Computer Vision For Body Tracking in Professional Enviroments

Pablo Vicente Moñivar

Máster en Investigación e Innovación en Tecnologías de la Información y las Comunicaciones

MÁSTERES DE LA UAM

2018 - 2019

Escuela Politécnica Superior

Universidad Autónoma de Madrid

UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Computer Vision For Body Tracking In Professional Environments

Author: Pablo Vicente Moñivar
Advisor: Sotiris Manitsaris
Supervisor: Jesús Bescós Cano

MASTER THESIS

Jul 2019

Pázmány Péter Catholic University
Faculty of Information Technology and Bionics

Resumen

El objetivo de este trabajo de investigación es construir las bases para una aplicación de smartphone que proporcione las funcionalidades necesarias para grabar datos de movimiento humano, entrenar algoritmos de machine learning y reconocer gestos profesionales.

El trabajo se ha realizado, en primer lugar, aprovechando las cámaras de los nuevos teléfonos móviles, ya sean infrarrojas o estereoscópicas, para grabar datos RGB-D. Luego, un algoritmo de estimación de posición basado en el aprendizaje profundo extrae el esqueleto humano en 2D y exporta la tercera dimensión utilizando la profundidad obtenida de las cámaras. Por último, utilizamos un motor de reconocimiento de gestos, que se basa en los modelos K-means, para discretizar los datos, y Hidden Markov Models(HMM), para reconocer y clasificar los gestos. El rendimiento del algoritmo de aprendizaje ha sido probado con gestos profesionales utilizando tres bases de datos manualmente grabadas.

Palabras Clave

Estimación de pose, mapas de profundidad, reconocimiento de gestos, Hidden Markov Models, K-means, smartphone

Abstract

The goal of this work is to build the basis for a smartphone application that provides functionalities for recording human motion data, train machine learning algorithms and recognize professional gestures.

First, we take advantage of the new mobile phone cameras, either infrared or stereoscopic, to record RGB-D data. Then, a bottom-up pose estimation algorithm based on Deep Learning extracts the 2D human skeleton and exports the 3rd dimension using the depth. Finally, we use a gesture recognition engine, which is based on K-means and Hidden Markov Models (HMMs). The performance of the machine learning algorithm has been tested with professional gestures using a silk-weaving, a TV-assembly and hand-made glass datasets.

Keywords

Pose estimation, depth map, gesture recognition, Hidden Markov Models, K-means, smartphone

Acknowledgements

I would like to express my gratitude to Sotiris, Alina and the entire team of the Ecole de Mines for the incredible opportunity they have given me and the excellent work environment they have generated. Thanks to my friends, Los Habitantes Del Zulo, for the experiences of these two unforgettable years, Trostris, for a lifelong friendship, and Telecos, for a friendship worth keeping. Thanks to Mercedes, the most special person in my life. Finally, I would also like to thank my family for always being there against all odds.

Pablo Vicente Monivar
July 2019.

Contents

Resumen **iii**

Abstract **v**

Acknowledgements **vii**

1 Introduction **1**

1.1 Motivation . . . 1
1.2 Objectives . . . 1
1.3 Document Structure . . . 2

2 State of the Art **3**

2.1 Introduction . . . 3
2.2 Capturing RGB-D frames . . . 3
2.2.1 iPhone XS camera . . . 4
2.2.2 Intel RealSense Depth Camera D435 . . . 5
2.2.3 Microsoft's Xbox Kinect . . . 6
2.3 Pose Estimation . . . 6
2.3.1 Pose Estimation Frameworks . . . 6
2.4 Machine and deep learning frameworks for mobile devices . . . 9
2.5 Communication between smartphone and server (WebSocket) . . . 9
2.6 Gesture recognition methods . . . 10
2.6.1 Dynamic Time Warping (DTW) . . . 10
2.6.2 Hidden Markov Models (HMMs) . . . 11
2.7 Evaluation framework and metrics . . . 12
2.8 Conclusions . . . 13

3 Design and development of the application **15**

3.1 Introduction . . . 15
3.2 Video recording using the RealSense depth camera . . . 15
3.3 Video recording using the smartphone . . . 16
3.4 Communication with pose estimation server . . . 17
3.5 Pose estimation . . . 18
3.6 Gesture recognition . . . 19
3.7 Smartphone application . . . 19

4 Dataset **21**
4.1 TV Assembly dataset 21
4.2 Cerfav dataset 21
4.3 Silk Weaving dataset 22

5 Evaluation **25**
5.1 Potential challenges and limitations 25
5.2 Comparative Evaluation 27
5.2.1 Pose estimation comparison using the TVA, SW and Cerfav datasets 27
5.2.2 Gesture recognition comparisons using the TVA dataset: 2D vs 3D and 2 vs 7 joints 28
5.2.3 Gesture recognition comparison using three different camera positions from the SW dataset 29
5.2.4 Comparison mixing gestures and data from the TVA and the SW datasets 29
5.2.5 Gesture recognition comparisons using the Cerfav dataset: 2D vs 3D 30
5.3 Conclusion 31

6 Conclusion and future work **33**
6.1 Conclusion 33
6.2 Future work 33

Bibliography **35**

List of Figures

2.1 IPhone XS back camera (Dual Rear Camera) 4
2.2 IPhone XS front camera (True Depth Camera) capturing depth 5
2.3 Architecture of Intel RealSense Depth Camera D435 5
2.5 Example of WebSocket communication 11

3.1 Architecture of the overall pipeline 15
3.2 AVFoundation pipeline followed by the application 17
3.3 Example of RGB-D frames from TV Assembly and Cerfav dataset . . 18
3.4 Gesture recognition pipeline followed by the application 19
3.5 Example of the smartphone application 20

5.1 Example of skeleton estimation in Silk Weaving; TV Assembly and Cerfav dataset 26

List of Tables

2.1 List of popular open-source frameworks for 2D pose estimation. mAP is Mean Average Precision . . . 9
2.3 Example of confusion matrix . . . 12

4.1 Example of frames from the conveyor surface dataset . . . 21
4.2 Example of frames from the Cerfav dataset . . . 22
4.3 Example of frames from the silk weaver museum dataset . . . 23

5.1 OpenPose results on the TVA and SW datasets . . . 28
5.2 Comparison in the gesture recognition using different joints and dimension and TV Assembly dataset . . . 29
5.3 Gesture recognition using all joints and 2 dimensions on Silk Weaving dataset . . . 29
5.4 Gesture recognition mixing gestures from SW and TVA datasets and using 7 joints and 2D . . . 30
5.5 Comparison in the gesture recognition using different joints and dimension and TV Assembly dataset . . . 31

Chapter 1

Introduction

1.1 Motivation

The role of professional actions, activities and gestures is of high importance in most industries. Motion sensing and machine learning have actively contributed to the capturing of gestures and the recognition of meaningful movement patterns by machines. Therefore, very interesting applications have emerged according to the industry. For example, in the factories of the future, the capabilities of workers will be augmented by machines that can continuously recognize their gestures and collaborate accordingly, whereas in the creative and cultural industries it is still a challenge to recognize and identify the motor skills of a given expert. Therefore, capturing the motion of workers or craftsmen using off-the-shelf devices, such as smartphones, has a great value. New smartphones are equipped with depth sensors and high power processors, which allow us to record data even without very sophisticated devices.

The motivation of this work is to give the possibility to the users to easily record, annotate, train and recognize human, actions, activities and gestures in professional environments.

1.2 Objectives

The general scope of this work is to build the basis of an application for a mobile device, which allows for data recording using its embedded sensors, estimate the human pose, extract the skeleton and recognize (offline) professional gestures. The pose estimation depends on the camera: RGB for 2D, and RGB-D, either infrared or stereoscopic, for 3D skeletons. The whole process can be controlled by the application. More precisely, the annotation of gestures, the selection of body joints, the skeleton visualization and the projection of the recognition results run locally on the phone while the pose estimation and machine learning algorithms are delegated to a GPU server.

1.3 Document Structure

The structure of the document is as follows:

- Chapter 1: This chapter summarizes the motivation and the objectives pursued in this master thesis, as well as an introduction of the topic developed in each chapter.
- Chapter 2: This chapter explains the concepts of pose estimation and gesture recognition, making an analysis and comparison of the most popular frameworks. Moreover, it analyzes different devices of RGB-D data capture and an overview of the Machine and Deep Learning frameworks for mobile devices. In addition, there is a brief explanation of the client-server communication and the metrics used in this work.
- Chapter 3: This chapter presents the overall pipeline followed to create the smartphone application. From the capture of RGB-D frames with the iPhone or the RealSense camera, to the recognition of gestures and estimation of skeleton carried out by the GPU server, explaining the communication client - server (smartphone - GPU) through WebSockets.
- Chapter 4: This chapter evaluates the accuracy and performance of different modules of the application, comparing the results in 2D against 3D, and including the pose estimation and the gesture recognition.
- Chapter 5: This final chapter exposes the conclusions obtained from this master thesis as well as the possible improvements and future work.
- References.

Chapter 2

State of the Art

2.1 Introduction

Nowadays, the majority of people have in their pocket a smartphone device that over the years, allows to carry out more sophisticated tasks. Improvements in acquisition methods, with the addition of stereoscopic and infrared cameras, as well as improvements in processing and analysis methods, embedding processors and graphics cards of the latest generation, have allowed the computer vision science to understand and develop applications based on deep learning for mobile phones.

Both pose estimation and gesture recognition are two booming computer vision techniques. Being able to estimate the human body and its skeleton provides a potential source of applications such as prevention of non-ergonomic postures, improvement of sports performance and, moreover, a potential source of human body data that can be exploited by other computer vision techniques such as gesture recognition.

In this chapter, we first explain different ways of capturing RGB and depth, and their respectivew cameras. Secondly, we do an analysis of some popular pose estimation frameworks as well as related works implemented in smartphones. After that, we define how we can transfer these data into a communication client-server. Finally, we explain the most common methods for gesture recognition as well as the metrics used to evaluate the performance of the application.

2.2 Capturing RGB-D frames

The current market offers a wide variety of devices capable of capturing RGB and depth frames (RGB-D frames), however, in this work, we have focused on three different devices: iPhone XS True Depth camera and Dual Rear camera, Intel RealSense

Depth Camera D435 and Microsoft's Xbox Kinect camera.

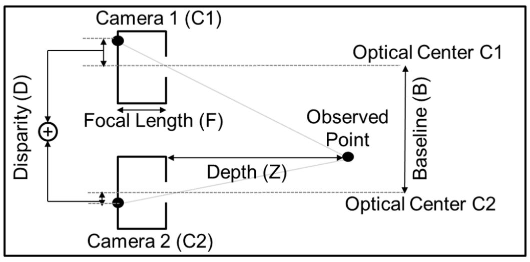
2.2.1 iPhone XS camera

Since the launch of iPhone X in 2017, the new Apple phones have a depth mapping system in their rear and front camera.

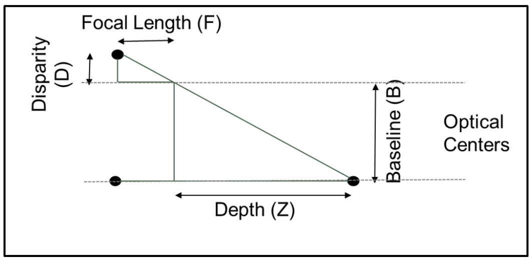
On one hand, iPhone XS uses a dual rear camera, composed by a wide-angle lens and a telephoto lens capturing data with the same frame rate, to obtain the disparity, defined as the displacement in the position for corresponding points between the images. The main feature of the camera is that it is stereo rectified, which means that both cameras are pointing in the same direction and have the same focal length, distance from the focal point to the image plane and, in addition, the distance between the two optical lenses refers to the baseline.

In the Figure 2.1a is shown how the rear camera captures the normalized disparity which mathematically equals to calculate the inverse of the depth (1/meters). First, the rays of lights from the observed object pass through the optical centers and are reflected on the image plane of each camera. Then, the mathematical relation tying the depth (Z), baseline distance (B), disparity (D) and focal length (F), showed in equation 2.1, results in the normalized disparity. Finally, the iPhone needs to filter and post-process the disparity to smooth the edges and fill the holes, which requires a heavy computation. [1]

$$\frac{B}{Z} = \frac{D}{F} \rightarrow \frac{1}{Z} = \frac{D}{BF} \tag{2.1}$$



(a) IPhone XS back camera (Dual Rear Camera) capturing disparity



(b) Capturing disparity when the observed object is aligned in the optical center C2

Figure 2.1: IPhone XS back camera (Dual Rear Camera)

On the other hand, the front camera, named True Depth Camera by Apple, is used to measure the depth in meters directly. It has a dot projector that launches over 30,000 dots onto the scene, generally the user face, which are then captured by an infrared camera. To ensure that the system works properly in the dark, there

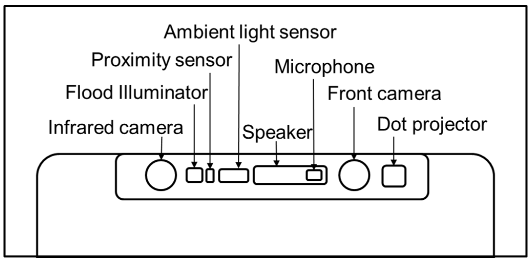


Figure 2.2: IPhone XS front camera (True Depth Camera) capturing depth

is an ambient light sensor and a flood illuminator which adds more infrared light when needed. The final result is more stable depth images with a higher resolution, 640x480 instead of 320x240 obtained by the dual rear camera. Figure 2.1b shows the architecture of the True Depth Camera.

Finally, both cameras capture RGB-D frames with a frame rate of 30 fps and by using a portrait mode, which means subtracting the foreground, object that is usually a person who is focused, from the background.

2.2.2 Intel RealSense Depth Camera D435

The Intel RealSense Depth Camera D435 [2] is an USB-powered camera that includes depth sensors and a RGB sensor. It uses stereo vision to calculate depth and its implementation is based on a left imager, right imager, that capture depth in a similar way to the dual rear camera of the iPhone, and an optional infrared projector to improve depth accuracy by projecting a static infrared pattern on the scene to increase texture on low texture scenes. Moreover, it has a RGB module (Color camera) to color frames providing texture data, with whom we can make an overlaying on the depth to create a color point cloud and a 3D reconstruction. The resolution is up to 1280 x 720 and the depth stream output frame rate is up to 90 fps.

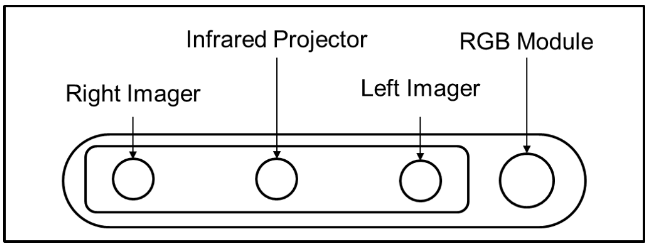


Figure 2.3: Architecture of Intel RealSense Depth Camera D435

2.2.3 Microsoft's Xbox Kinect

Kinect [3] is a Microsoft motion sensor add-on for the Xbox 360 gaming console. It is composed of three main hardware pieces: a Color VGA video camera, a depth sensor, and a multi-array microphone.

The VGA camera capture the RGB color space as well as body-type and facial features. Its resolution is 640x480 pixels and has a frame rate of 30 fps. The depth sensor contains a monochrome CMOS sensor and infrared projector that measures the distance of each point of the body of the player by transmitting invisible near-infrared light and measuring its "time of flight" after it reflects off the objects. Finally, it has a multi-array microphone that can isolate the voices of the player from other background noises.

2.3 Pose Estimation

Due to the growing need to understand and predict human behavior and motor skills, challenges such as human pose estimation become crucial in the field of Machine Learning. Human pose estimation can be defined as the problem of locating and representing in a coordinate system, point cloud, the set of keypoints or joints that shape a person into an image or video.

Pose estimation in multi-person scenarios is generally carried out by using a top-down approach or a bottom-up approach. In the first approach, a human detector is initiated and both, the joints and the skeleton of each person, are calculated separately. This approach make uses of existing techniques for single-person pose estimation. However top-down approach suffers from an early commitment when the detector fails, and the computational power increases exponentially with the number of people in the scene.

In contrast, bottom-up approach firstly detect and label all the joints candidates in the frame and secondly associate them to each individual person without using any person detector. It is, generally, a more complex approach, but is more robust to occlusion and complex poses.

2.3.1 Pose Estimation Frameworks

Different benchmarks of 2D human pose estimation are evaluated through annual challenges that aim to improve various key-parameters, such as the accuracy, how the algorithm performs with partial occlusions or using different keypoints, how to detect the pose of big number of individuals in the scene, etc. Some popular examples are the COCO Keypoint, MPII HumanPose and Posetrack challenges. Below, we will review and compare some of the popular top-down and bottom-up approaches for pose estimation.

2.3.1.1 AlphaPose

AlphaPose [4] is a a top-down method based on regional multi-person pose estimation (RMPE). It follows a pipeline in which it is first applied the object detector Faster-RCNN[5] to obtain the human bounding boxes. This bounding box will fed into a Spatial Transformer Network (STN), which select the region of interest, then, a parallel single-person pose estimator (SPPE) module, to improve robustness against imperfect human bounding boxes, and finally, a Spatial Detransformer Network (SDTN), which generates pose proposals. At the end, a Parametric Pose Non-Maximum-Suppression (NMS) is carried out to eliminate redundant pose estimations

2.3.1.2 DensePose

DensePose [6], developed by Facebook AI research, takes as input an RGB image and estimates the surface-based description of the human body. This framework starts by applying an adapt version of Mask-RCNN with a Feature Pyramid Network (FPN) [7] features, FPN brings robustness to CNN in detection at different scales, to predict the discrete part labels and continuous surface coordinates. The reconstruction of the surfaces is carried out by an inpainting process based on another convolutional neural network, this process allows to recover deteriorated part of the image or body parts that are hidden and is relies on the estimation performed at different scales by the FPN.

2.3.1.3 Human Mesh Recovery

Human Mesh Recovery (HMR) [8] framework aims to map each person and extract a 3D joint surface from the shape and the angles of the human body by only using RGB images. The framework takes an input image centered on a human in a feed-forward manner. Then, a convolutional encoder, bases on Skinned Multi-Person Linear (SMPL) [9] model, generates the features that will feed an iterative 3D regression module whose objective is to infer the 3D reconstruction of the human body. After that, HMR uses a discriminator network which select the meshes that belong to the real human.

2.3.1.4 DeepCut

DeepCut[10] is an example of bottom-up approach, it uses an adapt Fast R-CNN version called AFR-CNN to extract a set of joint candidates. Then, it uses VGG for extracting part probability score maps and, finally, associates the joints with Non-Maximum Suppression.

2.3.1.5 OpenPose

OpenPose [11] is a real-time bottom-up approach for detecting 2D pose of multiple people on an image. First, it takes an RGB image (Figure 2.4a) and apply a fine-tuned version of the convolutional neural network VGG-19 [12] to generate the input features of the algorithm. Second, these features enter a multi-stage CNN, created on the basis of the convolutional pose machines developed in [13], to predict the set of confidence maps (Figure 2.4b), where each map represents a joint, and the set of part affinities (Figure 2.4c), which represent the degree of association between joints. Lastly, after a non-maximum suppression in order to discretize the data, bipartite matching is used to associate body part candidates (Figure 2.4d) and obtain the full 2D skeletons (Figure 2.4e), i.e. the joint that share the higher weight are selected between all the possible pairs of associations.

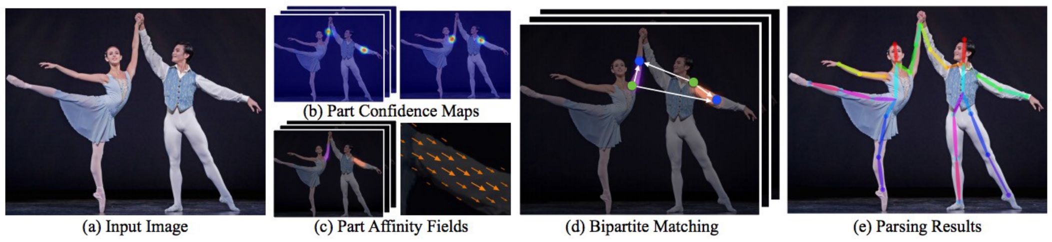


Figure 2.4: OpenPose pipeline Reprinted from [11]

2.3.1.6 Comparison between the frameworks

In our work, we have chosen to use OpenPose. The main reason is that OpenPose is a bottom-up approach, we avoid a possible early commitment, that includes a hand skeleton estimation, which has been considered an important perspective in our system. Table 2.1 shows a list of popular open source methods for 2D pose estimation, their Githubs, the machine learning framework used for the implementation and the classification obtained in their respective challenges.

Table 2.1: List of popular open-source frameworks for 2D pose estimation. mAP is Mean Average Precision

Method	Git.	ML framework	Benchmark / Dataset	Rank	mAP (%)
AlphaPose	[14]	PyTorch	COCO Keypoint challenge 2018	11	70.2
DensePose	[15]	Caffe2	Posetrack multi-person pose estimation 2017	7	61.2
HMR	[16]	TensorFlow	Common objects in context (COCO)	–	–
DeepCut	[17]	Caffe	MPII Multi-Person dataset	–	51.4
OpenPose	[18]	Caffe	COCO Keypoint challenge 2016	1	60.5

2.4 Machine and deep learning frameworks for mobile devices

A pose estimation embedded to a smartphone still represents a huge challenge for the scientific community. Neither Apple Store nor Google Play propose any application that provides pose estimation. The main reason is the computational power needed for running deep learning algorithms, as well as the lack of powerful graphical devices in smartphones. [19] shows different tests when running deep neural networks on Android smartphones. The use of TensorFlow-Lite, Caffe-Android or Torch-Android frameworks is currently possible for implementing CNNs on smartphones.

Nevertheless, at the end of 2017, Apple made a transition in the world of machine learning by launching the CoreML framework for iOS 11 that enable the running of machine learning models on mobile devices. The performance improved in the next version, CoreML2, at the end of 2018.

Today, there are available applications that do eye tracking based on gaze estimation by using CNNs [20]. Nevertheless, pose estimation requires much more computation power than eye tracking or face recognition. For this reason, the use of an external framework has been considered as a better solution within the context of this work.

2.5 Communication between smartphone and server (WebSocket)

In any web application, client-server communication is one of the most important aspects. The data communication must be as fast and smooth as possible, avoiding losses and interruptions and enabling an excellent user experience

The WebSocket Protocol (RFC 6455) [21], launched in 2011, enables a bidirectional communication and full-duplex over a single TCP socket, and between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code.

The connection is established on HTTP and begins with a handshaking process, when one device, the client, sends a WebSocket negotiation request, and the other device, the server, sends a WebSocket negotiation response. The handshake is shown in Table 2.2

Table 2.2: WebSocket handshake example [21]

Handshake from the client	**Handshake from the server**
GET /chat HTTP/1.1 Host: server.example.com Upgrade: websocket Connection: Upgrade Sec-WebSocket-Key: XXY= Origin: http://example.com Sec-WebSocket-Protocol: chat, superchat Sec-WebSocket-Version: 13	HTTP/1.1 101 Switching Protocols Upgrade: websocket Connection: Upgrade Sec-WebSocket-Accept: XXZ== Sec-WebSocket-Protocol: chat

Once established, WebSocket data frames can begin to be sent in both directions between the client and the server in full-duplex mode. In this way, a server can send any message to a client at any time and the client would receive it instantly, and the same in the opposite direction: a message we generate in the browser can be sent without having to establish a new connection because there is already one open.

Finally, another handshaking communication is needed to close the channel. Figure 2.5 shows an example of WebSocket architecture.

2.6 Gesture recognition methods

The implementation of deep learning for gesture recognition has become the common practice and can lead to very good results. The ChaLearn LAP Large-scale Isolated Gesture Recognition Challenge from the ICCV 2017, crowned [22] [23] [24] as the best deep learning algorithms for gesture recognition. However, the need for large training databases is not compatible with the constraints professional gestures where datasets are quite small. Therefore, in this research we will study others gesture recognition methods.

2.6.1 Dynamic Time Warping (DTW)

Dynamic Time Warping (DTW) [25] as well as Hidden Markov Models [26] are two machine learning methods widely used in pattern recognition. DTW is a template-based approach that is based on a temporal re-scaling of a reference motion signal and

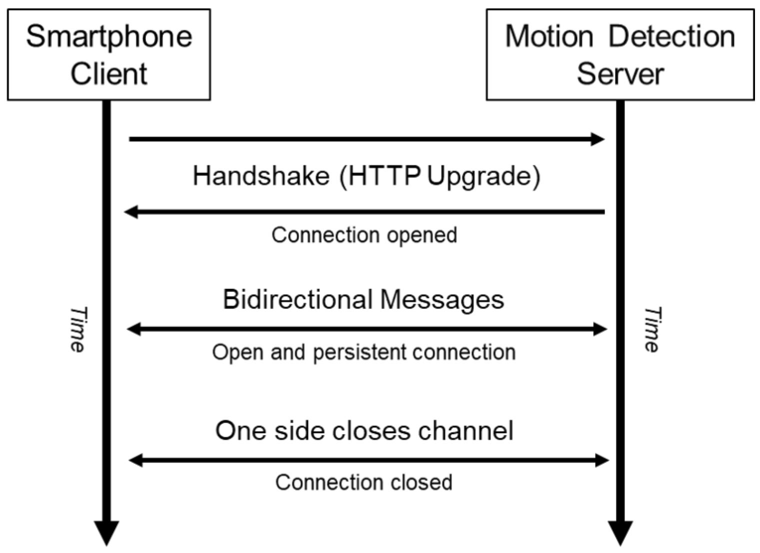


Figure 2.5: Example of WebSocket communication

its comparison with the input motion signal, i.e, it measures the similarity between two temporal sequences, such as in [27] where they use DTW for off-line recognition of a gestures. DTW is good for doing one-shot learning, because can detect similarity between two temporal sequences even that one is faster than the other, while HMMs is a robust duration-independent model-based approach.

2.6.2 Hidden Markov Models (HMMs)

Hidden Markov Models is a statistical model for modelling time series data based on the Markov chain or Markov property, i.e. each event depends only on the previous event ($P(X_t = j|X_1 = i_1, ..., X_{t-1} = i_{t-1}) = P(X_t = j|X_{t-1} = i_{t-1})$). In HMM, we make two assumptions: first, the observation at time t comes from a hidden process state (S) and second, it satisfies the Markov property. Therefore, joining these two assumptions, with S_t the current state and Y_t the observed variable, we obtain:

$$P(S_{1:T}, Y1 : T) = P(S_1, Y_1) \prod_{T}^{t=2} P(S_t|S_{t-1})P(Y_t|S_t) \tag{2.2}$$

We chose to use the work described in [28], which makes use of K-means to model the time series of motion data and HMMs for classifying and recognizing the gesture classes by using the Gesture Recognition Toolkit (GRT)[29].

2.7 Evaluation framework and metrics

In this section, we will explain the metrics used in Chapter 5 for evaluating the application.

- Confusion matrix: allows the visualization of the performance of a supervised learning algorithm relating the actual gesture to the predicted gesture. Table 2.3 shows an example of a confusion matrix.

		Actual Gesture	
		Gesture X	Gesture Y
Predicted Gesture (HMM)	HMM X	True Positive X	False Negatives Y False Positive X
	HMM Y	False Negatives X False Positive Y	True Positive Y

Table 2.3: Example of confusion matrix

- Recall: in this research we define recall as the percentage of total gestures performed and correctly classified by the algorithm. Equation 2.3 shows the general and the applied equation of the recall measure.

$$Recall(Rc) = \frac{True\,Positive}{True\,Positive + False\,Negatives} = \frac{\#(gestures\,correctly\,recognized)}{\#(gestures\,performed)} \tag{2.3}$$

- Precision: in this research we define precision as the percentage of total gestures performed and correctly recognized by the algorithm. Equation 2.4 shows the general and the applied equation of the precision measure.

$$Precision(Pr) = \frac{True\,Positive}{True\,Positive + False\,Positive} = \frac{\#(gestures\,correctly\,recognized)}{\#(gestures\,classified)} \tag{2.4}$$

- Accuracy: we define accuracy as the percentage of total gestures performed, correctly recognized and correctly classified by the algorithm. Equation 2.5 shows the general and the applied equation of the accuracy measure.

$$Accuracy(Ac) = \frac{True\,Positive + True\,Negative}{Total} = \frac{\#(gestures\,correctly\,recognized)}{\#(gestures)} \tag{2.5}$$

- F-score: defines the harmonic mean between the precision and the recall. Equation 2.6 shows the general equation of the f-score measure.

$$F\,score = 2\,\frac{Pr \times Rc}{Pr + Rc} \tag{2.6}$$

2.8 Conclusions

In the state of the art has been explained the possible pose estimation frameworks, devices or protocols that allows to create the target application. To conclude this section and to have an idea of the approach followed in this research, our application uses the iPhone XS for capturing frames RGB-D given their stereoscopic and infrared camera, the pose estimation is carried out by OpenPose thanks to its hands estimation module and its bottom-up architecture, k-means and HMM recognize gestures in the absence of a well annotated dataset and, finally, the communication client-server is performed by using WebSockets.

Chapter 3

Design and development of the application

3.1 Introduction

The smartphone handles the input and output steps of the pipeline. It records the RGB-D frames and shows the results (body skeleton and gesture recognition accuracy). An external motion detection server receives the frames by using websockets and, then, it estimates the skeleton and uses the information provided by the body joints to train and test a gesture recognition engine. The overall architecture is shown in Figure 3.1.

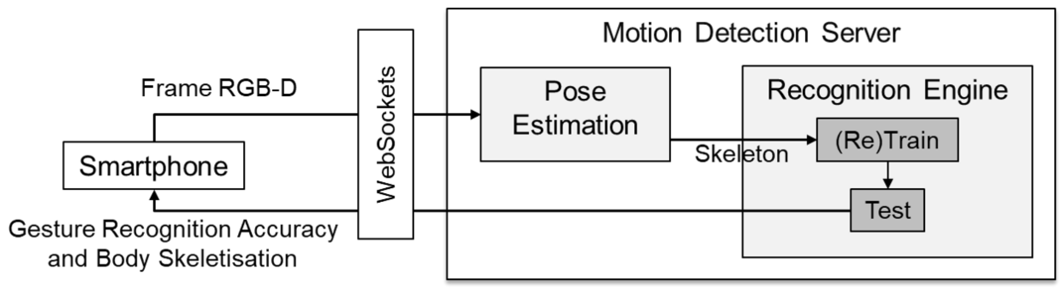


Figure 3.1: Architecture of the overall pipeline

3.2 Video recording using the RealSense depth camera

Although the goal of this research work is to make a smartphone application. Some of the application modules, such as pose estimation or gesture recognition modules, can be tested with a dataset similar to what a smartphone camera can provide. Therefore, the Intel RealSense Depth Camera D435 has been used for some of the tests, as well as for the creation of the first datasets.

Firstly, for working with the RealSense camera, it is necessary to have installed their software development kit (SDK) and the Intel RealSense viewer, interface that allows to select the camera presets and record the RGB-D frames.

Secondly, the RealSense camera saves the sequences in an unique ROS bag file extension, format created by the Robot Operating System (ROS). Therefore, it is necessary to perform a conversion to a format readable by the pose estimator (png or jpg). This conversion is implemented on the basis of the file conversion platform Intel rs-convert Tool.

Finally, after the conversion, we got RGB-D sequences with a resolution of 640x480 pixels and a frame rate of 30fps, the same ones used by the iPhone XS, with which we can train and test the gestures recognition module of the application.

3.3 Video recording using the smartphone

A specific module for depth recording has been developed in order to use any iOS device equipped with an RGB and/or RGB-D sensor to record data. More precisely, the new iPhone XS has been used in this work.

In order to capture the frames with both cameras, it is necessary to use the AV-Foundation framework for working with temporal audiovisual media. This framework is shown on Figure 3.2 and works through the creation of a session with at least one capture input (camera or microphone) and one capture output (photo, movie or audio). In this work, we have created a session with two inputs: the frontal camera (True Depth Camera) and the rear camera (Dual Rear Camera) of the iPhone, and two outputs: RGB data (Video Data Output) and depth data (Depth Data Output). In addition, we have defined the capture setting presets to 640x480 pixel and 30fps, similar to the one used by the RealSense. A very important task in the capturing pipeline has been creating a capture queue in order to coordinates time-matched delivery of data from multiple capture outputs (RGB and depth).

Depth data is captured in the form of depth (meters), which is the distance from the camera to the operator, or disparity (1/meters). The possible pixel format is Float16 (16-bit) or Float32 (32-bit), but since we want to use the data for training a Hidden Markov Model we select Float16 format, Float32 will increase considerably the training time. Finally, it is necessary to carry out a conversion from distances into pixel values in order to visualize depth maps. In Figure 3.3 some examples of depth maps are represented along with their RGB frames. In the Cerfav dataset frame we

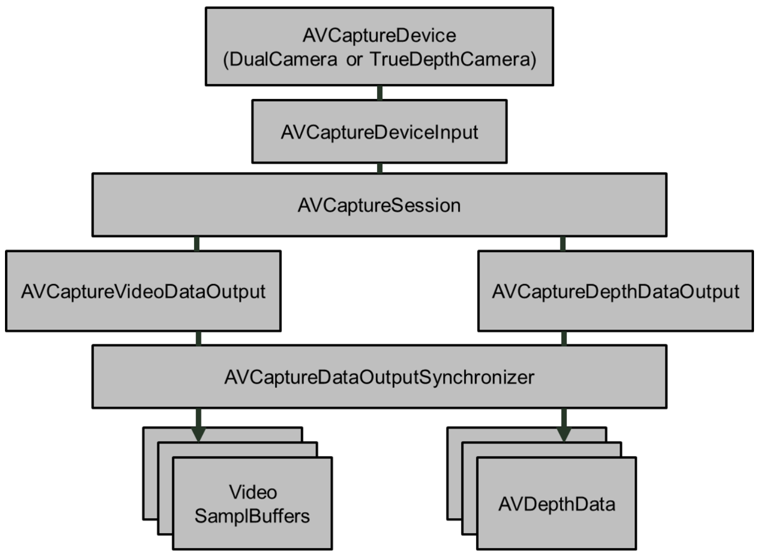


Figure 3.2: AVFoundation pipeline followed by the application

can see the effect of the portrait mode, where all the background has been subtracted and appears in black.

3.4 Communication with pose estimation server

Once the data has been recorded, it is sent to the GPU server by using WebSockets (RFC 6455). For compression purposes, before sending the RGB-D data, it is converted to jpeg format.

We sent the frames by using the WebSocket client library Starscream. The first step is to create the connection between the client (the iPhone) and the server (the GPU). The smartphone makes a connection request by defining the ip address of the server, the port and the protocol, in our case the echo-protocol. If the request is satisfactory the bidirectional channel of communication is opened. Then, we start sending frame by frame all the RGB-D data captured.

However, the depth information is large and difficult to compress, even if we are using jpeg format this still using a lot of bandwidth, jpeg gives the advantage of decreasing the latency and recovering each frame with high quality while using H.264 gives better compression but at the cost of latency. Therefore, this application module is still in an initial phase.

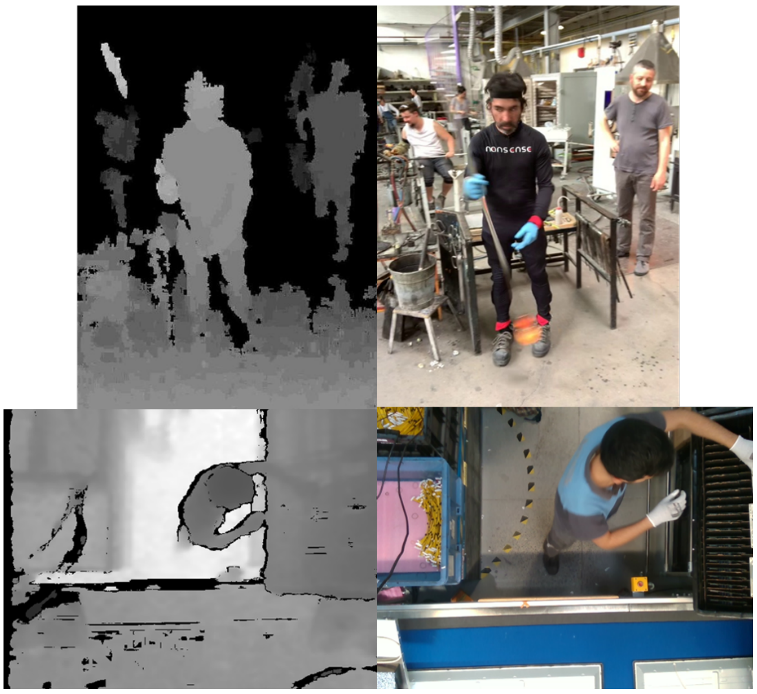


Figure 3.3: Example of RGB-D frames from TV Assembly and Cerfav dataset

3.5 Pose estimation

The goal of pose estimation is to obtain a series of keypoints that can be used by a gesture recognition engine as input, enabling to train a model that adapts to different situations or environments. OpenPose, in our case, estimates 25 body keypoints, 2x21 hand keypoints and 70 face keypoints. However, some of the estimated keypoints are useless for the recognition, either they are occluded or they do not carry any information about the gesture. Therefore, a joint selection module has been added to the application to give the possibility to the user to select the most appropriate keypoints depending on the use-case.

Once the joints of interest have been selected, a 3D model of the skeleton is created. In order to do this, a weighting is made between the values of the pixels obtained by OpenPose and the depth map, it means, if OpenPose returns the coordinate value x_{J_n} and y_{J_n} for the joint J_n with the depth map Z, we will take as depth value, z_{J_n}, the result of equation 3.5

$$z_{1J_n} = Z[floor(x_{J_n})][floor(y_{J_n})] \times (ceil(x_{J_n}) - x_{J_n}) \times (ceil(y_{J_n}) - y_{J_n}) \tag{3.1}$$

$$z_{2J_n} = Z[floor(x_{J_n})][ceil(y_{J_n})] \times (ceil(x_{J_n}) - x_{J_n}) \times (y_{J_n} - floor(y_{J_n})) \tag{3.2}$$

$$z_{3J_n} = Z[floor(x_{J_n})][floor(y_{J_n})] \times (x_{J_n} - floor(x_{J_n})) \times (ceil(y_{J_n}) - y_{J_n}) \quad (3.3)$$

$$z_{4J_n} = Z[floor(x_{J_n})][floor(y_{J_n})] \times (ceil(x_{J_n}) - x_{J_n}) \times (y_{J_n} - floor(y_{J_n})) \quad (3.4)$$

$$z_{J_n} = z_{1J_n} + z_{2J_n} + z_{3J_n} + z_{4J_n} \quad (3.5)$$

3.6 Gesture recognition

The joints obtained with the pose estimation and the data obtained from the depth camera are the input to the gesture classification algorithm. To make the recognition invariant to the position of each person in the frame, the neck joint has been taken as a reference point, and any frame without neck has been discarded.

The gesture recognition engine is based on supervised learning. Therefore, before making the gesture recognition, a labelled database has been manually created by manually selecting starting and ending time stamps of each gesture.

Once the database has been labelled, it is necessary to process it in order to organize it into logical groups. A .grt document has been created by concatenating in rows all the coordinates, 2D or 3D, of the different joints and the different frames of the each gesture. Moreover, we normalize the pixel values between 0 and 1.

Finally, the gesture recognition engine uses k-Means to obtain discrete-valued observations. Then, Hidden Markov Models is used to train the discrete data and to determine a gesture recognition accuracy. The platform GRT has been used for the entire process. Figure 3.4 shows the gesture recognition pipeline.

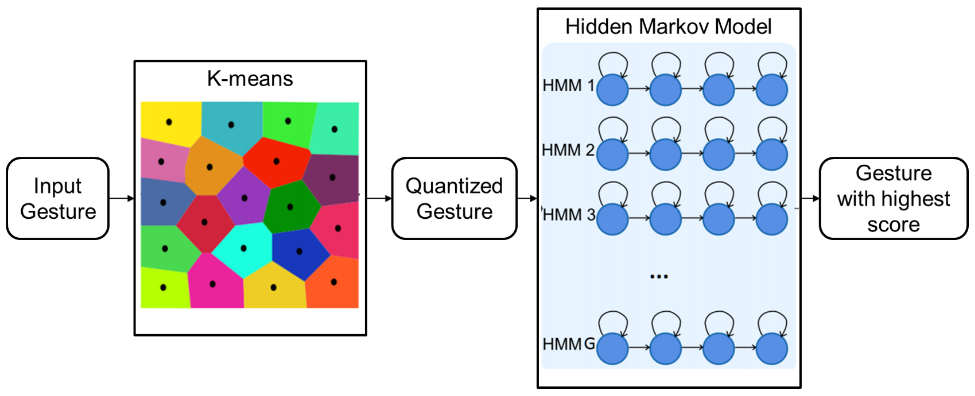
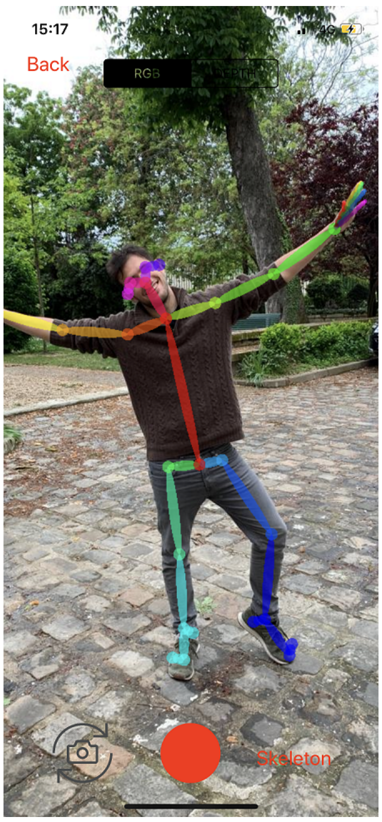


Figure 3.4: Gesture recognition pipeline followed by the application

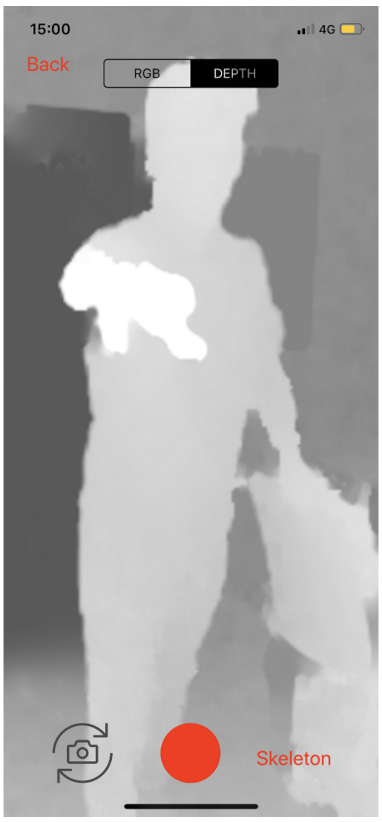
3.7 Smartphone application

The graphical user interface of the application consists on four main modules:

- Recording module: allows to record the sequences and visualize depth maps live. This application module is fully implemented and functional. An example of this module is shown on Figure 3.5a.
- Skeleton Visualization module: allows to visualize the skeleton estimated. This application module is in the process of implementation. Figure 3.5b shows how this module will look like.
- Training module: allows to select the appropriate joints for the gesture recognition. This module is fully operational.
- Labelling module: interface similar to a photo gallery where you must select the gestures that you want to recognize. This module is pending implementation.



(a) Skeleton visualization module using the rear camera



(b) Recording module, enabling depth, using the front camera

Figure 3.5: Example of the smartphone application

Chapter 4

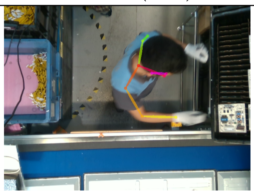
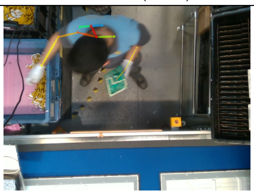
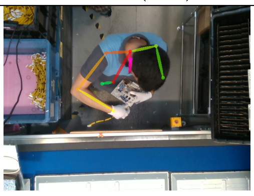
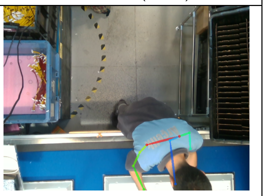
Dataset

4.1 TV Assembly dataset

The first dataset used has been named as TV Assembly dataset (TVA). It is made up of RGB-D sequences recorded from a top mounted view at a conveyor surface factory. Each sequence contains around 10.000 RGB frames together with the depth, all the frames have a resolution of 640x480 and a frame rate of 30fps.

Two different users have been recorded using the Intel RealSense Depth Camera D435. Table 4.1 shows the four gestures identified and labeled during the sequences along with the skeleton estimated of each of them.

Table 4.1: Example of frames from the conveyor surface dataset

Gesture 1 (G1)	Gesture 2 (G2)	Gesture 3 (G3)	Gesture 4 (G4)
			
Take the card from the left side box	Take the wire from the right side box	Connect the wire with the card	Place the card on the TV chassis

4.2 Cerfav dataset

The second dataset, Cerfav dataset, has been recorded in a center for researches and training in the glass-work. It has been recorded using the iPhone XS frontal camera, recording from different points of view and with motion. It consists of RGB-D frames with a resolution of 480x640.

Table 4.2: Example of frames from the Cerfav dataset

Gesture 1 (G1)	Gesture 2 (G2)	Gesture 3 (G3))
Insert the glass into the furnace	Move the bar from one side to the other	Shaping glass with the hand
Gesture 4 (G4)	**Gesture 5 (G5)**	**Gesture 6 (G6)**
Blow through the stick	Tighten the base of the glass with the pliers	Burn the base of the glass with a torch

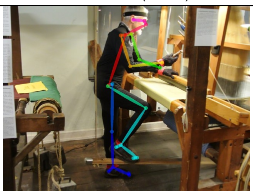
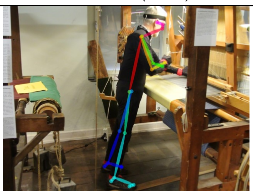
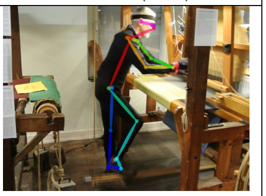
The overall task performed by the worker of this factory is longer than the previous dataset tasks and is composed of six different gestures. Table 4.2 shows the six gestures identified and labeled during the sequences along with the skeleton of each of them.

4.3 Silk Weaving dataset

Finally, the third dataset, Silk Weaving dataset (SW), contains sequences recorded from a lateral view and three different positions in a silk weaver museum. Each sequence contains around 6.000 RGB frames with a resolution of 640x480 and a frame rate of 30fps.

In SW dataset only one user has been recorded. Table 4.3 shows the three gestures identified and labeled during the sequences along with the skeleton of each of them.

Table 4.3: Example of frames from the silk weaver museum dataset

Gesture 1 (G5)	Gesture 2 (G6)	Gesture 3 (G7)
		
Press the treadle and push the batten	Move the shuttle sideways	Leave the treadle and pull the batten

Chapter 5

Evaluation

5.1 Potential challenges and limitations

In this application, the gesture recognition engine receives as input the joints estimated by OpenPose, therefore, the recognition of gestures highly depends on the quality of the pose estimation. In this section, we will analyze some of the most common problems we encountered during the pose estimation:

- Skeleton not estimated: the pose estimation algorithm does not detect any person in the frame and therefore does not return any joint. This situation usually occurs when the person continues performing their tasks outside the reach of the camera, or when the person is to blurred for being recognize.

- Skeleton incorrectly estimated: in this case, the algorithm detects and estimates the person in the image, however, the estimation is not correct. This a high influential error since the generated input is erroneous and, therefore, the Hidden Markov Model will use erroneous estimation for training.

- Partial skeleton estimation: this is the case where, either because the image is too blurry, or because the person is not detected correctly, the estimated skeleton is incomplete, i.e. not all joints are estimated.

- Skeleton occluded: in line with the previous case, if part of the human body is occluded, either by an object, or because one part of the body occludes the other, the skeleton will not be completely estimated.

- Wrong person estimated: as OpenPose is a multiple people estimation framework, it may happen that the person interested in being estimated in the framework is not the person with the highest score, being this, the discrete method to track the target person. This error together with skeleton incorrectly estimated

In figure 5.1 we can see some examples of skeleton estimation in the three different datasets. From left to right and from top to bottom we are going to explain the type of estimation in that frame: (1) Frame with skeleton not estimated. (2) Frame with skeleton incorrectly estimated. (3) Frame with skeleton occluded. (4) Frame with partial skeleton estimation. (5) Frame perfectly estimated. (6) Frame perfectly estimated. (7) Frame perfectly estimated. (8) Frame with skeleton occluded. (9) Frame with the wrong person estimated. (10) Frame with skeleton occluded.

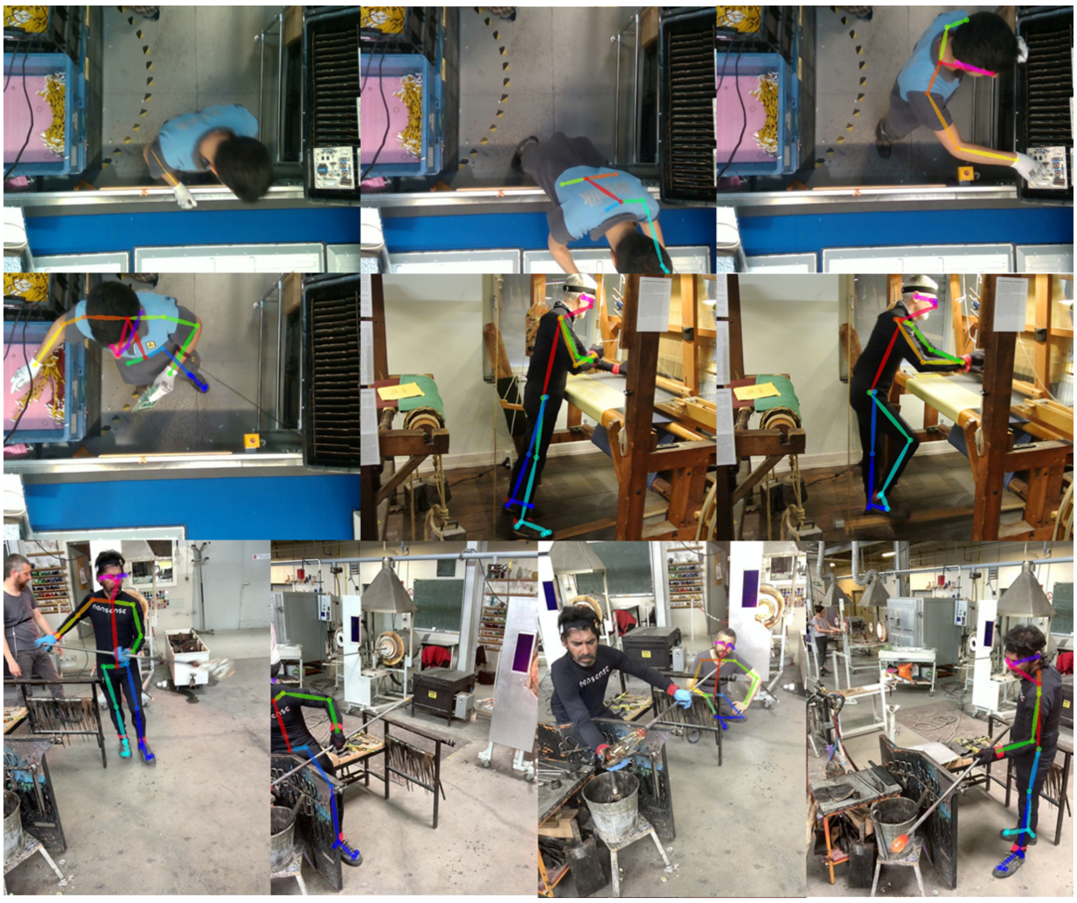


Figure 5.1: Example of skeleton estimation in Silk Weaving; TV Assembly and Cerfav dataset

5.2 Comparative Evaluation

In order to evaluate the performance of our pipeline and the algorithms we use the 80%-20% evaluation criteria. We randomly divide our dataset in 80% training set and 20% as testing set, repeating this procedure 10 times and computing the average values to generate the confusion matrix. We also use the Recall (Rc), Precision (Pc) and f-score metric.

The TVA dataset contains 48 repetitions of each gesture, the SW dataset contains 88 repetitions and the Cerfav dataset contains 13 samples from G1, 12 from G2, 23 from G3, 9 from G4, 16 from G5 and 5 from G6. Moreover, for the gesture recognition engine, we selected 30 clusters for the K-Means algorithm and 15 states for the HMMs, which follow an ergodic topology. Finally, five different tests have been done in order to compare the gesture recognition accuracy using the following criteria: 2D against 3D in TVA and Cerfav dataset, 2 joints against 7 joints in TVA dataset, different camera positions in SW and Cerfav dataset and mixing gestures from the SW and TVA datasets.

5.2.1 Pose estimation comparison using the TVA, SW and Cerfav datasets

A comparison of the pose estimation using the three datasets has been made. We compared the number of frames per gestures with (1) the percentage of frames without any estimated skeleton, thus without estimation at all; (2) the percentage of frames without any reference point, thus without the neck and, (3) the percentage of frames having the minimum useful estimation, thus at least the neck.

The results are shown on Table 5.1 and we can affirm that the lateral views provided by the SW dataset and all the views provides by Cerfav dataset have a better potential, than the TVA dataset, since a full skeleton has been estimated for all the frames. In the top mounted view of the TVA dataset, the algorithm struggled to estimate any information in 43% of frames for G2 and in 36% of frames for G4, because the user is not captured in many frames. With regard to the TVA dataset, we would expect that these results might have an impact in the gesture recognition accuracy. We would also expect that the small duration of the G1 of SW dataset might also affect its recognition accuracy. Moreover, from the duration of Cerfav we can notice how the duration of the frames varies depending on the gesture, G6 usually lasts 526 frames while G2 last 56 frames, we assume that the shortest gestures (G2 and G4) will tend to fail more than the long ones.

Table 5.1: OpenPose results on the TVA and SW datasets

Dataset	*TV Assembly*			*Silk Weaving*			
Gesture	**G1**	**G2**	**G3**	**G1**	**G2**	**G3**	**G7**
# Frames per sample	80.62	67.89	88.69	164.27	30.85	41.81	66.16
% Frames without any skeleton estimated	0.88	12.27	1.31	14.39	0	0	0
% Frames without reference point (without neck)	2.71	31.05	1.41	22.83	0	0	0
% Frames minimum useful estimation (at least the neck)	**96.41**	**56.67**	**97.28**	**62.78**	**100**	**100**	**100**

Dataset	*Cerfav*					
Gesture	**G1**	**G2**	**G3**	**G4**	**G5**	**G6**
# Frames per sample	448.92	56.00	466.69	92.44	235.25	526.60
% Frames minimum useful estimation (at least the neck)	**100**	**100**	**100**	**100**	**100**	**100**

5.2.2 Gesture recognition comparisons using the TVA dataset: 2D vs 3D and 2 vs 7 joints

The joints selected for training the gesture recognition engine with the TVA dataset are the upper-body joints. Table 5.2 compares the recognition accuracy by using 2 (wrists) or 7 (wrists, elbow, shoulders and head) joints in the 2D or 3D space. The highest results are obtained by using 7 joints in 3D, while the worst with 2 joints in 2D.

Moreover, as we expected, the low number of frames with the minimal useful pose estimation for G2 and G4 impacted the recognition accuracy for these gestures, while G1 and G3, which have data that give good pose estimation, achieved very high accuracy.

Additionally, on one hand, we notice that, while what we gain with the 3D is not so important compared with the 2D, the potential error in the accuracy (standard deviation) decreases for approximately 40% with the 3D. On the other hand, if we use 7 joints instead of 2, we increase the accuracy for more than 10%, meaning that not mostly the hands are involved into the effective gestures. In any case, the way the 3rd dimension is extracted is biased by the fact that OpenPose is already pre-trained using only the RGB, meaning that a complete re-training of the OpenPose with the depth might give better results. Finally, the number of joints that give better accuracy really depends on the nature of gestures.

Table 5.2: Comparison in the gesture recognition using different joints and dimension and TV Assembly dataset

2J-2D	*G1*	*G2*	*G3*	*G4*	**Pr(%)**	***2J-3D***	*G1*	*G2*	*G3*	*G4*	**Pr(%)**
HMM1	81	1	3	6	89.0	*HMM1*	80	1	8	3	87.0
HMM2	0	70	1	29	70.0	*HMM2*	0	73	1	15	82.0
HMM3	17	12	53	5	60.9	*HMM3*	1	28	0	85	66.3
HMM4	2	14	0	96	85.7	*HMM4*	1	28	0	85	74.6
Rc(%)	81.0	72.2	92.3	70.6	76.9 ± 7.9	**Rc(%)**	84.2	65.0	87.5	77.3	77.2 ± 4.7
7J-2D	*G1*	*G2*	*G3*	*G4*	**Pr(%)**	***7J-3D***	*G1*	*G2*	*G3*	*G4*	**Pr(%)**
HMM1	85	1	4	0	94.4	*HMM1*	98	1	1	0	98.0
HMM2	0	84	0	11	88.4	*HMM2*	0	82	1	5	93.2
HMM3	4	16	71	8	71.7	*HMM3*	4	7	68	7	79.1
HMM4	0	6	0	100	94.3	*HMM4*	6	114	0	94	82.5
Rc(%)	95.5	78.5	94.7	84.0	87.2 ± 6.0	**Rc(%)**	90.7	78.8	97.1	88.7	88.1 ± 3.4

5.2.3 Gesture recognition comparison using three different camera positions from the SW dataset

The accuracy in the Table 5.3 is 100%, meaning that the recognizer works perfectly for the gestures of the SW dataset. We think that the difference between the accuracy of the two datasets is mostly due two main reasons: 1. the top mounted view used in the TVA dataset and 2. the fact that in a number of frames the user is not captured in the TVA dataset, thus there is no any pose estimation. In addition, the three gestures made in the SW dataset have a greater variance in space than in the TVA dataset.

Table 5.3: Gesture recognition using all joints and 2 dimensions on Silk Weaving dataset

2J-2D	*G4*	*G5*	*G7*	**Pr(%)**
HMM1	183	0	0	100
HMM2	0	166	0	100
HMM3	0	0	181	100
Rc(%)	100	100	100	100 ± 0

5.2.4 Comparison mixing gestures and data from the TVA and the SW datasets

The last experiment we tried is mixing gestures and data from the two datasets. In Table 5.4, we calculate the precision, recall and f-score for each gesture when we train the gesture recognition engine with samples from both datasets. As a general conclusion, we notice that there is a decrease of accuracy for every gesture. Nevertheless, this decrease is not important given the fact that we have many users, thus a high variance in the way the gestures are executed, and different camera positions.

Table 5.4: Gesture recognition mixing gestures from SW and TVA datasets and using 7 joints and 2D

	G1	*G2*	*G3*	*G4*	*G5*	*G6*	*G7*
Pr(%)	81.9	56.7	80.9	62.8	98.3	98.8	99.5
Rc(%)	84.6	77.5	68.5	69.9	96.6	97.6	95.4
f-score(%)	**83.2**	**65.5**	**74.2**	**66.2**	**97.4**	**98.2**	**97.4**

5.2.5 Gesture recognition comparisons using the Cerfav dataset: 2D vs 3D

The final test is on the Cerfav dataset recorded with the iPhone XS camera. If we look at table 5.5 we can affirm that the results are numerically worse than those obtained with SW and TVA datasets. We attribute these results to:

- The lack of samples in some of the gestures, i.e. the gesture with more samples is the G3 with 23 samples and the gesture with less is G6 with 5 samples, TVA dataset and SW dataset have 48 and 88 samples per gesture respectively. In addition, this results in an unbalanced database.
- The problem of multiperson pose estimation, i.e. unlike the other datasets, in Cerfav dataset there are more than one person in the frame, this causes a multiple pose estimation and therefore, we need to select the person of interested. This selection is based on the person with the highest confidence value, that does not always coincide with the person of interest. A possible improvement would be to add a tracker although it would increase the computational cost.
- Cerfav dataset has been recorded from different points of views ans in movement, factors that may influence the recognition of gestures.
- The number of gestures is greater, six gestures while TVA has four and SW has three, and therefore increases the possibility of failure.

Despite all the challenges that took place in Cerfav dataset, the results obtained are good, especially for 3D case. The final accuracy obtained in the 2D case is 58.7% far surpassing a random recognizer (16.66%), while if we use the 3D data we improve the recognizer up to 62.5% accuracy rate, in addition to reduce the standard deviation again as in the same comparison but with TVA dataset. This last statement supposes that the experiment carried out to add the depth maps is beneficial in the gestures recognition.

Analyzing in more detail the results of the Table 5.5, we can notice that G6 is always the worst-recognized gesture, we assume this is because there are only 5 samples of this gesture, on several occasions G6 is not even selected for testing, in contrast, the gestures with more samples, G1, G3 and G5, are the ones with the highest precision

Table 5.5: Comparison in the gesture recognition using different joints and dimension and TV Assembly dataset

2D	*G1*	*G2*	*G3*	*G4*	*G5*	*G6*	**Pr(%)**
HMM1	14	11	0	0	0	0	56.0
HMM2	0	25	0	0	0	0	100.0
HMM3	0	11	25	4	1	2	58.1
HMM4	0	0	1	17	0	0	94.4
HMM5	2	8	2	9	11	2	32.4
HMM6	2	2	4	3	2	2	13.3
Rc(%)	77.8	43.9	78.1	51.5	78.6	33.3	58.7 ± 8.9

3D	*G1*	*G2*	*G3*	*G4*	*G5*	*G6*	**Pr(%)**
HMM1	9	19	0	0	0	0	32.1
HMM2	1	26	0	0	0	0	96.3
HMM3	0	11	34	0	0	0	75.6
HMM4	0	0	0	22	0	0	100.0
HMM5	2	7	1	4	13	0	48.1
HMM6	0	3	5	3	0	0	0.0
Rc(%)	75.0	39.4	85	75.9	100.0	0.0	65.0 ± 6.3

5.3 Conclusion

In this section we have experimented with 3 different datasets, Silk Weaving, TV Assebly and Cerfav dataset, in 2D or 3D. The best results in gesture recognition have been obtained with the SW dataset, while Cerfav dataset in 2D achieved the lower ones. However, the main hypothesis of these experiments has been confirmed: depth information improves gesture recognition in both accuracy and stability

Chapter 6

Conclusion and future work

6.1 Conclusion

In this work, we developed the first version of a smartphone application that allows users to record human motion using the RGB or the RGB-D sensors, annotate gestures, estimate the pose and recognize the gestures. The use of a smartphone in industrial or professional context is much easier than the use of highly intrusive body tracking systems. The long term goal of this application is to permit to industrial actors to record their own professional gestures, to annotate them and to use a user-friendly system for their recognition. We developed a module that extracts the 3rd dimension from a depth frame. We concluded that the 3rd dimension improves the gestures recognition, around 7% using Cerfav dataset, and the gestures recognition stability, decreasing by 40% the standard deviation in the accuracy of TV Assembly dataset. In addition to this, we observed that while hand are involved in most of cases, using 7 instead of 2 joints can give better recognition results, especially when the camera is top-mounted.

6.2 Future work

Our future work will be focused not only on improving the application, thus improving the interface, but also on the further development of professional gestures dataset. Therefore, the future work of this research project is:

- As the size of the datasets increases we will be able to consider also the use of Deep Learning instead of Hidden Markov Models.
- We plan to extend our pose estimation and gesture recognition system towards the direction of using finger motion as well.

- Retrain the pose estimator to be able to accept RGB-D images, for this we will need to create a well defined dictionary of gestures/activities and a balanced dataset.
- Increase the size and balance of databases, which will lead to an improvement and greater reliability in the precision of the results.
- Optimize client-server communication using cloud tools, such us iCloud.

Bibliography

[1] "Capturing depth in iphone photography, wwdc 2017." https://developer.apple.com/videos/play/wwdc2017/507/.

[2] Intel, "Intel realsense depth camera d400-series," 2017.

[3] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE MultiMedia*, vol. 19, pp. 4–12, April 2012.

[4] H. Fang, S. Xie, and C. Lu, "RMPE: regional multi-person pose estimation," 2016.

[5] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[6] R. A. Güler, N. Neverova, and I. Kokkinos, "Densepose: Dense human pose estimation in the wild," 2018.

[7] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *CoRR*, vol. abs/1612.03144, 2016.

[8] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik, "End-to-end recovery of human shape and pose," *CoRR*, vol. abs/1712.06584, 2017.

[9] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "SMPL: A skinned multi-person linear model," *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, vol. 34, pp. 248:1–248:16, Oct. 2015.

[10] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele, "Deepcut: Joint subset partition and labeling for multi person pose estimation," 2015.

[11] Z. Cao, G. Hidalgo, T. Simon, S. Wei, and Y. Sheikh, “Openpose: Realtime multi-person 2d pose estimation using part affinity fields,” 2018.

[12] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2014.

[13] S. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, “Convolutional pose machines,” *CoRR*, vol. abs/1602.00134, 2016.

[14] “Alphapose.” https://github.com/MVIG-SJTU/AlphaPose.

[15] “Densepose.” https://github.com/facebookresearch/DensePose.

[16] “End-to-end recovery of human shape and pose.” https://github.com/akanazawa/hmr.

[17] “Deep(er)cut: Multi person pose estimation.” https://github.com/eldar/deepcut.

[18] “Openpose.” https://github.com/CMU-Perceptual-Computing-Lab/openpose.

[19] A. Ignatov, R. Timofte, W. Chou, K. Wang, M. Wu, T. Hartley, and L. V. Gool, “AI benchmark: Running deep neural networks on android smartphones,” 2018.

[20] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba, “Eye tracking for everyone,” 2016.

[21] A. Melnikov and I. Fette, “The WebSocket Protocol.” RFC 6455, Dec. 2011.

[22] L. Zhang, G. Zhu, P. Shen, and J. Song, “Learning spatiotemporal features using 3dcnn and convolutional lstm for gesture recognition,” *ICCV workshop*, 2017.

[23] H. Wang, P. Wang, Z. Song, and W. Li, “Large-scale multimodal gesture recognition using heterogeneous networks,” *ICCV 2017 Worhshop*, pp. 3129–3137, 2017.

[24] P. Wang, W. Li, S. Liu, Z. Gao, C. Tang, and P. Ogunbona, “Large-scale isolated gesture recognition using convolutional neural networks,” 2017.

[25] *Dynamic Time Warping*, pp. 69–84. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[26] L. Rabiner and B. Juang, “An introduction to hidden markov models,” *ASSP Magazine, IEEE*, vol. 3, pp. 4 –16, Jan. 1986.

[27] A. Corradini, “Dynamic time warping for off-line recognition of a small gesture vocabulary,” pp. 82–, 2001.

[28] E. Coupeté, F. Moutarde, and S. Manitsaris, "Multi-users online recognition of technical gestures for natural human–robot collaboration in manufacturing," *Autonomous Robots*, Feb 2018.

[29] N. Gillian and J. A. Paradiso, "The gesture recognition toolkit," *Journal of Machine Learning Research 15*, 2014.