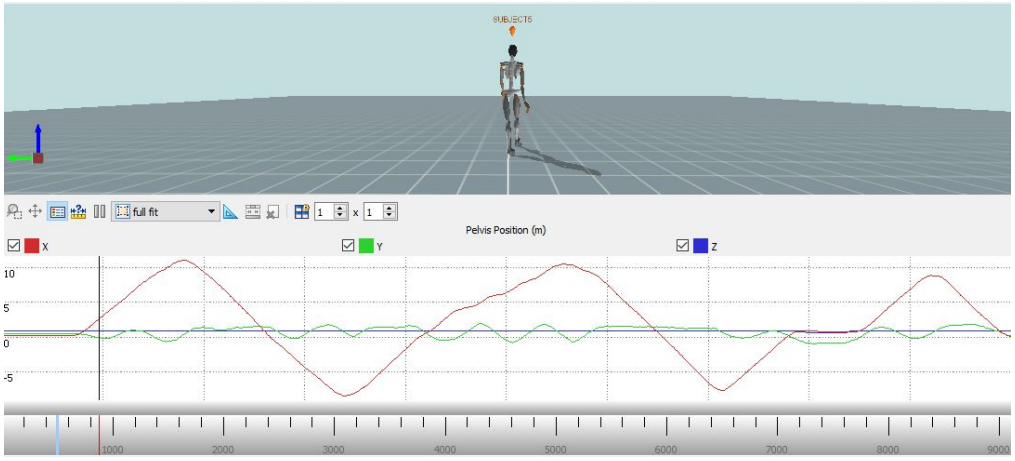


Figure 2.3: Position of the pelvis plotted in MVN Analyze.

sensor data and kinematic data of each segment. MVN files can be exported to C3D, FBX, BVH or MVNX format. For this project, the most proper choice was exporting as MVNX (MVN Open XML format), as it can be opened using Microsoft Excel or MATLAB (in this case using an already made script provided by XSens). In addition, the XML format makes available to write simple libraries to be able to manage the MVNX file from other programming languages.

Both processing and exporting can be done in batches, which saves the time which would be used in opening each file on its own.

2.1.5 Advantages and disadvantages

The power of MVN Awinda against other motion capture based on optical tracking relies on its portability and freedom of movement. The only requirement is the sensors have to be within the range of the receptor connected to the computer. If one uses a laptop, one can move around the receptor and still be able to get reliable data. While it is advisable that the station remains still during the recording, it can be moved to new locations without needing to re-calibrate the sensors.

All data captured is purely recorded at each individual sensor and then processed in MVN Analyze. As this approach is not based on visual marks, the subject joints are unrequired to be fully in sight of any camera. This opens up a extensive variety of possibilities in terms of recording scenarios and activities, like going to other rooms or walking through crowds.

These two advantages open the possibility to easily capture the movements of a subject in the wild, that is in real scenarios both outdoors and indoors.

Regrettably, MVN Awinda have shown severe sensitivity to slight errors during calibration, as minimal movements when the subject had to remain still lead to high accuracy errors in the pose estimation. On top, the dimensional accuracy is unperfect because some segments dimensions remain undefined, even if all the measurements of the participant are entered in MVN Analyze.

2.2 GoPro Camera

2.2.1 Features

The videos were recorded using a GoPro [3] Hero 5 Black camera (see Figure 2.4 [4]). As will be seen in Section 4.1, most of the sequences to be recorded required full freedom of movement of the participants. On top, as the camera will be head mounted and some of the tasks performed included considerable degree of motion (e.g. running), image stabilization was necessary to obtain good quality individual frames, which is provided by the GoPro.

This camera provided a variety of options in terms of frame rates and resolutions, offering 4K at 30 fps, 2.7K at 60 fps, 1440p at 80 fps, 1080p at 120 fps, 960 at 120 fps and 720 at 240 fps. As the MVN data are captured at 60 Hz (i.e. 60 sets of 3D joints position are obtained every second), the video recording was set at 60 fps and 2.7k. In fact, the camera recorded at a frame rate of 59.94 fps. Therefore, to be able to synchronize the data from MVN Awinda and the camera, the videos will have to be post-processed to obtain a video with an exact frame rate of 60 fps. For this project, Apowersoft [5] Video Converter was used for this purposes, although there are many other options available.

Additionally, this camera allows the user set the field of view (FOV) of the recording, from wide (resulting in a fish-eye look) to a linear, which provides undistorted footage. For simplicity purposes, the linear field of view was selected.

Model	GoPro Hero 5 Black
Video resolution	2.7k (1920x1080)
Frame rate	59.94 fps
Field of view	Linear

Figure 2.4: Camera model and settings used.

[3] https://gopro.com
[4] Image from [14]
[5] https://www.apowersoft.es/

3 Data projection onto a video

In order to check the accuracy of MVN Awinda in capturing the movements of a person, a series of sequences were recorded, using a camera and MVN Awinda.

3.1 Camera calibration

Cameras project 3D points from the real world to a 2D plane, generating images. The calibration of a camera is the process of determining the internal parameters of the camera that define how those points are projected and, therefore, how the image is generated. Calibrating the camera is required for tasks like 3D interpretation of images, robot interaction with the world (Hand-eye coordination) or projecting 3D data on real images (augmented reality). The internal parameters to be found are the focal length, the image center and the lens distortion parameters.

3.1.1 Intrinsic and extrinsic parameters of a camera

The model of the projection using only the intrinsic parameters can be seen in Figure 3.1, using the pinhole ideal camera model. The intrinsic matrix that defines this model is composed of the focal length and the image center. The focal length is the distance between the pinhole and the projection plane, as can be seen in Figure 3.1. Ideally, f_x and f_y are equal but, in practice, they may differ due to the image post-processing, lens distortion, flaws in the camera sensors or simply due to calibration error. Different focal lengths result in images with rectangular pixels insted of square ones. The image center (c_x, c_y) is the intersection with the projection plane of a line perpendicular to the plane passing through the pinhole. The units of both image center and focal length are pixels. As an infinite number of pinhole cameras could be build to produce the same image (all with different sizes), it's preferable to express it's parameters with relative units (pixels) than absolute units (millimeters, centimeters, ...).

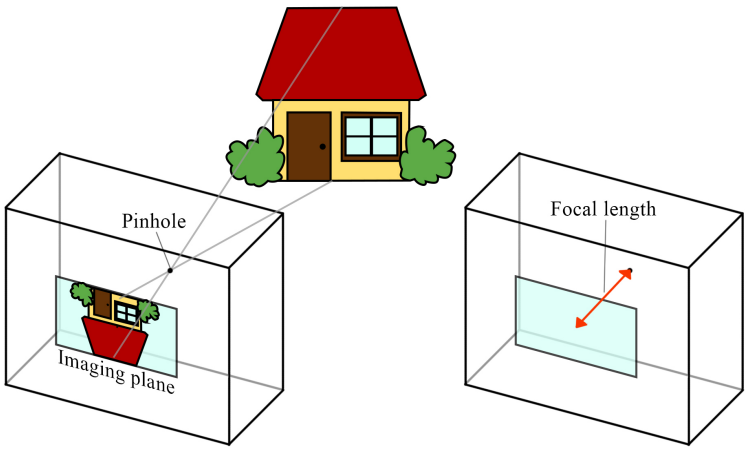


Figure 3.1: Left: Pinhole camera diagram. Right: Focal length diagram.

The intrinsic matrix is build as seen in Equation 3.1. It can decomposed as a sequence of translation and scaling. The translation redefines the coordinates origin to one of the four corners of the image, conventionally the top left or the bottom left ones. The scaling will define the size of the projected objects in the image.

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & c_x \\ 0 & 1 & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{Translation} \times \underbrace{\begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{Scaling} \tag{3.1}$$

What differences real cameras from pinhole cameras is the use of lenses. Due to its shape and positioning, they can introduce distortion to the final image. There are two principal kinds of distortion:

- **Radial distortion:** It's caused by the spherical shape of the lens. This distortion is caused by the reflection of the light while passing through the lens. The center of the image remains undistorted as the light rays enter the lens perpendicular to the surface. The light that enters closer to its edge suffers more refraction and, therefore, suffers more bending. Given C as the image center, P as a point of the undistorted image and P' as the distorted version of P, if there is pure radial distortion, the position of P' is obtained by moving the point P radially, in the direction of CP

  In Figure 3.2 it can be seen how the radial distortion can curve line inwards (barrel distortion) or outwards (pincushion distortion).

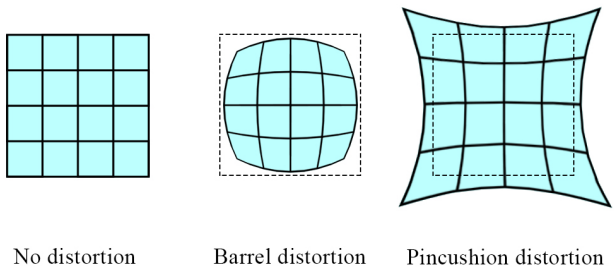


Figure 3.2: Radial distortion

- **Tangential distortion:** It's caused when the lens plane and the camera sensors plane are not parallel (as can be seen in Figure 3.3). This makes the final image look crooked making parts of the imaging plane look closer and the radially opposite part, further away. Given C as the image center, P as a point of the undistorted image and P' as the distorted version of P, if there is pure tangential distortion, the position of P' is obtained by moving the point P tangentially, in the perpendicular direction of CP.

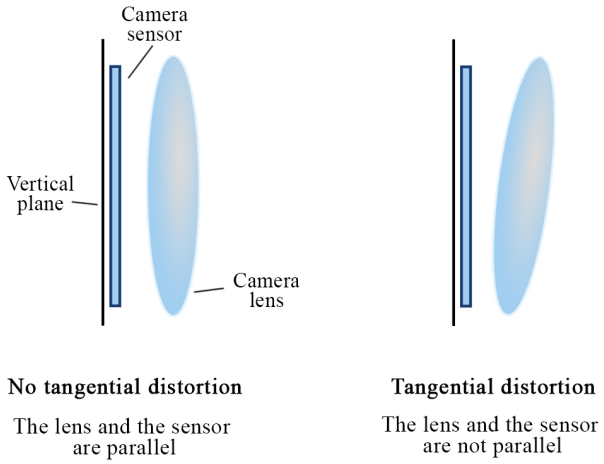


Figure 3.3: Tangential distortion

Lens distortion can be modeled as seen in Equation 3.2 where k_1, k_2 and k_3 are radial distortion coefficients and p_1 and p_2 are tangential distortion coefficients. In Equation 3.2. u and v are the coordinates of the distortion free-image, u_{dist} and v_{dist} are the corresponding image coordinates with distortion applied and $\delta_u(u,v)$ and $\delta_v(u,v)$ are distortion in u and v directions.

$$\begin{aligned} r^2 &= u^2 + v^2 \\ \delta_u(u,v) &= u \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) + 2p_1uv + p_2 \cdot (r^2 + 2u^2) \\ \delta_v(u,v) &= v \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) + 2p_2uv + p_1 \cdot (r^2 + 2v^2) \\ u_{dist} &= u + \delta_u(u,v) \\ v_{dist} &= v + \delta_v(u,v) \end{aligned} \tag{3.2}$$

In addition to the intrinsic parameters, one needs to know the extrinsic parameters, which represent a rigid transformation from the 3D world coordinate system to the 3D camera's coordinate system. The 3D camera's coordinate system has its origin at the camera, the Z-axis pointing towards where the camera is facing and the X and Y axis parallel to the image plane. Therefore, the extrinsic parameters are composed by a rotation matrix and a translation vector that express this coordinate system change and the position and orientation of the camera in the 3D world coordinate system.

3.1.2 Camera calibration with *OpenCV*

To obtain the intrinsic matrix and the distortion coefficients, one needs to perform a camera calibration. For this project, the library *OpenCV* [6] for *Python* was used. To do so, one need to take pictures with the camera of a known pattern. From the several options available, a chessboard was used. Given the number of corners in the chessboard and the length of a square edge,

[6] https://opencv.org/

the *OpenCV* library is able to detect the corners in the picture and to calculate all the intrinsic parameters.

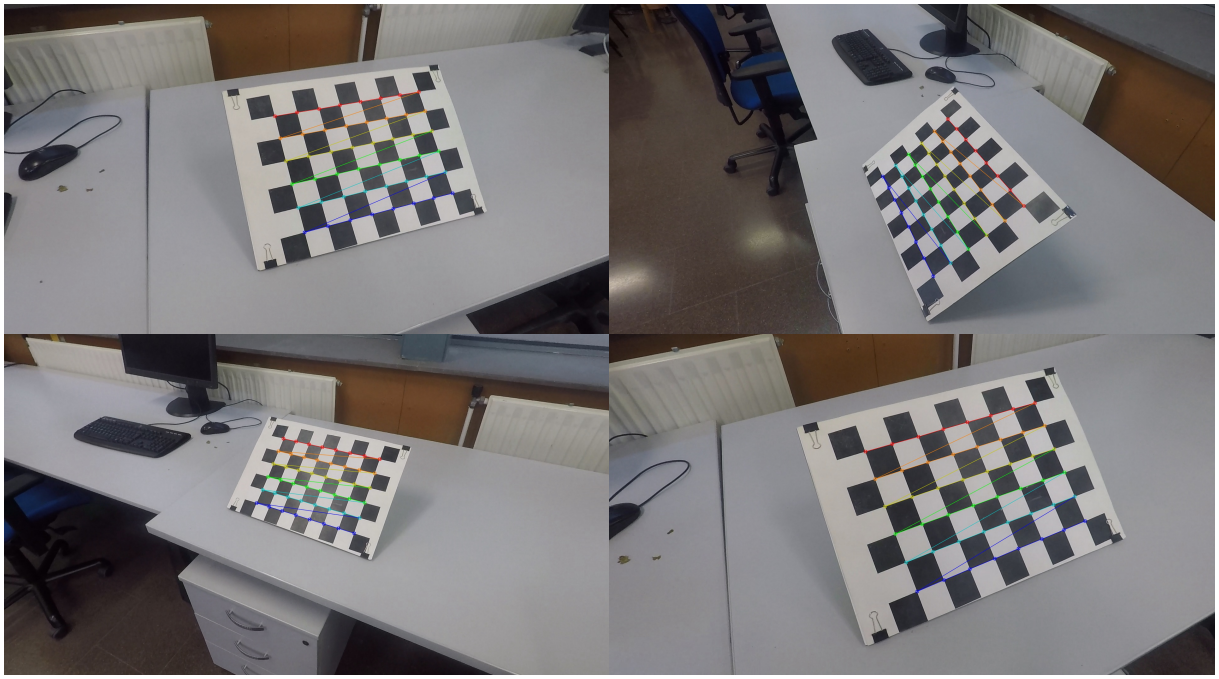


Figure 3.4: Examples of chessboard images used for calibration

Due to errors during calibration and minor physical changes in the camera, the intrinsic parameters of the camera may change between calibrations. After several calibrations, the results for the camera used in this project only seemed to change (more than 5% of variation) when the calibrations were done with more than a week between them. Therefore, the intrinsic parameters used were obtained the closest to the time of the recordings as possible. Generally, the calibration was made the same day as the recordings and rarely, within the same week, but never with larger time gaps.

3.2 Obtaining the camera position and orientation

Camera calibration provides the intrinsic parameters of a camera but, in order to be able to project 3D data in the 3D world coordinate system to an image taken by the camera, one needs to know the extrinsic parameters too. As seen previously, the extrinsic parameters is a coordinate system change from the 3D world coordinate system to the local 3D coordinate system of the camera. The transformation is done as explained in Equation 3.3.

$$\vec{x}^{\,C} = R_{3x3}^{W\to C} \cdot \vec{x}^{\,W} + T_{3x1}^{C} = R_{3x3}^{W\to C} \cdot \left(\vec{x}^{\,W} + T_{3x1}^{W}\right) \tag{3.3}$$

$$\begin{bmatrix} u\cdot s \\ v\cdot s \\ s \end{bmatrix} = A \cdot \vec{x}^{\,C} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}^{C} \tag{3.4}$$

In Equation 3.3 one uses a rotation matrix $R_{3x3}^{W \to C}$ to rotate the axis of the world coordinate system to the camera coordinate system. The columns of this matrix correspond to the axis of the world coordinate system expressed in the one of the camera. This matrix can also be built putting the camera axes expressed in the world coordinate system as the columns and then invert the matrix. The origin change is made adding the translation vector. The vector T_{3x1}^{W} is minus the position of the camera origin expressed in W and the vector T_{3x1}^{C} is minus the position of the camera origin expressed in C. Then, in Equation 3.4, one uses the intrinsic matrix to project the 3D points in the 3D camera coordinate system to the imaging plane. To get the 2D coordinates (u, v) one needs to divide the resulting vector by s, a scaling factor.

To obtain the camera pose in the world coordinate system, one can use a Perspective-n-point (PnP) algorithm. Determining the camera position and orientation from a set of n 3D-to-2D correspondences [7], where $n \geq 4$, is known as the PnP problem. This is a fundamental problem in camera vision, as it has multiple application in robotics and augmented reality. Even though several solutions to this problem have been found, most of them have complexities of order $\mathcal{O}(n^5)$ or even $\mathcal{O}(n^8)$. The Efficient PnP (EPnP) method [15] is an accurate $\mathcal{O}(n)$ solution to the PnP problem. EPnP proposed a new parameterization of the problem based on barycentric coordinates more accurate solution than previous approaches.

Simply put, given an image of a known set of 3D points (their coordinates are known) and the coordinates of those points in the image taken, one can estimate the $R_{3x3}^{W \to C}$ and T_{3x1}^{W} matrices using the EPnP algorithm. For this project, a MATLAB implementation of the algorithm was used.

In this project, MVN Awinda provides the 3D coordinates of all the joints of the subject throughout a sequence of movements and the video-camera provides a sequence of images of the same movements. If the camera were in a third person point of view and was static, one could apply the PnP algorithm to a single frame. Given a frame were a set of the subject joints are visible, one get the coordinates of those joints in the image by manual selection or using a Deep Learning approach as [2]. Then, as the camera orientation and position in the 3D world coordinate system would not change throughout the sequence, the results of applying the EPnP to a single frame could be applied to all the video.

Since the camera is egocentric, its position and orientation expressed in the world coordinate system are not constant, as it moves with the head of the subject. Luckily, MVN Awinda provides the orientation and the position of the head. As the camera is head-mounted, the relative pose of the camera to the head is constant. Therefore, one can define a local coordinate system based on the orientation and position of the head joint. Then, if the 3D points provided by MVN Awinda are expressed in the head local coordinate system and one applies the EPnP algorithm, the camera pose obtained would be relative to the pose of the head. Then, the matrices obtained could be used throughout all the video. A diagram of the different coordinate system used can be seen in Figure 3.5.

[7] A set of n 3D points and their corresponding n projected points in the camera image

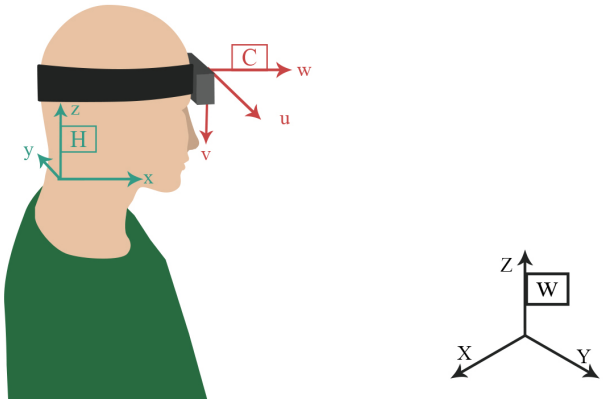


Figure 3.5: Different coordinate system used: world (W), head (H) and camera (C)

3.3 Projecting the 3D data onto the video

Using an EPnP algorithm [15] to obtain the camera pose throughout the sequence, one can project the data obtained with MVN Awinda onto a video recorded with the GoPro camera. After recording all the experiments, the camera was calibrated in order to obtain the intrinsic matrix A and the distortion coefficients. Then all the frames from the sequences were exported from the video and undistorted using the distortion coefficients. Then, the camera pose was obtained using a PnP algorithm on a single frame or on a small set of them, manually selecting the joints that were visible on frame. Then, the projection was made as shown in Equations 3.3 and 3.4.

Before projecting the data onto egocentric videos, sequences with the camera in a static pose and in a third person point of view were used. In this sequences, the points do not need to be expressed in the local coordinate system of the head, as the pose of the camera in the 3D world coordinate system is static. Using third person sequences will give a full view of the 3D skeleton projected onto the subject, providing more information about the accuracy of the joints recorded. In Figure 3.6 one can see some projection examples.

It's difficult to obtain fully dimensional accuracy from the MVN Awinda. To obtain the projections from Figure 3.6, all the measurements from the subject that one can introduce in the MVN Analyze software were used. These measurements (Table 2.2) lets one define completely the lengths of all the segments from the legs and the length of the arm (from the shoulder to the wrist), but leaves undefined the exact length of the upper arm and the fore arm. Also, the hands are the most critical part of the calibration of the MVN Awinda and it's easy to have some calibration error go unnoticed.

These two previous factors explain most of the error seen during the projections of the data obtained with the MVN Awinda. Generally, the part that had more error in the projections were the wrists, which often present some displacement. Due to this displacement, the 2D-3D correspondences made with wrists during the EPnP had some error in them, leading to a slightly incorrect result. Therefore, the error in the MVN Awinda propagated to the EPnP algorithm. This error was mostly seen during the projection of sequences recorded in third person, where minimal errors in the position of the camera were enhanced.

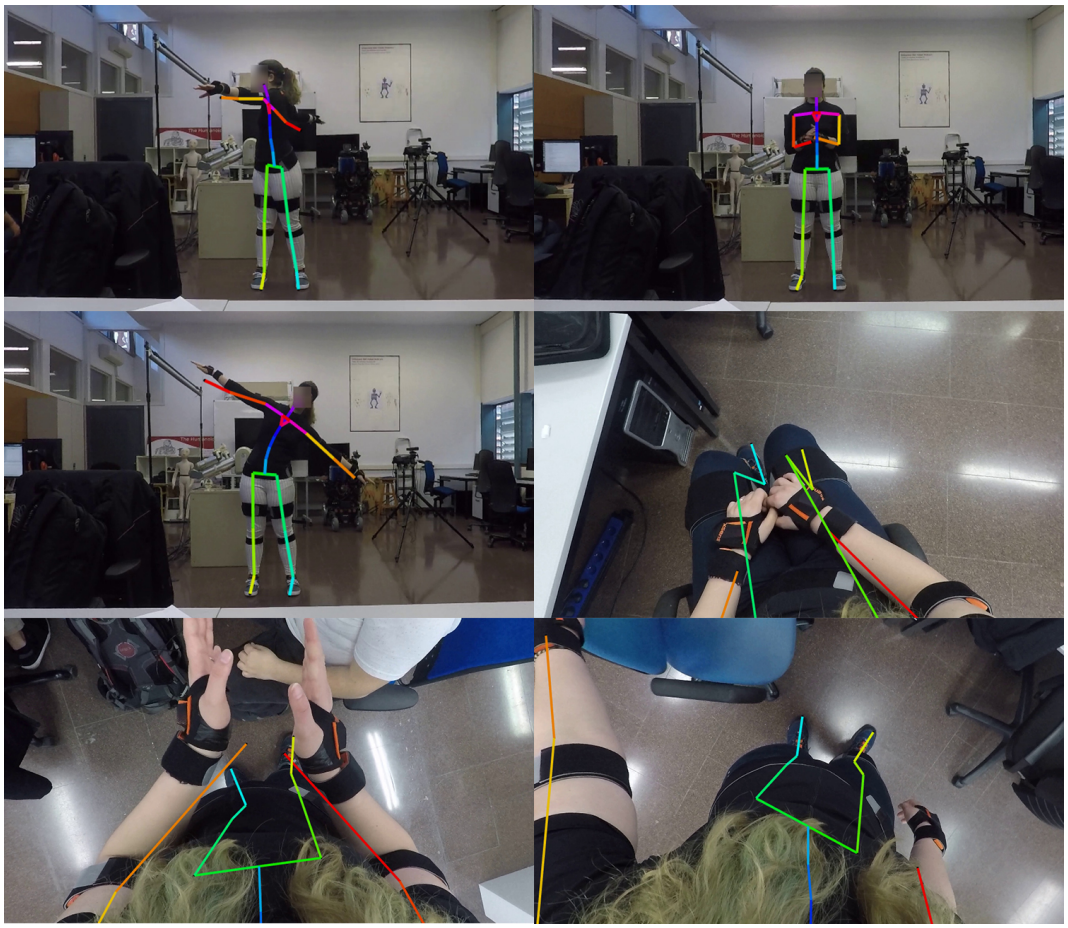


Figure 3.6: Projections of MVN Awinda data onto a video

4 Building the dataset

4.1 Data collection

The dataset was collected by 7 different individuals around the same age and in the surroundings and the inside of the Facultat de Matemàtiques i Estadística (FME). Each participant was asked to perform 6 different activities:

- **Indoors**

    1. **Walking:** The participants were asked to walk inside the building, as natural as possible, to sit in a chair and to get up from the chair.

    2. **Eating & Drinking:** The participants were asked to eat handling silverware and to drink water from a glass. The participants were sitting during this task.

    3. **Washing the dishes:** The participants were asked to wash a plate, silverware and a glass. As they were wearing sensors on their hands, the tap was closed.

- **Outdoors**

    4. **Walking:** When walking outdoors, the participants were also asked to sit in a bench and to perform some mild exercises, like squatting.

    5. **Running:** The participants were asked to jog at slow/moderate speed. Also, they were asked to jump in place and to perform some jumping jacks. Due to external causes, some of the running recordings were needed to be made indoors.

    6. **Riding a bike:** The participants were asked to ride a bike at slow/moderate speed.

These sequences were defined based on three different criteria:

- **Environment:** Indoor/outdoor
- **General pose:** Sitting-like or standing-like
- **Dynamism:** More dynamic or less dynamic

The 5 different actions that were perfomed by the participants could be classified as seen in Table 4.1.

	Environment	General Pose	Dynamism
Walking	Indoor/outdoor	Standing-like	More-dynamic
Running	Indoor/outdoor	Standing-like	More-dynamic
Riding	Outdoor	Sitting-like	More-dynamic
Eating and drinking	Indoor	Sitting-like	Less-dynamic
Washing	Indoor	Standing-like	Less-dynamic

Table 4.1: Classification of the tasks used in the dataset

The participants recorded the dataset voluntarily, without any kind of economical rewarding. Instead, the participants were gifted with an appetizer and with a visit to the Insitut de Robòtica i Informàtica Industrial (IRII). The egocentric video images and the 3D pose of the participants were captured separately.The 3D pose during all the sequences were obtained with MVN Awinda.

4.2 Data synchronization

Each pair of video and 3D pose sequence needed to be synchronized in post-processing. While MVN Analyze is able to record synchronized human motion data and video (see page 111 on MVN Awinda Manual [12]), it's only possible using the Ethernet camera, that must be connected with an Ethernet cable to the computer. This camera was not available for this project and it would not have served to the purposes of this project. As the recorded video must have a first person point of view, a portable wireless camera was needed.

A first approach to this problem would be to synchronize manually the video and the data, visualizing both and and tweak them until they match. This method is slow and inefficient, so it was quickly rejected.

The approach used at the end is only partially manual. In order to be able to synchronize both sequences of data, each subject will do a series of claps at the start of the recordings. The user have to manually select the exact frame of the video where the clap is produced. Then, the claps can be automatically detected on the data, using the acceleration data from the hand. When clapping, both hands slow down extremely quickly, resulting in high peaks in their acceleration. Those peaks can be found in the data using a signal analysis algorithm.

For this project, the *Python* library *Scipy*[8] is used, with the *findPeaks* function from its *signal* module. As seen in Figure 4.1, many spikes are produced for the same clap. For grouping them, the minimal distance between spikes in the *findPeaks* function has been set to 10 frames ($\sim 0.2s$). As other peaks may occur during the sequence, the user have to introduce a range of frames in which the claps are produced.

The goal is to obtain a synchronization value (s) that can map between the video frames indices and the data ones, which is computed with correspondences of frames between video and data given by the claps detection previously explained. A synchronization value will be computed for each correspondence and then it's averaged. Therefore, the number of correspondences is inversely proportional to the error in the value. Given a sequence of n claps, all claps will be found in all three components of the acceleration. Also, the detection can be repeated a total of R times to reduce the error even further. This results in a total of $3 \cdot R \cdot n$ correspondences that can be averaged and then rounded to obtain the final value, as can be seen in Equation 4.1. In this project, the sequence was synchronized with 3 claps and the synchronization process was repeated 3 times.

[8] https://www.scipy.org/

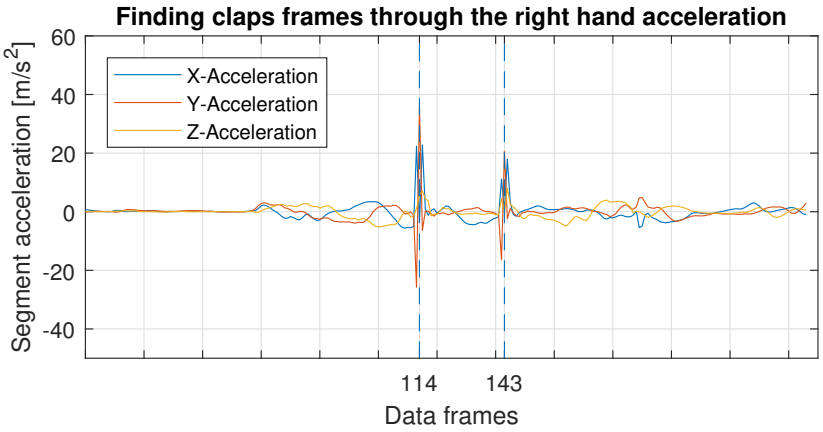


Figure 4.1: Acceleration spikes during two claps

$$s = \left\lfloor \frac{1}{3 \cdot R \cdot n} \sum_{r=1}^{R} \sum_{i=1}^{3} \sum_{j=1}^{n} (v_{i,j,r} - d_{i,j,r}) \right\rfloor \tag{4.1}$$

4.3 Dataset structure

The dataset itself is formed by a set of folders, each one containing all the frames of one of the sequences recorded. Each frame was downsized from a resolution of 1920x1080 to 960x540 (each dimension scaled by a half). Also, using the *Pillow*[9] package for Python, all the frames were saved again as an optimized JPEG making the encoder make an extra pass over the image in order to select optimal encoder settings and therefore reduce the image size without losing image quality. This was necessary in order to build a dataset with a manageable size.

In each sequence folder, there is a CSV file (Comma Separated Values) containing the 3D position of all the joints during the sequences. Each row of the CSV file follows the structure shown in Equation 4.2.

$$[Filename\ of\ the\ frame], x_1, y_1, z_1, ..., x_{23}, y_{23}, z_{23} \tag{4.2}$$

4.4 Ethical issues

To build the dataset, it was necessary to work with healthy adults to record the data. All the participants were involved in the project on a voluntary basis. Each participant received a consent where they were informed about all the aspects of the recording, such as risks, objectives or the methodology of the session. Every candidate had the right to stop the capture at any time and to withdraw from the project if wanted. Before each recording, every subject was aware of

[9] https://pillow.readthedocs.io/en/stable/

what will be done during the session and was free to refuse to perform any proposed task. All data recorded from each participant is anonymous, as there is no register that relates a certain person with a specific recording.

4.5 Summary

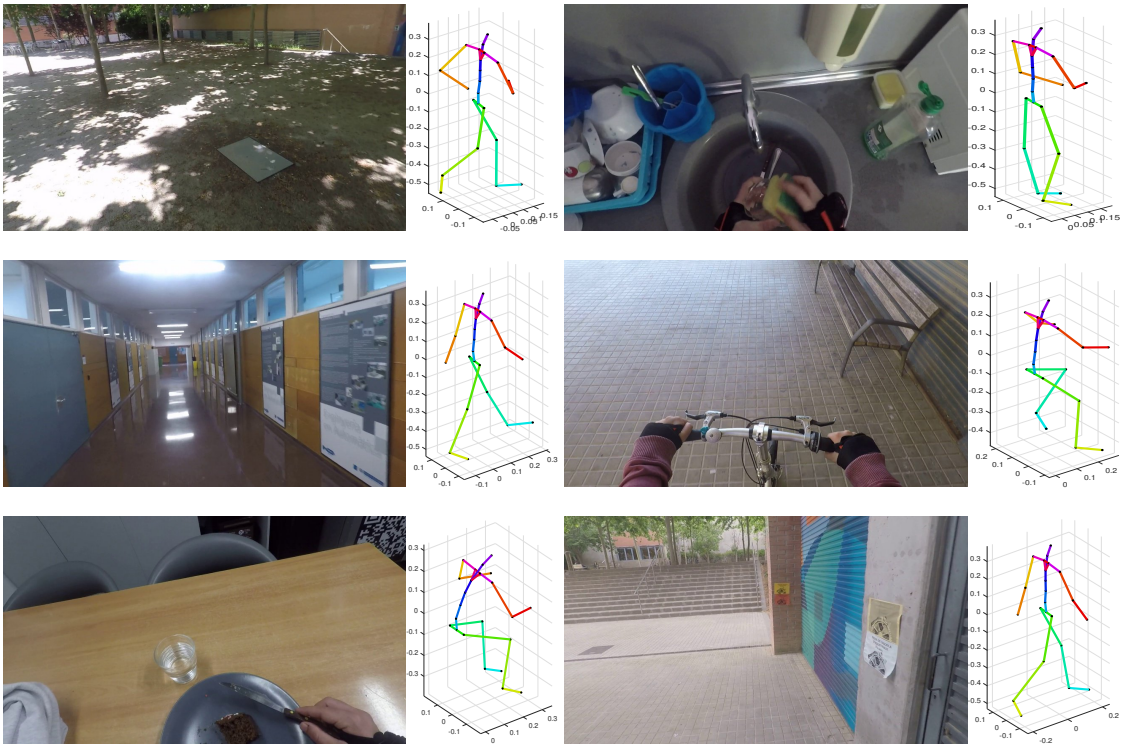


Figure 4.2: Dataset examples

The dataset is composed by a set of frames of a video recorded with a head mounted camera and the 3D pose of the camera wearer at each frame. The data was captured at 60 frames per second. The whole dataset contains 38 different sequences of an approximate duration of 2 minutes each ($\sim$ 7200 frames per sequence).

In Figure 4.3 one can observe the activities and environments that were used in the dataset, as well as how many of each there are. The most common activity is walking, as it was done both indoors and outdoors, and the most common environment is indoors as, due to external causes, some of the outdoors recording have to be canceled. In Figure 4.4 one can see the different lengths of the recorded sequences in frames. At the end, the sequences were longer than 2 minutes or 7200 frames (vertical red line in Figure 4.4) and only three of the sequences ended being much more shorter than the rests, between 5700 and 6300 frames or 1 minute 35 seconds and 1 minute and 45 seconds.

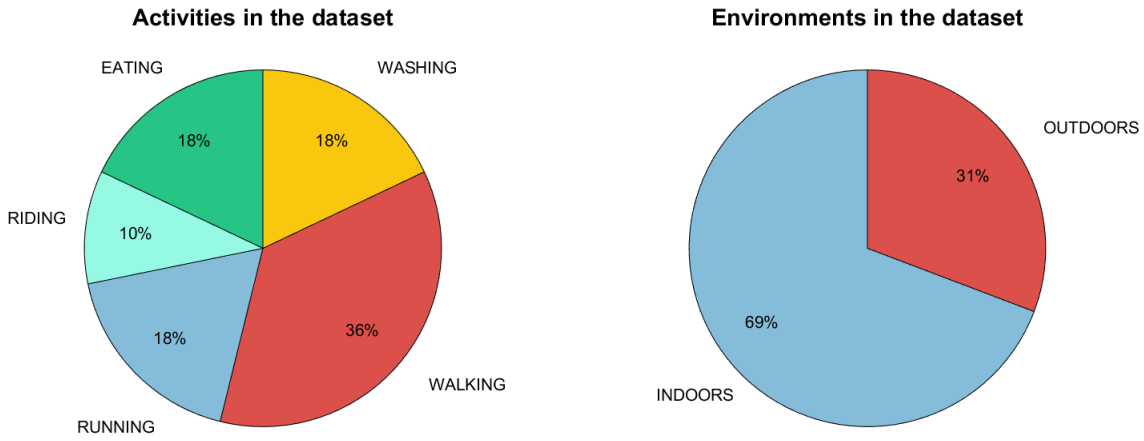


Figure 4.3: Left, activities in the dataset. Right, environments in the dataset.

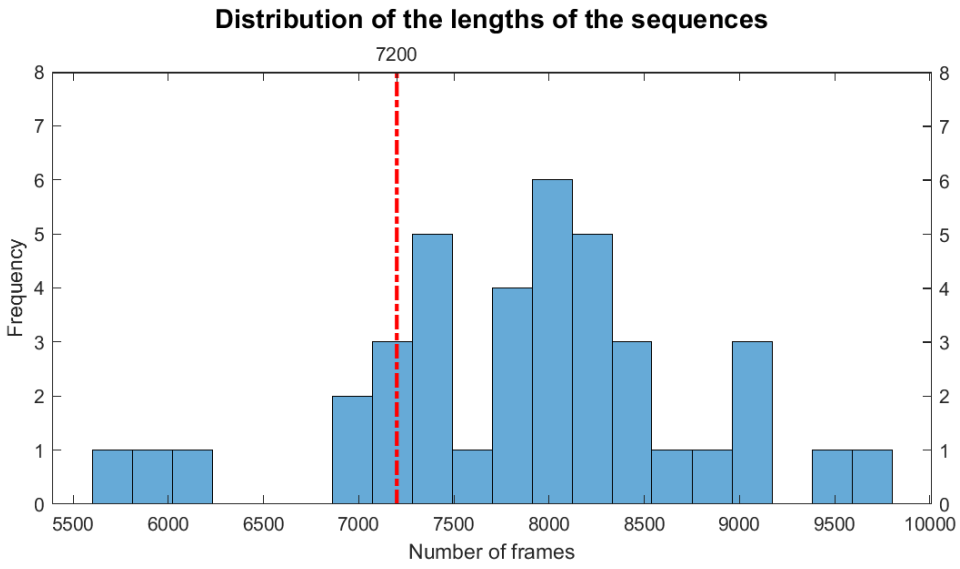


Figure 4.4: Frames per sequence (7200 frames are the equivalent of 2 minutes of footage)

5 Activity classification through a Deep Learning approach

5.1 Objective

A neural network will be trained with the dataset previously created to check its usefulness. The network will have to classify the activity performed in a recording based purely on the sequence of frames of an egocentric video. To do so, features will be extracted from each frame and its previous ones and they will be used as an input for an LSTM. The general architecture is based on the one used in [3] but without estimating the pose of a second person interacting with the camera wearer. Instead, the position of the hands and feet of the camera wearer will be searched, which will provide crucial information about the human pose.

5.2 Network Architecture Overview

The network architecture (see Figure 5.2) used in this project is based on the one utilized in the You2Me approach [3]. Beforehand, one must extract a set of features (see Figure 5.1) from the frames that will work as "clues" for the LSTM to estimate the 3D pose. The dynamic features are extracted computing a set of homographies from the 15 previous frames and the current one, that will provide information about the most recent human motion.

In the following sections, all architecture components are detailed.

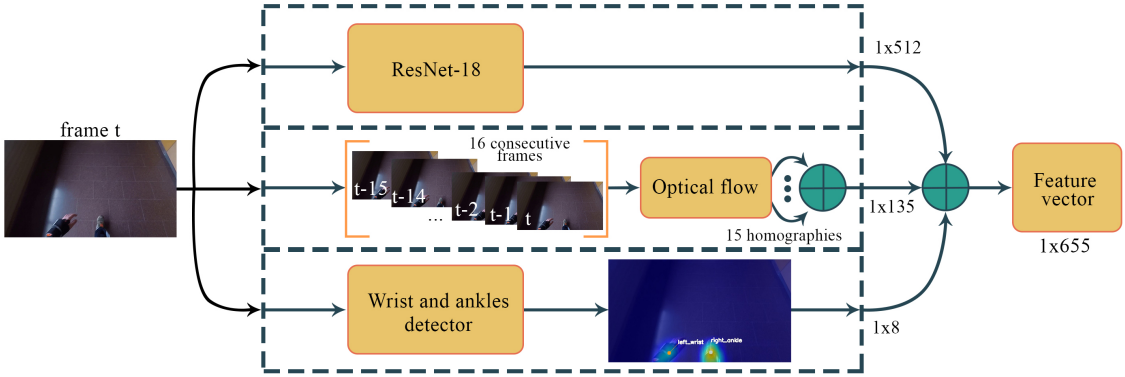


Figure 5.1: Feature Extractor

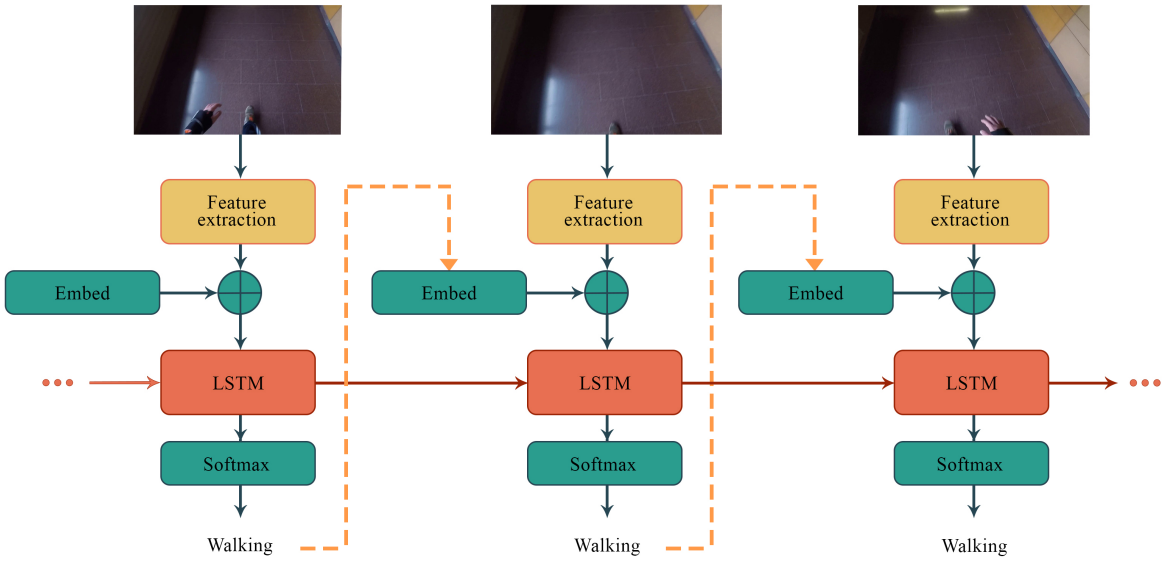


Figure 5.2: LSTM Architecture

5.2.1 Dynamical features

As seen in [3,5], different movements of the camera wearer generates different motion patterns in the video. For example, the motion patterns tend to be vertical when the subject is sitting down or radial when it's moving forwards. These motion patterns are obtained by finding point correspondences between frames. Afterwards, these motion patterns are expressed as a sequence of homographies between successive frames.

A homography is a 3D transformation between 2 planes. Given two images of the same set of points but with a different perspective, one can define a set of 2D point correspondences. These are determined by finding the coordinates in each image plane of a set of projected 3D points. Given this set of 2D correspondences, one can use a least squares method to estimate the homography matrix between the images.

$$s \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H_{3x3} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{5.1}$$

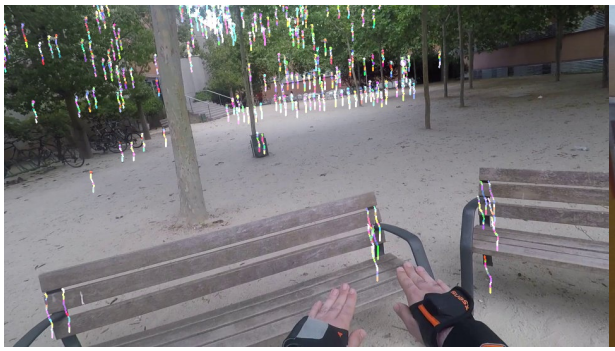
Optical flow was used to find the point correspondence between frames. At the first frame, some points are selected to be tracked, as described in [16], and then are tracked through the frame using Lucas-Kanade optical flow. The points in frame f_t are initially selected and, then, their correspondences are found at frame f_{t+1}. These correspondences will be used as initial points to compute the next correspondences at frame f_{t+2}, with the goal of save computing time. As some points may move out of the frame as the sequence is processed, if the number of initial

points is less than an arbitrary threshold higher than 4 [10], then the initials points are computed against using [16]. For this project, the threshold was set to 6 points.

Homographies are scene invariant when the camera is purely rotating, which is not strictly the case. As the data were captured at 60 frames per second, the time between frames is $\sim 17ms$ and, therefore, the camera translation between successive frames is reasonably small. Given this, one can assume the camera is purely rotating between frames.

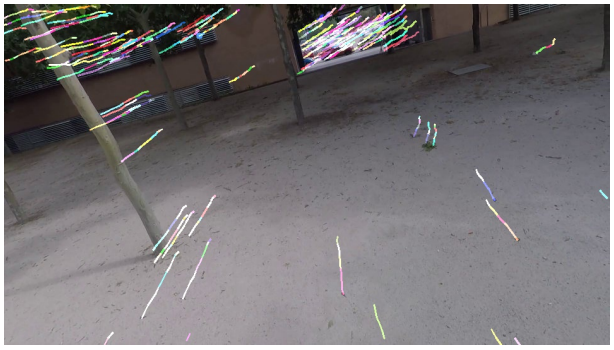
For this project, 15 homographies were computed between 16 consecutive frames (the current one and the previous 15). Each homography was normalized with the top-left corner. Then, all the homographies were vectorized and combined into a single vector $m_t \in \mathbb{R}^{135}$.

In Figure 5.3 one can see different motion patterns using frames from the dataset. The patterns are generated by drawing a line for each tracked point, connecting all its positions through the frames. One can observe how the lines are practically vertical when the subject is squatting and how they point outwards if the subject is moving forward. In addition, the strokes from the cycling pattern and the running pattern are considerably different, although both are pointed outward. The running one present a bump corresponding to the movement of the head. Alternatively, when the subject is cycling the lines are more smooth..



(a) Squatting motion pattern

(b) Running motion pattern



(c) Cycling motion pattern

Figure 5.3: Different motion patterns in sequences from the dataset

[10] The minimal number of points to compute a homography is 4.

5.2.2 Static features

As seen in Section 5.2.1, the dynamic features reveal significant information about the subject activity, when the camera pose is changing relatively quick. For more static activities, they do not provide enough information. To account for this, one can extract static features from the current frames. The scene appearance can provide relevant information of the task that's being executed. For example, if a plate and a set of silverware are seen on the frame, the subject is likely eating and, therefore, sitting.

To extract these static features, a ResNet-18 [17] pre-trained on ImageNet was used. ResNet architecture was introduced by Microsoft in 2015, wining the ILSVRC (ImageNet Large Scale Visual Recognition Challenge). Deeper Neural Networks present the vanishing gradient problem in which, as the loss is backpropagated during training, the weights gradients keep getting smaller. When this happens, they may be unable to fully modify the net and optimize it. The core idea behind ResNet is introducing an identity shortcut connection that skips one or more layers. Then, while backpropagating, the gradients can flow through the shortcut connections to the initial stages.

The last fully connected layer of the pretrained ResNet-18 was dropped, using the average pool one as an output to obtain a vector $s_t \in \mathbb{R}^{512}$.

5.2.3 Body part detection

Another set of features that could provide valuable information is to know whether some body parts of the camera wearer can be seen on screen and where are they, if they are in the frame. To look for joints, a pre-trained version of the model created in [2] was used. This CNN model uses two branches to predict the joint positions of the images: one predicts a set of 2D confidence maps (one for each joint) and one predicts a set of 2D vector fields of part affinities. The Part Affinity Fields (PAF's) encode the position and orientation of the limbs in the image, which allows the model to improve the predictions for the body joints position. The results of these two branches are then concatenated and sent for the following stage.

For this project, the PAF's were unuseful, as the limbs of the camera are out of view most of the time, and no association could be made. Therefore, only the branch that predicts 2D confidence maps for the first stage was used. Then, the position of each joint are determined from the confidence maps by finding a peak in each confidence map, as well as its probability.

The initial model uses a set of different scales when computing the confidence maps, which enables to find body joints of various sizes. As the joints that will typically be seen in an egocentric point of view (hands and feet) are in the same range in terms of size on screen, there is no need in using a set of scales and a single value [11] was utilized.

[11] The scale value used was 0.20444444444444446.

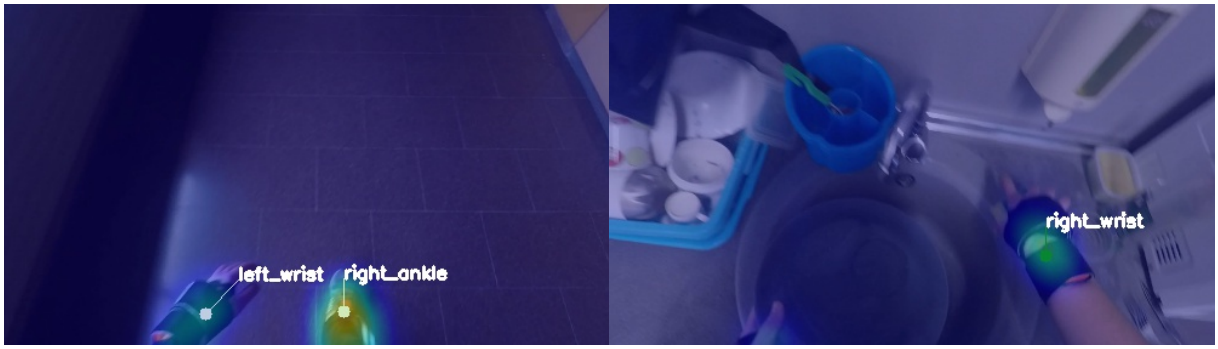


Figure 5.4: Examples for the joint detection

As only hands and feet are seen sometimes on the frame, all other joints were filtered out as they would exclusively appear when any false-positive[12] happened. Therefore, only four points in the frame were looked for, both wrists and ankles. If a joint was unfound, the u and v coordinates were equal to zero. Otherwise, the u and v coordinates of the joint in the image were normalized by the width and the height of it, respectively. Then, these four 2D vectors were concatenated in a single vector $b_t \in \mathbb{R}^8$ in the following order: right wrist, left wrist, right ankle and left ankle.

5.2.4 Long Short Term Memory networks (LSTM)

The features extracted from the current frame are finally concatenated in a feature vectors $f_t \in \mathbb{R}^{512+135+8}$. This vector works as an input for a LSTM network, a type of recurrent neural network. Traditional neural networks, also known as Fully Connected Networks, are unable to process sequential information. Recurrent Neural Networks (RNN) address this issue, which can be thought as a sequence of traditional neural networks, each one passing a message to the next one. This message is recalled as the hidden state of the network. In Figure 5.5 there is a visualization of the structure of an RNN.

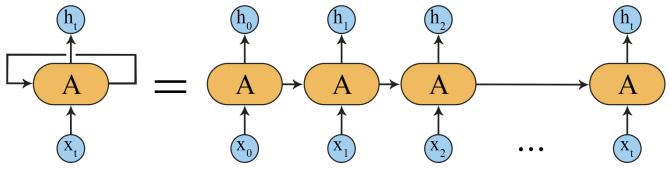


Figure 5.5: Unrolled structure of an RNN

Basic RNN work fine when only recent information from previous inputs is needed, but in cases where more context is needed, RNN fall short in memory. This is due to the vanishing gradient problem [18] that makes the gradient more and more smaller as it's back propagated, vanishing it.

[12] As the model was trained using images in a third person point of view, the perspective of the egocentric images may induce the model to false-positives

When more memory is needed, Long Short Term Memory networks (LSTM's) can be used, as they don't present this problem. The structure of an LSTM can be visualized in Figure 5.6. The core idea behind LSTM's is the cell state, which works as a conveyor belt passing through the LSTM. The cell state carries the information from the previous input, it is carefully transformed by structures called gates and then it's passed to the following state. The cell state does not substitute the hidden state that could be found in RNN, the hidden state is also present in LSTM's.

First, the forget gate layer decides which information is going to be "forgotten" and, therefore, deleted from the cell state. Next, the input gate layer decides which information from the current input and the previous hidden state is going to be added to the cell state. Ultimately, the output gate layer generates the current hidden state using the updated cell state, the previous hidden state and the current input.

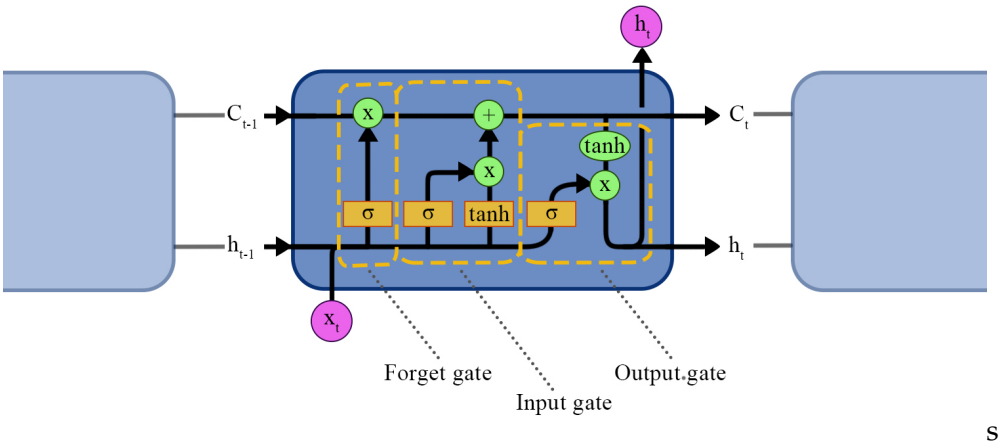


Figure 5.6: Basic structure of an LSTM

5.3 Training

As each sequence will be tagged as a unique task, all the activities in the sequence that do not correspond to that task have to be removed. For example, at the last minute of most running recordings, some squats and jumps were performed. If they were not taken out, they would introduce noise to the system. Also, most of the sequences didn't start right away with the main activity. Therefore, from each sequence a middle portion will be utilized for training.

The dataset was separated using the participants, using all the sequences of five participants for training and two of them for testing. At any combination used, it was ensured that all tasks were present in the training and the testing set. During training, 85% of the training set was used for training and 15% for validation.

For training the model, a batch size of 64 was used. The LSTM was defined with a fixed embedding dimension of two and a hidden state dimension of 16. The model was trained for 11 epochs in total, with a learning rate equal to 0.0001 for the first 8 epochs and then it was decreased to 0.00001.

The loss for the network was the cross entropy loss across the processed sequence for predicting the proper task. The dataset does not contain an equal number of frames per task. In Figure 4.3 one can see that there are more frames for walking than for the other activities. Put differently, the dataset is unbalanced. When this happens, it's sensible to weight the loss for each category. This means that, before executing the backward propagation of the loss, it will be multiplied by a different weight in function of the category of the ground truth. The weight for each category is calculated as seen in Equation 5.2, where $total$ stands for the total number of data in the dataset and n_i stands for the number of occurrences in the dataset of the category i.

$$w_i = \frac{total - n_i}{total} \tag{5.2}$$

5.4 Results

The final model trained had a 92.58% of accuracy in the training set, 93.65% in the validation set and 61.10% in the testing set. In Figure 5.7 one can see the training and validation curves for the loss and the accuracy.

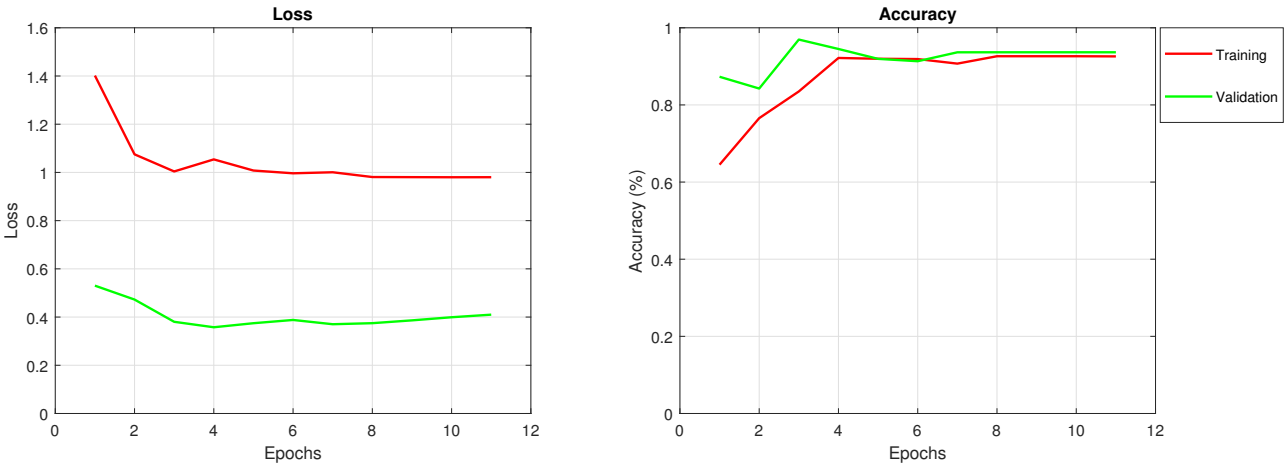


Figure 5.7: Loss and accuracy curves during training and validation, when no user in test set has been seen during training

The testing accuracy is lower than the one during training and validation but it's significantly better than the accuracy that one would get by random guessing ($\sim 20\%$). Therefore, the model seemed to have learnt some general rules that can be applied outside the training dataset but some rules that do not generalize well.

In Figure 5.8 one can see the accuracy of the model during testing for each one of the categories. The model excels in classifying the walking and riding sequences (100% of accuracy), fails with the eating (6.96%) and the running (8.97%) ones and does a good job classifying the washing ones (68.09%). In Table 5.1 there are 5 random sequences from the testing set, one for each different task, and the respective predictions made.

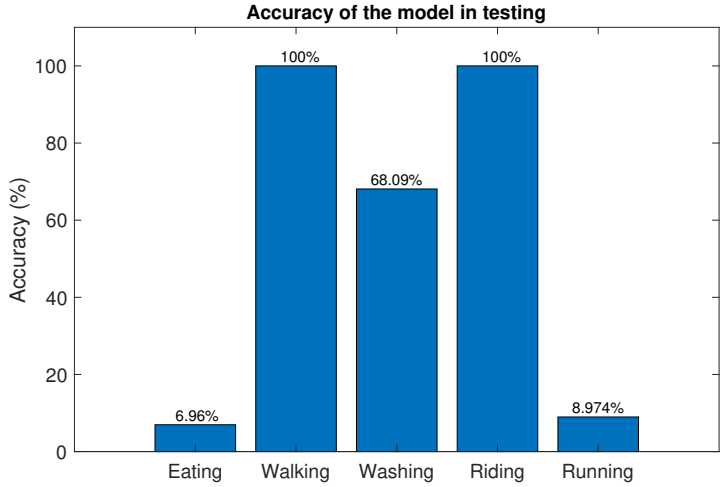


Figure 5.8: Accuracy during testing for each category

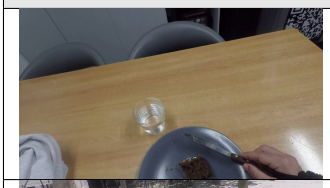
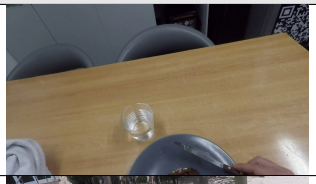
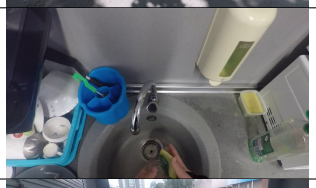
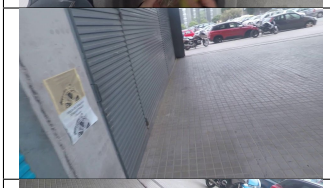
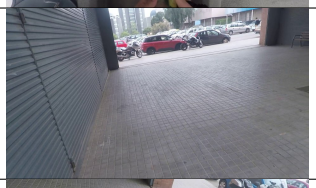
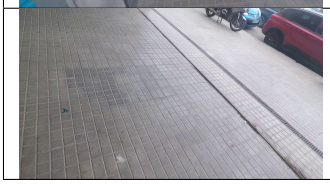
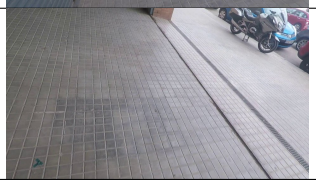
First frame	Final Frame	Ground Truth	Predicted
		Eating	Washing
		Walking	Walking
		Washing	Washing
		Running	Walking
		Riding	Riding

Table 5.1: Prediction examples for the final model

In Figure 5.9 one can see the confusion matrix of the predictions during testing. It's a table of counts where each row corresponds to the predicted output of the model and each column to the ground truth. Ideally, one wants that all the counts are placed in the diagonal of the matrix and that the rest of the elements outside the diagonal are equal to zero. The model seems to

be identifying most of the eating segments as washing ones and most of the running sequences as walking ones. While it is misclassifying many eating and running examples, it's misunderstanding them with tasks of similar dynamism, which corresponds to resembling camera motion patterns. Therefore, it seems reasonable that the model could mistake running with walking as their motion patterns are similar, or eating and washing as the camera moves much more slower than in the other sequences. Hence, the model could be understanding the different dynamisms of the sequences correctly.

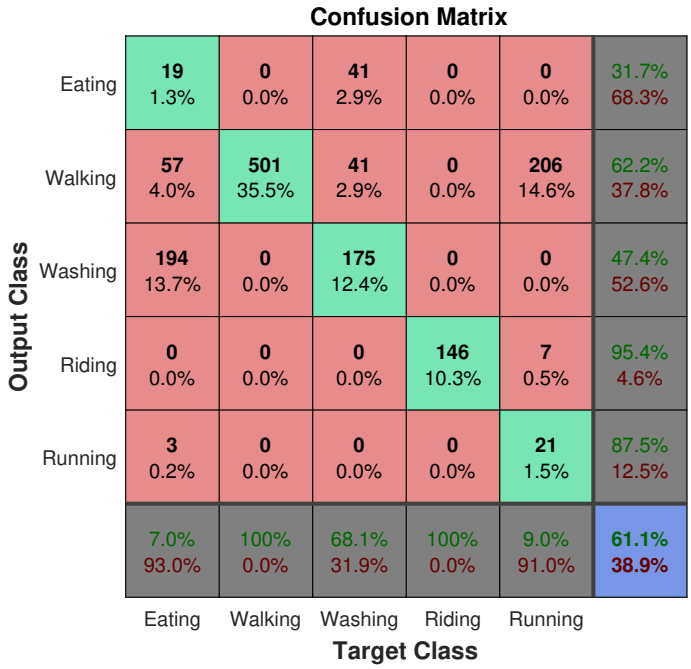


Figure 5.9: Confusion matrix

Separating the dataset using the subjects is a good practice, as one can actually check how the model would perform with a totally new sequence recorded by a different person from the ones that it has seen. However, it also leads to worse results in accuracy during testing. In order to find out how much of the error during testing is due to the difficulty of the test, the same model will be trained using the same dataset but with a different splitting. All the batches from all the sequences will be loaded and shuffled. Then, $5/7$ of the data will be for training and $2/7$ for the final test. Also, the training data will be separated, 85% for actual training and 15% for validation. The reason behind the $5/7$ separation is to utilize the same amount of data to train the model as before. The network was trained for 15 epochs in total, with a learning rate equal to 0.0001 for the first 8 epochs and then it was decreased to 0.00001.

The curves of loss and accuracy during training and validation can be seen in Figure 5.10. The curves seem less smooth than the previous ones in Figure 5.7, but the final results are relatively similar. This model had a 93.97% of accuracy in the training set, 97.20% in the validation set and 68.36% in the testing set. The model predicts better results while testing than the first model (61.10%).

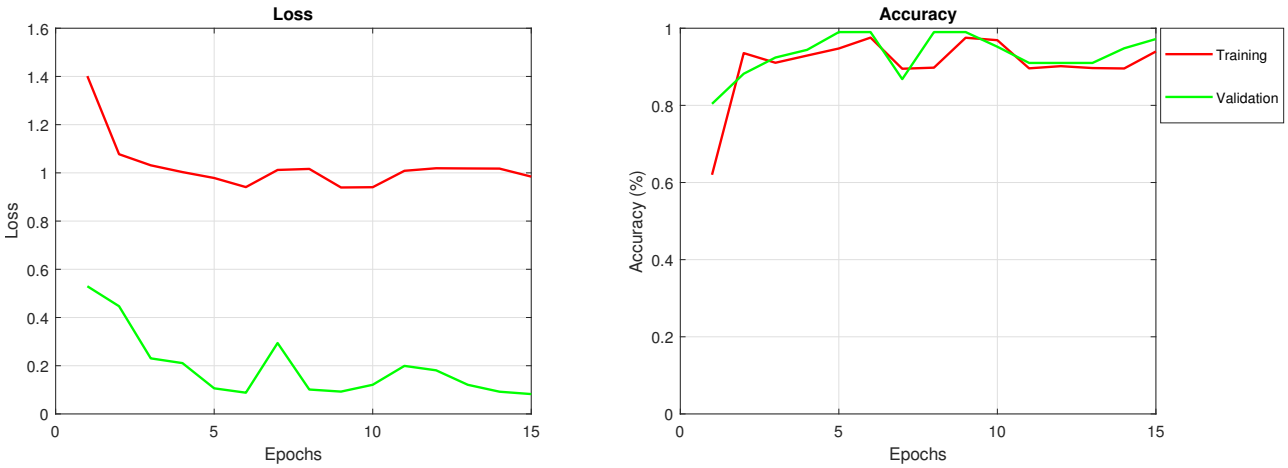


Figure 5.10: Loss and accuracy curves during training and validation, when all users in the test set have been seen during training

6 Pose inference trough a DeepLearning approach

6.1 Objectives

The core objective of this section is to propose a model design that could be trained to estimate the 3D pose of the camera wearer at each frame of an egocentric video. The same network architecture from Section 5 will be used. Instead of trying to classify the sequence in the various activities performed, the goal is to infer the posture of the camera wearer at each frame. To do so, the whole group of 3D poses in the dataset will be discretised in a set of 500 poses using K-means.

6.2 Preprocessing the dataset

The whole dataset is formed by more than 300,000 frames with its corresponding human poses. A LSTM can be trained to perform regression but are generally more robust in classification tasks. As done in [3], the large number poses in a dataset can be quantized to a narrower set of postures. To do so, all the poses from the dataset have been grouped using K-means.

To determine the number of groups K, different values within the range $K \in [1, 500]$ have been tried. To select the optimal, the average distance from all the poses to the centroid of their corresponding group was computed. As all the poses were normalized in scale and orientation, the distance is expressed as a percentage of the width of the shoulders. In Figure 6.1 one can perceive how the average distance to the centroids decreases as the number of clusters increases.

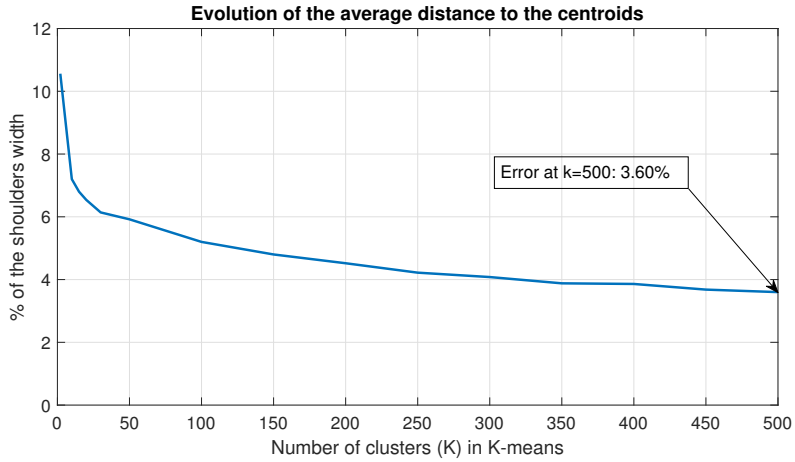


Figure 6.1: Average distance of the poses to the centroids of their correspondent clusters (in percent of the distance between shoulders) against the number of clusters

K-means is a unsupervised machine learning algorithm that looks for a specific number of clusters K in the data. A cluster, in Data Science, is a group of data point that is aggregated due to their closeness in the space. In most cases, this closeness denotes some kind of similarity in the data. In this project, each observation is a set of 3D points, each one representing the 3D pose of the camera wearer. The 23 3D points are concatenated in a 69-dimensional vector. Vectors that are close in the 69-dimensional space will represent similar human poses.

To find the clusters, K initial centroids are created in random locations of the space. Then, the position of the clusters is optimized recursively, minimizing the distance of the observation to their designated clusters. In this project, Euclidean distance was used, but there are many others available, like Manhattan or Pearson.

For this project, $K = 500$ was used. The average error is of 3.6% of the shoulders width. Assuming a shoulders width of 50 cm, the average error is of 1.8 cm. From this, one can see the quantization of the data is quite fine-grained. In Figure 6.2, one can observe five examples out of the 500 clusters computed. In each plot, both the centroid (in red) and some random poses that belong to the cluster of that centroid (in gray) are overlapped. From this, one can observe the poses within the same group are relatively similar.

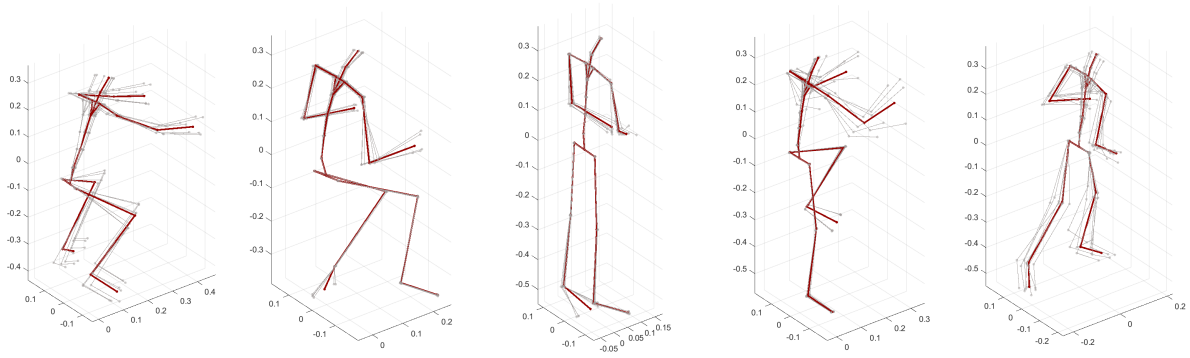


Figure 6.2: Clusters examples. In red, the average pose of the cluster. In gray, some poses of the cluster

6.3 Network Architecture

The network architecture proposed is the same as the one used in the previous section for classifying the task. Initially, one must extract a set of features (see Figure 5.1) from the frames that will work as "clues" for the LSTM to estimate the 3D pose. The LSTM will have to predict a vector of length 500, where the element i corresponds to the probability of belonging to the pose cluster i.

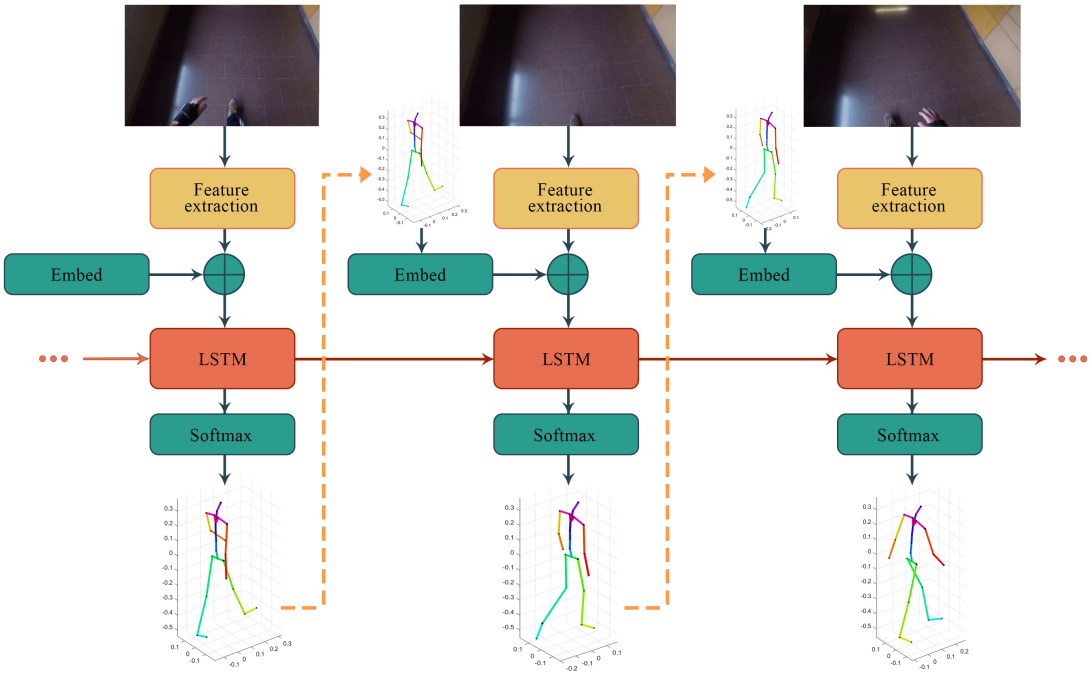


Figure 6.3: LSTM Architecture

7 Budget

Although this work is a Final Degree Project, the budget showed in this section was calculated as it was a real project conducted by an actual company. Therefore, the main engineer pay rate was the one of a technical industrial engineering instead of a student.

7.1 Personnel cost

This project was developed by a student, with the help of several researchers from the IRII. The amount of work involved in this project by the personnel can be summarized as following:

- Four months of work by the student with a daily dedication between 4 and 8 hours per day, 6 days per week
- Assistance by the director and codirector of the project with an average attention of 3 hours per week each
- Additional assistance by two researchers with an average attention of 1 hour per month each

Given this amounts of work and assuming and hourly rate of 20 €/h for the student as a techincal engineer and 50 €/h for the researchers[13], one can compute the personnel cost of the project. The breakdown of the cost can be seen in Table 7.1.

	Dedication (h)	Hourly rate (€/h)	Cost (€)
Technical Engineer (student)	600	20	12 000
Director and codirector	90	50	4 500
Researchers (advisors)	8	50	400
Total	-	-	**16 900**

Table 7.1: Personnel cost breakdown

7.2 Equipment and license cost

The MVN Awinda and its license were acquired by the Manipulation and Perception Group at the IRII at the price of 50,000 euros, the GoPro camera utilized was bought at the cost of 330 euros, the main computer used was purchased for 600 euros and the GPU used for training costed 1000 euros. None of the previously mentioned products were obtained solely for this project and it's expected to be operated during all their lifetime. Therefore, the equipment cost will be computed by dividing the total worth of each hardware by its lifespan and multiplying it by the time that it has been used for this project.

[13] Approximate pay rate for external hiring of a CSIC researcher

Equipment	Total cost (€)	Expected lifespan (months)	Time of usage (months)	Cost (€)
MVN Awinda	50 000	240	4	833.33
Go Pro	330	36	4	36.67
Main Computer	600	60	4	40
GPU (Tesla k40c)	1000	60	4	66.67
Total	-	-	-	**976.67**

Table 7.2: Equipment and license cost breakdown

7.3 Energetical cost

The energetical cost of these project comes purely by the electricity usage of the different equipments used and of the workplace. To compute the energetical cost, a rate of 0.11 €/kWh [14] will be used.

Source	Power usage (W)	Usage time (h)	Energy (kWh)	Cost
Main computer	220	600	132	14.52
GPU (Tesla k40c)	245	150	36.75	4.04
Workplace[15]	360	600	216	23.76
Total			**384.75**	**42.32**

Table 7.3: Electricity usage and energetical cost breakdown

7.4 Total cost

Concept	Cost (€)
Personnel cost	16 900
Equipment cost	976.67
Energetical cost	42.32
Total	**17 919**

Table 7.4: Caption

[14] This value is extracted by actual rates from ENDESA. As the rates change every hour between 0.10 and 0.12 €/kWh, the average was taken.

[15] Consumption assumed based on the average annual consumption of $52.5 Kwh/m^2$ extracted from [19]. The office has an approximate surface of $60 m^2$.

8 Environmental impact

As no physical product have been produced in this project, the study of the environmental impact is limited to the energy consumed during the realization of the project and the environmental impact of the equipment used.

8.1 Energy sources

To estimate the environmental impact of the electricity consumed throughout the project, one needs to take into account where the energy came from. According to the last press release [20] by *Red Eléctrica de España* (REE) the energy sources in the peninsular area of Spain during the period from January to May in 2019 are the ones showed in Figure 8.1.

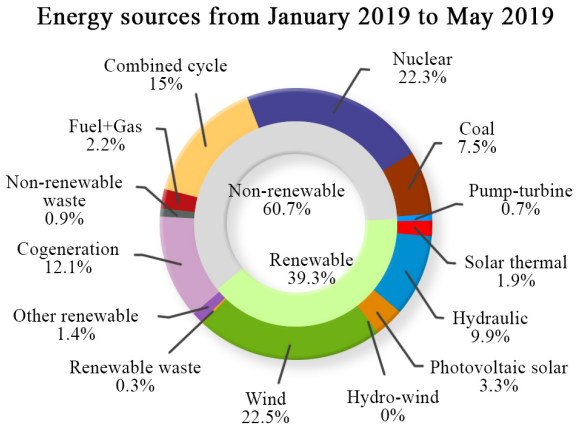


Figure 8.1: Energy sources in Spain from January 2019 to May 2019

Using the carbon footprint of different energy sources expressed in $g\ CO_2\ eq/kWh$ extracted from [20], one can estimate the carbon footprint due to the electricity consumed in this project. To do so, the total energy consumption of the project was taken from Table 7.3 and the percentages from Figure 8.1 were utilized to estimate how many of that energy came from each energy source. Then, using the carbon footprint associated with each energy source, one can estimate the carbon footprint of all the electricity consumed. Because of lack of information in Figure 8.1 about the combustibles used in *Fuel+Gas* and *Cogeneration*, it was assumed that one half of the energy was produced using oil and the other half using natural gas.

Energy source	Carbon intensity ($g\ CO_2\ eq/kWh$)	Percentage of energy used (%)	Energy used (kWh)	Carbon footprint ($kg\ CO_2\ eq$)
Coal	1001	7.5	29.08	29.11
Oil	840	7.15	27.72	23.29
Natural Gas	469	7.15	27.72	23.29
Photovoltaic solar	48	3.3	12.79	0.61
Geothermal	45	0	0	0
Thermal solar	22	1.9	7.37	0.16
Biomass	18	0	0	0
Nuclear	16	22.3	86.47	1.38
Wind	12	22.5	87.24	1.05
Ocean	8	0	0	0
Hydraulic	4	9.9	38.39	0.15
Total	-	-	-	**68.76**

Table 8.1: Carbon footprint of the electricity used in the project

8.2 Equipment

Part of the environmental impact of the equipment has already been taken into account the previous analysis of the electricity consumed during the project. In this section, other parts of the impact generated by the equipment will be analyzed.

Both MVN Awinda and the GoPro share the most critical factors of their environmental impact: the lithium battery and the materials used in their manufacture. These batteries produce massive environmental and social impact due to the extraction of the metal and the its lack of recyclability. In addition, both GoPro and MVN Awinda are mostly fabricated out of plastic, material that is making the headlines nowadays due to the environmental damage that have caused over the last decades. The GPU case is made from plastic too, and it will also be taken into account.

8.2.1 Lithium batteries

Lithium-ion batteries production has been increased by eight times over the last decade due to the demand of electrical products and electric vehicles. There are two principal sources of lithium: *spodumene*, a silicate found in pegmatites, and the lithium chloride found in certain lakes.

Bolivia, Chile and Argentina are known as the *Lithium Triangle*, as their salted lakes constitute the 75% of the known lithium reserves. The ecosystems of these countries have an extremely low amount of annual rainfall. Therefore, their communities, fauna and flora have shown high susceptibility to the water availability, as small changes in the amount of water accessible can lead to large negative impacts. As lithium extraction requires processing large amounts of water, there is a growing concern in the negative impact that collecting this metal may have.

Despite the great deal of attention that this concern has currently got in the international press, there are not many studies about this potential environmental impact. In addition, the con-

ditions under which the lithium extraction is conducted (lack of cleanliness, child labour ...) have captured the attention and the concern of many organisations, like Amnesty International or UNICEF. Recently, the *Global Battery Alliance* have been created [21], an organisation whose goal is to work in ensuring the battery production remains socially responsible and economically and environmentally sustainable.

Lithium batteries life-cycle is relatively short and the recycling business of these batteries is also emerging, because of the increase of its demand. Regrettably, recycling them is an expensive and, usually, non-profitable process. This situation is quite ordinary for most types of batteries in the market (cobalt, iron phosphate and lithium, nickel, cadmium...), with the exception of the lead-acid ones, whose recycling process have shown to be profitable.

The cause of the costly recycling operation of the lithium batteries is the complexity of it, which includes the their gathering, shipping, classification, shredding, splitting into metallic and non-metallic materials, neutralization of toxic substances and the fusion and refining of the recovered metals.

8.2.2 Material

The actual lists of materials for the equipment used were unavailable. Instead, the cases of the Go Pro, the MVN Awinda sensors, MVN Awinda charging stations and the GPU used are assumed to be made out of HDPE (high density polyethylene), as it's a common plastic used for robust packaging. In Table 8.2 one can see the estimated carbon footprint produced by all the cases previously mentioned. To estimate this carbon footprint, a value of 3380 $kg\ CO_2\ eq$ per tonne [16] of HDPE produced was used.

	Estimated Weight (g)	Carbon Footprint ($kg\ CO_2\ eq$)
17 motion trackers cases	272	0.92
3 charging stations	300	1.01
GoPro case	25	0.08
GPU case	200	0.68
Total	**797**	**2.7**

Table 8.2: Estimated carbon footprint due to HDPE cases

8.2.3 Summary

In relation to other more product oriented projects, the environmental impact of this one is relatively non-existent, as there was no production that involved high use of energy sources or material. Nonetheless, every engineering project has to be aware of its ecological impact, small or large.

[16] Value extracted from [22]

Conclusions

This project has shown that MVN Awinda, being based on inertial and wireless sensors, is well suited to build a dataset of human motion in a vast variety of environments. This opens up the possibilities of building datasets with motion data recorded outdoors and in challenging environments, where visual based movement capture systems would have not worked effectively. Therefore, more genuine and relevant information can be obtained. This comes at the expense of losing a minor portion of segment length accuracy.

Also, the network developed and trained for task classification have shown that a dataset built using MVN Awinda and a GoPro can be effectively utilized for Deep Learning purposes. For future works, a larger and more balanced dataset built using the same equipment could be expected to train a similar model with better results. In addition, it would be capable to successfully train a network for estimating the camera wearer 3D pose implementing a similar model to the one proposed in this project.

Acknowledgements

To begin with, I would like thanking Francesc Moreno and Mariella Dimiccoli, director and codirector of this project, for all their support and for being always there to solve any doubt that could appear.

On top of that, I want to thank everyone from the IRII for making posible this enriching experience, specially to the Perception and Manipulation Group. Also, special thanks to Albert Pumarola and Antonio Agudo for their advices and collaboration throughout this project, and to Maria Alberich for her help during the project.

Last not least, I want to heartily thank Helena, my family and friends for the support that they have always given to me, specially these last months. Special thanks also to the willing participants who voluntarily helped me build the dataset: Aitor, Carlos, Eloi, Helena, Marina, Mohamed and Xavi.

Glossary

- **Accuracy:** In Machine Learning, percentage of the samples that have been correctly classified by a network.
- **Cluster:** In Data Science, a close group of data.
- **Convolutional Neural Network (CNN):** Deep Learning algorithm with takes an image as an input and consists of an input layer, an output layer and several hidden layers that include multiple convolutional layers, pooling layers, fully connected layers and normalization layers.
- **Field of View (FOV):** Area captured by the camera imaging sensor.
- **GoPro:** Wearable camera used in this project.
- **IRII:** Institut de Robòtica i Informàtica Industrial.
- **K-means:** Clustering algorithm. Its goal is to find a number K of groups in the data.
- **LSTM:** Long short-term memory, a type of artificial recurrent neural network.
- **Learning rate:** In Machine Learning, hyperparameter that controls the rate at which a network learns. It defines the length of the step while optimizing the model using stochastic gradient descent.
- **Loss function:** In Machine Learning, method of evaluating how well the algorithm models the dataset. It maps the output of the network to a real number and it's the function that will be optimized while training a network.
- **MVN Analyze:** Software used to calibrate and capture data with the MVN Awinda system.
- **MVN Awinda:** Human motion recording system based on wireless inertial sensors placed onto the person.
- **Recurrent Neural Network (RNN):** Neural Network architecture design to be able to recognize characteristics in sequential data (audio, music, video...) and learn from it.

Bibliography

[1] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660, 2014.

[2] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1302–1310, 2017.

[3] E. Y. K. Ng, Donglai Xiang, Hanbyul Joo, and Kristen Grauman. You2me: Inferring body pose in egocentric video via first and second person interactions. *CoRR*, abs/1904.09882, 2019.

[4] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018.

[5] Hao Jiang and Kristen Grauman. Seeing invisible poses: Estimating 3d body pose from egocentric video. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3501–3509, 2017.

[6] Ye Yuan and Kris Kitani. *3D Ego-Pose Estimation via Imitation Learning*, pages 763–778. 09 2018.

[7] Dima Damen (Aldamen), Teesid Leelasawassuk, Osian Haines, Andrew Calway, and Walterio Mayol-Cuevas. You-do, i-learn: Discovering task relevant objects and their modes of interaction from multi-user egocentric video. In *Proceedings of the British Machine Vision Conference*. British Machine Vision Association (BMVC Proceedings), 2014.

[8] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*, 2018.

[9] Alireza Fathi, Yin Li, and James M. Rehg. Learning to recognize daily actions using gaze. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, pages 314–327, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[10] Hamed Pirsiavash and Deva Ramanan. Detecting activities of daily living in first-person camera views. pages 2847–2854, 05 2012.

[11] Stefano Alletto, Giuseppe Serra, Simone Calderara, and Rita Cucchiara. Understanding social relationships in egocentric vision. *Pattern Recognition*, 48(12):4082 – 4096, 2015.

[12] © Xsens Technologies B.V. *Xsens MVN User Manual*.

[13] Martin Schepers, Matteo Giuberti, and G Bellusci. Xsens mvn: Consistent tracking of human motion using inertial sensing. Technical report, Xsens Technologies, 03 2018.

[14] Amazon page for the GoPro Hero 5 Black. https://www.amazon.es/GoPro-Hero5-Black-Deportiva-Bluetooth/dp/B01M14ATO0. Accessed: 26-05-2019.

[15] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81, 02 2009.

[16] Jianbo Shi and Carlo Tomasi. Good features to track. *Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 600, 03 2000.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[18] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 6(2):107–116, April 1998.

[19] Enectiva. Office electricity consumption. https://www.enectiva.cz/es/blog/2015/06/ideas-energia-edificio-de-oficinas/. Accessed: 26-05-2019.

[20] REE Press Office. La demanda de energía eléctrica de españa. https://www.ree.es/es/sala-de-prensa/notas-de-prensa/2019/06/la-demanda-de-energia-electrica-de-espana-desciende-un-0-8-en-mayo. Accessed: 26-05-2019.

[21] We Forum. We forum. https://www.weforum.org/global-battery-alliance/home. Accessed: 29-05-2019.

[22] Intertek. An environmental comparison of polymers. Technical report, May 2011.