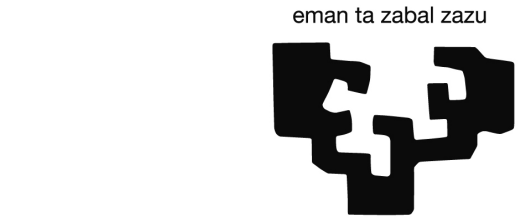


Computationally efficient deformable 3D object tracking with a monocular RGB camera

A dissertation submitted for the degree of Doctor in Computer Science

Jon Goenetxea Imaz

Advisors

Fadi Dornaika

Luis Unzueta

Donostia-San Sebastián, September 2020

(cc)2020 JON GOENETXEA IMAZ (cc by 4.0)

Nire amari.

Abstract

Monocular RGB cameras are present in most scopes and devices, including embedded environments like robots, cars and home automation. Most of these environments have in common a significant presence of human operators with whom the system has to interact. This context provides the motivation to use the captured monocular images to improve the understanding of the operator and the surrounding scene for more accurate results and applications.

However, monocular images do not have depth information, which is a crucial element in understanding the 3D scene correctly. Estimating the three-dimensional information of an object in the scene using a single two-dimensional image is already a challenge. The challenge grows if the object is deformable (e.g., a human body or a human face) and there is a need to track its movements and interactions in the scene.

Several methods attempt to solve this task, including modern regression methods based on Deep Neural Networks. However, despite the great results, most are computationally demanding and therefore unsuitable for several environments. Computational efficiency is a critical feature for computationally constrained setups like embedded or onboard systems present in robotics and automotive applications, among others.

This study proposes computationally efficient methodologies to reconstruct and track three-dimensional deformable objects, such as human faces and human bodies, using a single monocular RGB camera. To model the deformability of faces and bodies, it considers two types of deformations: non-rigid deformations for face tracking, and rigid multi-body deformations for body pose tracking. Furthermore, it studies their performance on computationally restricted devices like smartphones and onboard systems used in the automotive industry. The information extracted from such devices gives valuable insight into human behaviour a crucial element in improving

human-machine interaction.

We tested the proposed approaches in different challenging application fields like onboard driver monitoring systems, human behaviour analysis from monocular videos, and human face tracking on embedded devices.

Acknowledgements

"El guerrero victorioso gana primero y luego va a la guerra, mientras que el guerrero derrotado va a la guerra y trata de ganar.",
Sun Tzu, El arte de la guerra.

Y con este texto termina la aventura del guerrero derrotado que consiguió ganar.

Hace varios años, yo era un soldado inexperto que se aventuró en un viaje incierto con la valentía inconsciente que solo los ignorantes se atreven a blandir. Lleno de confianza, emprendí un camino que pensaba corto y sosegado, pero la realidad no tardaría en sacarme de mi error.

Por fortuna, aquel soldado no estaba solo, y fue gracias a esas personas que estas leyendo estas líneas.

En aquellos comienzos Fadi Dornaika, Luis Unzueta y Blanca Cases se atrevieron a guiarme, y he de decir que lo han hecho de forma magnífica hasta este día. No sabíamos donde acabaría el viaje, pero definimos un rumbo y echamos a andar.

Durante todo el camino, Vicomtech me dio las herramientas que necesitaba para sobrevivir, incluyendo el inestimable consejo de Jorge Posada como director científico. A todo esto se le suma la protección y el amparo de unos compañeros inmejorables, capitaneados por Oihana Otaegui, a quien agradezco especialmente su constante apoyo durante todo el proceso. Gracias también a Maria Teresa Linaza, por su ayuda en nuestras aventuras por Europa, a Luis Unzueta y Unai Elordi, por reconquistar las Américas y a Nerea Aranjuelo y Juan Diego Ortega por todo el trabajo de campo y simuladas carreteras. Ellos, y el resto de mis compañeros, han sido la brújula y el mapa que han hecho que en cada montaña a atravesar hubiera un sendero para caminar.

Y entre todos luchamos contra el peor de los enemigos: lo desconocido. Para el que no lo sepa todavía, lo que desconoces siempre tiene un plan para sorprenderte.

Y luchamos. Luchamos mucho. Ganamos algunas veces y perdimos muchas más. Y cada derrota daba paso a una nueva pelea.

Pero cada vez que mordía el polvo, mi familia venia en mi ayuda. Sin el ánimo de Laura y la sonrisa de mis hijos Luka y Kima, nunca podría haber llegado tan lejos.

Gracias a todos por vuestro apoyo y dedicación.

Y después de todo lo vivido en estos años, sigo siendo el mismo soldado; inexperto, aunque algo menos ignorante. Pero sé algo que muchos todavía ignoran: *Ante un viaje peligroso, lo importante no es el destino ni la calidad de tus botas, sino la gente con la que te rodeas en el camino.*

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
  - 1.1.1 Capture devices . . . 2
  - 1.1.2 Gesture and pose interpretation . . . 4
  - 1.1.3 3D Tracking . . . 4
  - 1.1.4 Computational limitations . . . 7
- 1.2 Contributions . . . 8
- 1.3 Thesis organization . . . 10

2 Related work **13**
- 2.1 Object modeling . . . 13
  - 2.1.1 Human body models . . . 16
  - 2.1.2 Human face models . . . 18
- 2.2 3D object tracking . . . 20
  - 2.2.1 Human body as a rigid multibody deformable object . . . 21
  - 2.2.2 Human face as a non-rigid deformable object . . . 23
  - 2.2.3 Methods based on Deep Neural Networks . . . 27
- 2.3 Facial gesture recognition . . . 28
- 2.4 Computationally limited systems . . . 32
- 2.5 Image capture and 3D reconstruction . . . 36
- 2.6 Discussion . . . 39

3 Feature point detection **41**
- 3.1 Related work . . . 42
- 3.2 Face landmark detection by image gradient analysis . . . 43
  - 3.2.1 Lightweight facial feature detection . . . 43
  - 3.2.2 Experimental results . . . 51
  - 3.2.3 Conclusions . . . 51

3.3 Face landmark and gesture detection by DNN regression . . . 51
3.3.1 Multi-level approach . . . 52
3.3.2 Facial attribute definition . . . 54
3.3.3 Network structure . . . 55
3.3.4 Learning process . . . 56
3.3.5 Experimental results . . . 60
3.3.6 Conclusions . . . 65

4 Face fitting with non-rigid deformations 67
4.1 Facial Feature Back-Projection (FFBP) . . . 68
4.1.1 3D deformable model definition . . . 68
4.1.2 Baseline fitting method . . . 70
4.2 Multi-Stage Back-Projection (MSBP) . . . 73
4.2.1 Deformable 3D face model . . . 74
4.2.2 3D face model alignment . . . 75
4.3 Face tracking in a Video Sequence . . . 81
4.3.1 Baseline tracker . . . 82
4.3.2 Fast tracker . . . 82
4.3.3 Fast tracker with points filtering . . . 84
4.4 Experimental results . . . 85
4.4.1 Experimental setup . . . 86
4.4.2 3D orientation comparison . . . 88
4.4.3 3D Shape Estimation Comparison . . . 91
4.4.4 Computation Time Comparison . . . 92
4.4.5 Performance Analysis on ARM Architectures . . . 93
4.4.6 Qualitative results . . . 97
4.5 Conclusions . . . 98

5 Body pose estimation with multi-body deformations 101
5.1 Method . . . 102
5.1.1 Camera calibration . . . 102
5.1.2 Body pose estimation . . . 103
5.2 Automatic multi-body projection adjustment process . . . 107
5.3 Experimental results . . . 110
5.4 Conclusions . . . 118

6 Application fields **119**
6.1 Driver inattention monitoring 120
6.1.1 Methodology 122
6.1.2 Results 134
6.1.3 Conclusions 137
6.2 Sport skill analysis by 3D body pose extraction 138
6.2.1 Data capture and analysis 139
6.2.2 Extraction of the athlete's 3D motion 141
6.2.3 3D motion analysis 145
6.2.4 Experimental results with Pallapugno videos 148
6.2.5 Conclusions 150

7 Conclusions and future work **153**
7.1 Conclusions 153
7.2 Future work 155
7.3 Relevant publications 157
7.3.1 Journals 157
7.3.2 Books and book chapters 157
7.3.3 Conferences, congresses and workshops 158
7.3.4 Patent applications 159

A Appendix Title **181**
A.1 Candide model modified deformations 181

List of Figures

1.1 From left to right, an RGB image, a IR image and the depth map from a RGBD sensor. Each color of the depth map represents a different distance value. 3
1.2 A visual representation of discriminative and generative approaches. 5
1.3 A tracked sequence (from left to right) of a face using an automatic 3D face tracker. The upper row shows the original image captured by a colour camera and the lower row shows the same images with a 3D face model overlapping the face representing the tracking output. 7
1.4 A graphical representation of the process to fit a 3D model to an RGB image. The figure includes the case of fitting non-rigid deformable models for face gesture reconstruction and multi-body deformable objects for human pose estimation. . . 8
1.5 The pipeline shown in Figure 1.4 with a more detailed object region definition step, and including frame-to-frame update elements. 10

2.1 Examples of each model representation class: a) structural face representation, b) 2D area based face representation, c) volumetric face representation, d) structural or kinematic body representation, e) contour-based body representation and f) volumetric body representation. Face image *b* extracted from [34], face image *c* extracted from [35] and images *d*, *e* and *f* extracted from [17] 14
2.2 Some examples extracted from annotated 2D body pose datasets. Original image extracted from [17]. 15
2.3 Some examples extracted from annotated 3D body pose datasets. Original image extracted from [17]. 17

2.4 On the left, some examples of the body poses and shapes using SMPL body model. The image shows the body pose estimated using SMPL model configuration (brown) vs the ground truth 3D scan model (gray). On the right, some examples of the poses and shapes using Stitch body mode. Each colour represents a different deformable body part. 18
2.5 Face feature definition examples from different annotated facial image datasets: a) [46, 47] b) [48] c) [49] d) [50] e) [51] f) [52] g) [53] h) [54]. Image extracted from [47] 19
2.6 Different 3D vertex densities of the deformable face model proposed by [63]. 21
2.7 On the left, 3D body pose estimations using [36]. On the right, 3D body pose estimations using [10]. 22
2.8 Examples of face deformations caused by facial cardinal expressions (top row) and examples of different human faces (bottom row). The facial expressions in the top row are (from left to right): neutral, happy, surprised, sad, afraid, disgusted and angry. Images extracted from KDEF dataset [76]. 24
2.9 Left image shows some examples of the fitting results using [19]. Right image shows the reconstruction of the 3D shape of a face using [79]. 25
2.10 On the left, some examples of the 3D model generated by [62], and on the right, the results of using this model with a different actor. 26
2.11 The left image shows the region-based deformable 3D model proposed in [81]. Right images show two examples of 3D model fitting on a face image. 28
2.12 Action Units #2 (left) and #10 (right). Top row shows the rest state of the action, while the bottom row shows the active state. 31
2.13 On the left, a Raspberry Pi 4 Model B with a Broadcom BCM2711 SoC. On the right, a carrier board designed by "Connect Tech" for the Jetson Xavier NX from Nvidia. 34
2.14 On the left the Jetson Xavier NX SoM from Nvidia. On the right the Coral SoM from Google. 35
2.15 On the left, Movidius Myriad-2 USB AI-accelerator (VPU) from Intel. On the right, Coral USB AI-accelerator (TPU) from Google. 35

2.16 Simplified representation of the pinhole camera model. 37
2.17 Different images of the same face taken with a full-frame camera and using objectives with different focal lengths. Original image by Stephen Eastwood[1] 38

3.1 Proposed fitting approach. From left to right and top to bottom: (1) The detected face region and the *faceROI* derived from it (thicker line), (2) *faceROI* and the *eyeSROIs* derived from it (thicker line), (3) *faceROI*, the estimated *eyeROIs* and the *eyebrowSROIs* and *mouthSROI* derived from them (thicker lines), (4) *faceROI*, the estimated *eyeROIs* and the *noseSROI* derived from them (thicker line) and (5) the detected facial features. 46
3.2 Facial feature detection procedure steps. From left to right and top down: (1) eye point detection, (2) eyebrow point detection, (3) mouth point detection, (4) nose point detection, (5) contour point detection and (6) OAM tracker model initialization example. 47
3.3 The detected 32 facial points. Note that the words *left* and *right* are relative to the observer rather than the subject. . . . 48
3.4 Eye points geometry derived in a fixed way from the estimated *eyeROIs*. 48
3.5 Flowchart of the proposed multi-level analysis pipeline. 52
3.6 From left to right, a representation of the 32 landmarks detected in the first level, a representation of the landmarks detected in the second level and a combined representation of all the detected features. Each colour represents the landmarks for each face zone. 53
3.7 Examples of (from left to right) an eyebrow patch, a right eye patch and a mouth patch using the image shown in Figure 3.6. The red cross represents the *target location* of the *reference landmarks* for each patch. Note that the reference point of the mouth patch is out of the patch region. 54
3.8 The general structure of the multi-task network designed for level 1. The layers in the trunk generate all the features for the leave layers, sharing the output values with both leaves. The leave layers are the output of the network. 56

3.9 The neural network configuration for the attribute estimation process. The first row shows the network configuration for the first estimation level, including the input and the output elements. The rows below show the network configuration for each estimation model in the second level. A single eye model estimates the eye attributes for both eyes, using symmetry to estimate the attributes of the second eye. For each model, 'C' represents the kernel size of the convolutional layer and the 'P' represents the configuration of the pooling layer. 57
3.10 Some examples of training patches generated with the described perturbations. From top to bottom, the original face images, the patches for model in level 1, patches for eyebrow model and patches for mouth model. 61
3.11 Two examples of the synthetic eye render (left column) and some of the eye region training patches generated with those images. The eye patches include the augmentation perturbations. 62
3.12 Each image shows the estimated gesture probabilities as bars (left), the tracked facial feature points (dots) and the estimated eye gaze vector (purple arrows). The gesture definitions are, from top to bottom, head pose left profile (HPLP), head pose left-front (HPLF), head pose front (HPF), head pose right-front (HPRF), head pose right profile (HPRP), mouth gesture smile (MGS), mouth gesture frown (MGF), mouth gesture kiss (MGK), mouth gesture neutral (MGN), eyebrow gesture frown (EGF), eyebrow gesture raise (EGR) and eyebrow gesture neutral (EGN). 63
3.13 Visual representation of the MSE and time values for Kazemi and Sullivan [18], Baltrusaitis et al. [111] and the proposed method (M3). 64

4.1 An adaptation of the graphical representation of the process to fit the 3D model presented in section 1.2. 67
4.2 Face model fitting steps for FFBP (top row) and MSBP methods (bottom row). The left image shows the automatically detected face region. The central image shows the same face with the 2D landmarks overlapped. The right image shows the result of the 3D model deformation and alignment. 69

4.3 The geometries of the Candide-3 and the Candide-3m face models. 70
4.4 The upper image in the figure shows the shape of the deformable 3D model used in this study. The lower row shows the frontal view of the same deformable 3D model (left) and the mean landmark shape of the used 2D landmark detector model (right). In the current representation, all the 3D deformation parameters (SUs and AUs) are set to zero value. . . . 76
4.5 Example of different deformations of the model. Each column shows the same deformation with positive deformation values (top row) and negative deformation values (bottom row). The shown deformations correspond to (from left to right) lip corner depressor (AU), left eyebrow raiser (AU), mouth stretcher (AU), eye vertical position (SU), mouth vertical position (SU) and face width (SU). 77
4.6 Example of the optimisation steps for the 3D model alignment: a) the initialisation of the fitting in the centre of the space, b) the raw estimation of the position and rotation of the 3D object, c) the shape parameter optimisation, and d) the animation parameter optimisation. 78
4.7 Representation of the face location update between the frames #24 and #25 of the sequence *410* of the '300VW challenge' dataset [127]. The tracker extracts the face template (middle) from the frame #24 using the projection of the 3D vertices. Then, it finds the template in the red area in the frame #25. . 84
4.8 Fitting example for each fitting method using the same image. The top row shows the projection of each model on the image. The bottom row shows the side view of the same fitting. Each column represents a method. From left to right FFBP, Huber et al. [7], Baltrusaitis et al. [6], Bulat et al. [8], Feng et al. [85], Kazemi and Sullivan [18] + MSBP and Baltrusaitis et al. [6] + MSBP. 86
4.9 Example of an annotation of the AFLW2000-3D dataset with the ground truth vertices in blue. Each image shows in red the FAP locations for: a) the orientation estimation and b) the shape estimation comparison. 88
4.10 Graphical representation of the face orientation, including the reference plane. 89

4.11 Mean shape error, mean rotation error and mean elapsed time measurements for each of the tested methods, normalised in a range from zero to one. The highest value for each measurement type is set to one while the rest of the values are scaled accordingly. The two horizontal lines show the accuracy values of our fastest method against the rest. In all cases, smaller is better. 94
4.12 Times needed for each device to detect the face in the image(left), to detect the facial landmarks in the detected region (center) and to fit the 3D model using the detected landmarks (right). The vertical axis shows the time (in milliseconds) and the horizontal axis the sample index. 95
4.13 Some results of the fitting process using the AFLW2000-3D dataset. 97
4.14 Proposed method running on an iPhoneSE (left) and a PixelC(right) using images captured with the integrated front camera in real-time. 98

5.1 An adaptation of the graphical representation of the process to fit the 3D model presented in section 1.2. 101
5.2 The CLFBA method components with their corresponding inputs and outputs. 103
5.3 (a) The 3D kinematic model's hierarchical structure (based on H-Anim [136]), (b) its body dimension parameters and (c) its posing features for IK control (located at end-effectors and intermediate upper and lower limb joints). 104
5.4 Examples of biomechanical rotation limits of body joints: (a) shoulder swing limits, (b) knee flexion limits and (c) a cervical vertebra twist limit. 106
5.5 Flowchart of the automatic multi-body projection adjustment process. 107
5.6 Samples from the HumanEva-I dataset sequences used in the experiments, with the pose estimations from Vicon overlapped. From left to right, S1-Box1-C3, S1-Walk1-C2, S3-Box1-C2 and S3-Jog1-C1 111
5.7 An example of the obtained pose reconstruction results (S1-Walk1-C2): a) CLFBA, b) [27], c) [140] and d) [141] 116

6.1 Multi-screen simulator setup for driver behaviour analysis, based on human-machine interaction, including PoG and 3D face tracking 121
6.2 Workflow of the multi-planar PoG estimation and 3D face tracking approach. 125
6.3 A generic deformable 3D face model and some of its deformation parameters compatible with our method. 126
6.4 Examples of the distortion that happens in the normalised appearance of the most distant eyes in non-frontal faces, when the head's yaw angle is changed. 130
6.5 The considered zones of interest in the simulator to analyse the driver's PoG. 131
6.6 Confusion matrix of the predictions obtained by our approach for the considered gaze zones. 136
6.7 An example of PoGx signal where saccades and fixations can be appreciated, along with the level of noise. 136
6.8 Examples of the approach running in an iPhone SE, while the user puts thick glasses on and the system keeps working. . . . 137
6.9 Three examples of an old broadcast video captured for TV. . . 140
6.10 The figure shows: a) floor plane tool alignment with the reference element, b) 14 2D body joint landmark location, and c) the final pose reconstruction. 142
6.11 Image (a) shows the estimation of the automatic joint detector for a given frame [25]. Image (c) shows the same frame with the needed 2D landmark correctly positioned for the 3D reconstruction. The image (b) shows the relation between both representations, and how the missed landmarks (in red) are computed. 144
6.12 Five frames processed with the automatic pose detector. The pose in some of the frames is not correctly estimated. 145
6.13 The complete end-to-end motion comparison pipeline 146
6.14 The image shows the key-frames of the 3D reconstruction of Pallapugno serve skill. The coloured lines represent the key-frame of: the start of back-swing (red), the back-swing to front-swing transition (green), the player-ball impact (blue), and the front-swing to follow-through transition (grey). 149

6.15 Comparison of the right shoulder (up) and right elbow (down) extension features for two Pallapugno serve actions. The feature's measurements over time as well as their similarity are presented. The vertical lines denote specific key-frames that mark the phase transitions as in Figure 6.14 151

A.1 The added and modified SUs and AUs in Candide-3m with respect to Candide-3, showing their variation from -1 to 1 values, where 0 corresponds to the neutral configuration. . . . 182

List of Tables

2.1 The main features of the presented face tracking methods. . . 29
2.2 Main studies based on Deep Learning approach for human 3D body pose reconstruction and tracking. 30
2.3 Main studies based on DNN methods for face tracking. 30

3.1 Results of the landmark estimation time measurement. 64

4.1 List of shape and action units included in the deformable 3D model. 74
4.2 Results of the Orientation Estimation Comparison. 91
4.3 Results of the Shape Reconstruction Comparison. 92
4.4 Mean Computation Times for each Device and Task. 95
4.5 Processing Time Comparison Between Devices. 96

5.1 Camera parameter estimation results of [133] (used for the camera configuration in CLFBA and [27]) and Vicon in the three camera viewpoints (C1-C3) from HumanEva-I dataset. . 112
5.2 2D pose estimation errors in pixels per joint, for each body limb and for the full body, of CLFBA, [27], [140] and [141] with respect to Vicon in HumanEva-I dataset. 114
5.3 3D pose estimation errors in cm per joint, for each body limb and for the full body, of CLFBA, [27], [140] and [141] with respect to Vicon in HumanEva-I dataset. 115
5.4 Body part size estimations in cm of CLFBA and Vicon in HumanEva-I dataset 117
5.5 Average time measurements for the optimization process, depending on the parameters to be obtained 117

6.1 Comparison among different state-of-the-art eye gaze estimation systems and ours. 133
6.2 Weights allocated to the motion features of each joint in the three different phases of the Pallapugno serve 149

Chapter 1

Introduction

1.1 Motivation

Advances in fields such as robotics and artificial intelligence are progressing by leaps and bounds. Today we can create autonomous robots capable of walking like humans, talking like humans, and listening like humans allowing a more natural interaction between the machine and the user.

Natural speech interfaces are tools already implemented in commercial products and are being used in many homes to improve user experience with multimedia applications and with home automation. Such interfaces improve communication with specific machines, but their understanding of the user's situation is limited to the information provided through the voice.

The average human can see and interpret non-verbal communication coming from their interlocutor. Above all, they can evaluate a large amount of information by visually interpreting face gestures and/or body pose. This kind of interpretation is something that machines have not yet managed to do.

Improvements in this interpretation capacity need new ways to extract the information from the primary sources of non-verbal communication: the facial gesture and the pose of the body. To this end, the system needs to track the face or body and interpret the poses and movements, both being challenging elements to track, especially in a three-dimensional environment.

The main task of a tracking system is to estimate the position and orientation of an object in the scene using a captured data sequence. This information is sufficient to monitor possible interactions between different

scene elements (e.g., distances between objects or velocities), but ignores the internal information of the tracked object. In the case of a rigid object (e.g., a mug or a box), its intrinsic information remains constant. However, for deformable objects (e.g., a cat or a person) the intrinsic shape can change, offering valuable information for the interpretation of the scene as a whole. Thus, the next important task is to estimate the internal pose or configuration of the object itself.

Presumably, a three-dimensional object also has a three-dimensional deformation. This assumption is especially crucial in a context where the interpretation of the scene depends on the interaction of the different elements within it. For example, we can understand that a person is greeting or pointing, but the orientation of the gesture is an essential part of this communication. The same goes for facial gestures, which can be interpreted differently according to the orientation of the head, for example.

1.1.1 Capture devices

The first barrier between the tracking system and the real 3D scene is the capture system. These hardware elements provide the data needed to analyse the scene, so we must make sure that it provides the essential information.

There are several types of capture setups. Some setups are as simple as a colour camera, while others combine several infrared cameras, spotlights and special suits full of reflective markers. The capture system will largely determine the accuracy of the final result, but the context of the final application usually limits the use of certain setups.

Since the use of specific suits or wearables is too intrusive in natural environments (e.g., driver behaviour analysis), this section will focus on capture systems that do not use them. Besides, the combination of multiple sensors is unfeasible in several scenarios, for instance, in the case of smartphones. Thus, we limit the possible setups to those that use a single capture sensor.

The most commonly used devices in these types of environments are: infrared (IR) cameras, visible light cameras (red-green-blue or RGB), and 3D capture sensors combining RGB data with a depth map (RGBD). Figure 1.1 shows an example of each image type.

IR cameras capture light in a spectrum close to the colour red, which is not visible to the human eye, and project light intensities in a plane. While sunlight has an IR component, Infrared cameras still need a specific infrared light source, consisting of an infrared spotlight. This extra illumination in-

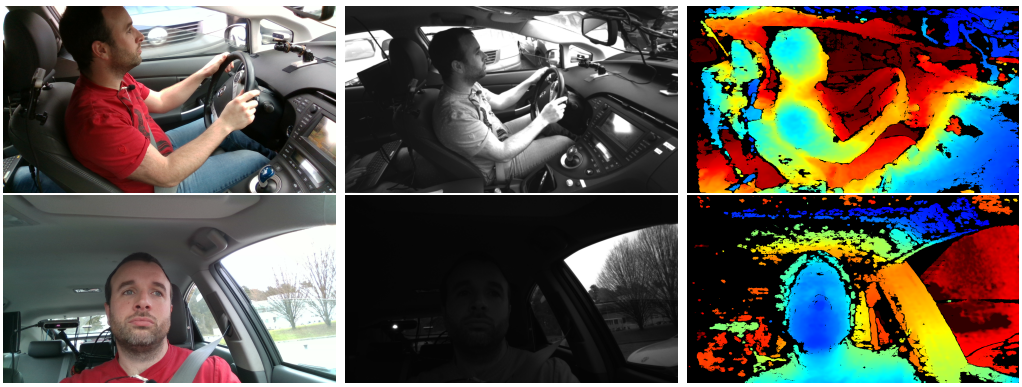


Figure 1.1: From left to right, an RGB image, a IR image and the depth map from a RGBD sensor. Each color of the depth map represents a different distance value.

creases power consumption, which could be high depending on the area to be monitored by the system. Although infrared light is not visible, it affects the tissues of the eye [1] and can be dangerous in some instances, so it is necessary to limit its use if it is necessary to illuminate a face directly.

RGB cameras capture the visible light of a scene and project the colour data into a plane. They are widely used tools and are integrated into several types of devices like smartphones, laptops, commercial intercoms and surveillance platforms. However, similar to IR cameras, they need a light source, and this makes them problematic in dark environments like under illuminated rooms or night scenes.

Both device types (IR and RGB cameras) project the light information in a single plane, losing the depth information in the process. Some studies [2–4] propose using a multi-camera system to reconstruct the 3D information via triangulation, but those setups include additional constraints and considerations [5].

Other studies [6–11] use a predefined deformable 3D model to estimate the depth values of the face or body by re-projection. The tracking system adapts the appearance of the 3D model modifying its deformation parameters while estimating its position and orientation to fit its projection in the image, taking into account the camera parameters.

As an alternative, RGBD cameras capture RGB information plus the depth value estimation of each pixel. The main capture techniques for RGBD

cameras are structured light projection (e.g., Microsoft Kinect) and time-of-flight sensors (e.g., Microsoft Kinect 2). In both cases, the estimation tends to be noisy and with low resolution, in addition to being sensitive to external factors such as IR light sources (i.e., sunlight). To correctly interpret the captured 3D information, methods like [12] use a deformable 3D model similar to those used with RGB and IR cameras. This model helps to reduce the noise in the captured data and prevents errors caused by occlusions and IR light reflections.

1.1.2 Gesture and pose interpretation

The automatic interpretation of the facial gesture and the human body pose allows for a more in-depth interpretation of the scene.

This information reinforces the analysis of other elements in the scene, providing general context and additional information [13]. For example, facial gesture estimation can contextualise a parallel process of speech recognition, or it could determine the inattention of the user during specific actions. Similarly, 3D body pose information can reinforce a human behaviour analysis process. Moreover, both sources of information (i.e. facial gesture and body pose) complements each other, allowing a deeper understanding of human actions.

In motion capture and performance transfer applications, facial gesture information improves the facial gesture transfer from a real actor to a virtual avatar [14]. In the animation process, the expressiveness of the virtual avatar can differ significantly from that of the actor, due to possible differences in face shape and expressiveness between them (e.g., cartoon characters). Systems that only rely on the relative movement of facial elements with respect to the neutral facial gesture may lose expressiveness in the transfer process, requiring a post-production phase to correct these losses. Instead, a subjective analysis of the gesture can be applied directly to its representation in the avatar, making the avatar smile while maintaining its original expressiveness, for example.

1.1.3 3D Tracking

3D tracking consists of the interpretation of the 3D movements of a 3D element in a 3D scene over a time frame. The type and nature of the interpreted

data depend on the adopted tracking method and the needs of the final application.

There are different categorisations for tracking methods, depending on the adopted approach [15–17]. However, the most meaningful in this context could be *generative vs discriminative* categorisation.

At a technical level, discriminative methods segment the model deformation space during training and then discriminate between different kinds of data instances on inference (see left image in Figure 1.2). They rely on the conditional probability $p(Y|X)$ to model the possible deformations.

In contrast, generative methods use a previously learned deformation model to encapsulate the underlying join probability distribution $p(X, Y)$ of the deformations (see right image in Figure 1.2). To generate the deformation model, we can use both automatic learning processes and structured deformable models based on deformation constraints; such as kinematic models in the case of the human body.

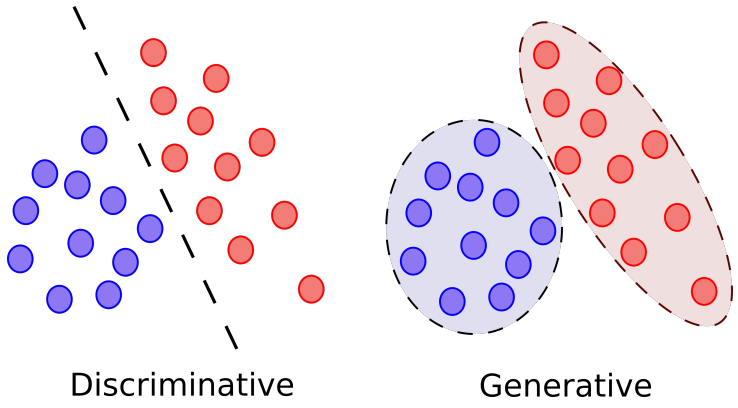


Figure 1.2: A visual representation of discriminative and generative approaches.

In the field of two-dimensional facial shape reconstruction, two representative examples of those categories are the methods proposed by Kazemi and Sullivan [18] for discriminative methods and Baltrusaitis et al. [19] for generative methods.

In [18], the authors propose a method based on ensembles of regression trees, which iteratively improves an initial face shape estimation using previously trained binary trees. A binary tree is a flowchart-like structure, where

each node discriminates between two possible outputs, deciding which is the next node to evaluate. At the end of the tree, the leave node stores the regression values learned during the training phase.

The method proposed in [19] relies on a PCA (principal component analysis) deformation space, which models the possible facial deformations based on the samples of the training dataset. During inference, an iterative process estimates the configuration to fit the PCA model with the evaluated face image.

In summary, generative methods [10, 19, 20] rely on a previously defined model of the tracked object, and they adapt the model configuration to estimate the status of the object in the scene. In contrast, discriminative methods [18, 21–23] rely on a direct mapping between the tracked object and a previously defined deformation space. Discriminative methods are usually computationally faster on inference compared to generative methods. They use fast regression or boosting approaches to estimate the features learned during the mapping learning phase, reducing the search space and computational complexity. However, generative methods require less data for training and are more robust for object configurations not present in the data used to define the deformation space.

Apart from this categorisation, the tracking pipeline has different steps [15]. Based on this pipeline, we can highlight two approaches: a) tracking by continuous re-detection and b) tracking by a frame-to-frame update.

Tracking by re-detection methods does the object detection and pose estimation in each of the frames. Then they relate the detections in consecutive captures to generate the full tracking sequence. Conversely, methods based on the frame-to-frame update perform the detection of the object in the first frame then, for consecutive frame sequences, use the information of the previous frame to update the position, orientation and pose in the current frame.

Frame-to-frame update-based methods avoid the object detection step, which is a computationally expensive process, reducing the required computation resources. The object information from previous frames makes the tracking more robust to partial occlusions. However, this information can degrade due to cumulative estimation errors or heavy occlusions, reducing tracking accuracy. To fix the degradation, they need an extra monitoring task when the object tracking is lost (i.e. the tracking accuracy is not acceptable) and needs to re-detect the object restarting the tracking.

Figure 1.3 shows some example frames of a tracking sequence using these

Figure 1.3: A tracked sequence (from left to right) of a face using an automatic 3D face tracker. The upper row shows the original image captured by a colour camera and the lower row shows the same images with a 3D face model overlapping the face representing the tracking output.

types of techniques.

The first column of Figure 1.3 shows the detection frame. In this instance, the tracker identifies the face or the body and estimates the initial configuration of the 3D model. The following frames (from left to right) show how the tracker updates the deformation of the 3D model during the video sequence.

There are different techniques to update the model configuration between frames [15]. Each tracking system has distinctive characteristics that must be evaluated depending on the context of the final application. A system that needs a very detailed reconstruction of the face has some features and limitations that, for example, a system that only needs the position and orientation of the head does not have.

1.1.4 Computational limitations

There are application environments that can significantly benefit from the possibilities offered by this kind of tracking system. However, at the time of defining a tracking system, it is essential to consider the singularities of the application context to evaluate its needs and limitations, and it is necessary to analyse what the constraints are in each case. For example, an autonomous robot depends on its battery to operate, so the computing power for the interaction and interpretation system is limited. In driver monitoring systems

for vehicles, the onboard hardware analyses the data locally to avoid latencies, so the energy and computing power is also limited. Furthermore, a PC without these external limitations could have problems with computing processes that exceed its processing capacity.

These are some examples of computationally limited environments. Following sections analyse the most widely adopted hardware architectures in these kinds of environments, and they show some solutions depending on computing needs.

1.2 Contributions

This study focuses on the estimation of the 3D pose inherent in the human body using systems with computational limitations as the final deployment target. More precisely, it addresses techniques for estimating the 3D human body pose and 3D gesture of a human face (including head position and orientation). To capture the scene data, it uses a single RGB camera due to its simplicity, efficiency and the high amount of data captured in each frame (e.g., high-resolution frames, texture and colour). The presented techniques can also be applied to track other 3D elements such as hands and fingers, or lung and heart movements for biomedical applications.

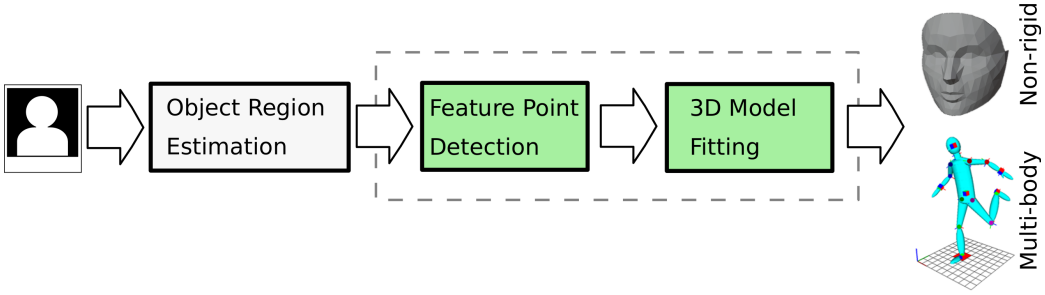


Figure 1.4: A graphical representation of the process to fit a 3D model to an RGB image. The figure includes the case of fitting non-rigid deformable models for face gesture reconstruction and multi-body deformable objects for human pose estimation.

The high variability of the body and the face makes it difficult to consider all the possibilities in advance. Hence, it makes more sense to do the

tracking using a method based on a generative approach. In this context, and according to the scheme proposed in [15], a general tracking pipeline will include three consecutive steps:

1. Object region estimation.

2. Feature point detection.

3. 3D model fitting.

Figure 1.4 shows a visual representation of the enumerated steps. Following the proposed scheme, the first step locates the tracked object and defines its region in the image. The second step detects a series of feature points of the object in the defined image region (in 2D), and the third step uses the detected feature points and a generic deformable 3D model to estimate the 3D configuration of the object. Some methods [24, 25] combine the first and second step in a single computation, but we represent them separately for a better generalisation.

Based on this pipeline, the PhD study outlines the following contributions:

- An efficient facial feature point detection system based on a learning-free feature detection method [26] .

- An efficient facial feature point detection system based on a learning-based feature detection method, which also extracts other facial attributes like facial gesture estimation, head pose and eye-gaze.

- The definition of a 3D deformable model based on rigid multi-body deformations for human 3D body pose estimation [27].

- The definition of a 3D deformable model based on non-rigid shape deformations for human 3D face gesture estimation [28, 29].

- An efficient 3D fitting method for human body pose estimation [30–33].

- An efficient 3D fitting method for human face gesture estimation [26, 29].

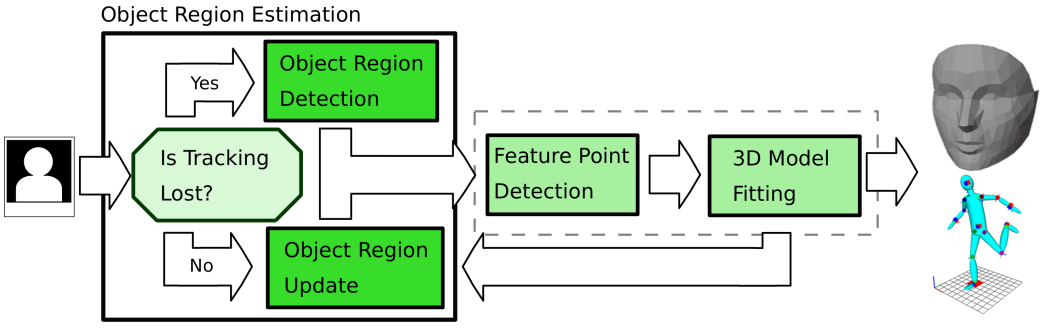


Figure 1.5: The pipeline shown in Figure 1.4 with a more detailed object region definition step, and including frame-to-frame update elements.

Moreover, it defines a strategy to efficiently track 3D elements in RGB video sequences [29], using the information of the previous frame to estimate the location of the object, and avoiding the costly computation of the object region detection step.

Figure 1.5 shows a graphic representation of this strategy. It includes a measurement step to decide if the object is being tracked correctly. If this step concludes that it is not, an automatic object region detection method locates the object in the entire image, followed by the tracking if it finds the object. Otherwise, if the measurement step determines that the object is correctly tracked, a 2D region update method estimates the new object 2D region using the information from the previous frame. Note that for the first frame of the capture sequence, there is no active tracking, so the system uses the automatic region detector to initialise it. This introduced scheme reduces computational complexity since the frame update process is more efficient than object detection.

1.3 Thesis organization

The thesis content is structured in 7 chapters.

Chapter 1 is the current chapter and introduces general concepts, motivations and contributions achieved during the research period.

Chapter 2 exposes the relevant approaches and related work presented in the state-of-the-art literature.

Chapters 3 to 5 expose outline the methods proposed to efficiently track

deformable objects in monocular video sequences following the pipeline presented in the previous section. More specifically, chapter 3 focuses on the feature point detection phase and proposes several methods to efficiently detect the two-dimensional facial features from monocular images, including experimental results with in-the-wild video captures and real-time performance. Chapter 4 proposes an efficient method to fit non-rigid 3D deformable models to faces on monocular images and also introduces the mentioned tracking approach. Chapter 5 describes an efficient method to fit rigid multi-body 3D deformable human body models to human poses in monocular images.

Chapter 6 shows several application examples based on the methods proposed in previous chapters.

Chapter 7 outlines the conclusions and perspectives.

Chapter 2

Related work

This chapter analyses some of the elements that make up a 3D tracker based on deformable 3D objects. According to the proposed scope, it presents the most relevant works in the field, focusing on the human body pose and human face gesture recognition, reconstruction and tracking. The tracking involves recovering a set of variable configuration values from a captured image sequence. The configuration parameters include the three-dimensional pose and some specific settings related to the deformable model, among others.

It first reviews the representation types and models used for human body and face representations. Next, it examines the most relevant methods for their tracking and reconstruction for 2D and 3D solutions followed by a summary of some topics on facial gesture recognition, including different gesture notations and the most relevant gesture recognition methods. Then, it exposes different implementations of computationally limited systems and some of the causes of those limitations. Finally, it includes some notes about the different 3D-2D projection methods, which is a crucial element for the correct estimation of objects in a three-dimensional scene.

2.1 Object modeling

An object tracking system has different constraints and needs depending on the scope of the final application. Some applications may need to track an object within the image plane (i.e., 2D tracking), or they may need more detailed information such as the position of the object in the scene (i.e., 3D tracking), including interactions with other objects. Due to the diversity of

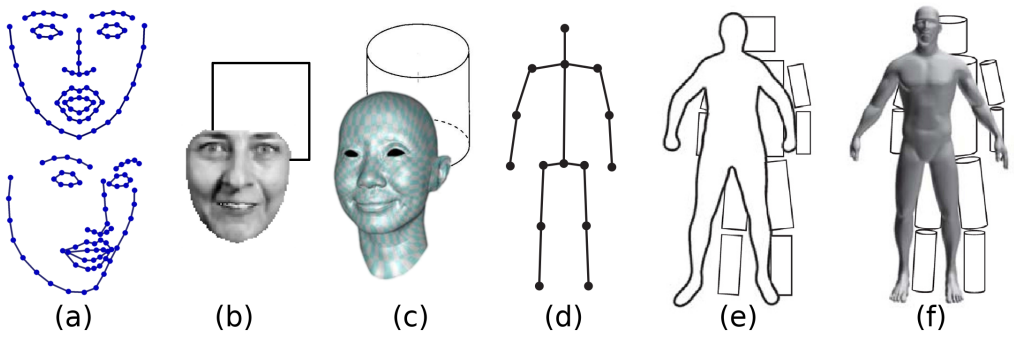


Figure 2.1: Examples of each model representation class: a) structural face representation, b) 2D area based face representation, c) volumetric face representation, d) structural or kinematic body representation, e) contour-based body representation and f) volumetric body representation. Face image *b* extracted from [34], face image *c* extracted from [35] and images *d*, *e* and *f* extracted from [17]

scenarios, the information required of the object and its representation may vary.

The system needs to represent the appearance of the object so that it contains the information necessary for the tracking process. Depending on these needs, the representations can be assigned into three categories:

- Structural or skeleton-based (i.e., kinematic or structural definition)
- Contour-based (i.e., 2D patch and 2D contour definition)
- Volume-based (i.e., 3D surface definition)

In all three cases, the information density can vary depending on the requirements. Figure 2.1 shows some examples of each class for face and body representations.

A structural model consists of a small set of landmarks (2D or 3D depending on the case) which represent the feature points for the body or the face. They also include a set of edges defining the relationship between the landmarks for a better visual interpretation (see images *a* and *d* on Figure 2.1).

Contour-based or silhouette-based models use the object's outline or 2D image information to represent the object (images *b* and *e* in Figure 2.1).

The detail level of the contour definition could vary from a simple rectangle to represent the bounding box to a fitted silhouette of the body or face.

Volumetric models represent the three-dimensional appearance of the object (images c and f in Figure 2.1) and the required level of detail depends on the final application. These types of models can be represented using a simple cylinder (or more than one depending on the structure of the object) or by more detailed models that adjust to the peculiarities of the surface structure of the object itself.

During the tracking, the model must fit the specific appearance of the object, so the tracking system must deform the model accordingly. Depending on the object and the tracking method, the deformation system may vary. In the further sections, we list some strategies used to define these models in the specific cases of body pose and face gesture reconstruction and tracking scenarios.

Figure 2.2: Some examples extracted from annotated 2D body pose datasets. Original image extracted from [17].

2.1.1 Human body models

The ability to represent the singularities of the human body pose using essential elements like landmarks and edges makes the structural body models a common choice for body pose reconstruction methods. Each landmark usually represents a body joint and edges the bones or rigid connections between joints. Note that it is not a realistic representation of the human skeletal structure, so not all body bones and joints need to be represented. Thus, the structure does not match with the human bone structure in some cases. Figures 2.2 and 2.3 show some examples of 2D and 3D body representations extracted from manually annotated datasets.

In the case of structural 3D body representations, some articulations (i.e., the spine) could be simplified to avoid representation complexity and reduce computation during tracking. This kind of 3D body representation is also known as *kinematic body representation*, and encodes movement kinematics or pose constraints in some cases [36].

Methods such as [37, 38] use an underlying two-dimensional body structure representation that includes the body trunk (with reduced spine articulation) and limbs. In contrast, other approaches like [24] also include some feature points of the face and head. These methods define the 2D model configuration by locating each joint of the defined structure on the image plane. The projection of the pose in the image plane makes it unnecessary to consider the size of the limbs or the body symmetries. However, in a 3D representation, ignoring these types of elements can lead to asymmetric body reconstructions or inconsistent tracking estimations. Therefore, methods such as [10, 36] propose a 3D kinematic model whose configuration combines global position and orientation parameters with the definition of the angles of each of its joints. Therefore, body movement can be filtered and smoothed during the tracking sequence.

The silhouette of the body contains part of the pose information. For example, methods like [39] use the body silhouette to recover human 3D body pose. However, the information about the limbs of the body suffers when they overlap other body areas like the trunk, for example. Thus, methods like [40, 41] divide the body into smaller rectangular parts and uses the image texture information to detect or track the body structure. The separation of body parts makes it possible to track smaller elements of the body, but it is still prone to tracking errors due to occlusions.

A way to better handle the self-occlusions of the body is to estimate the

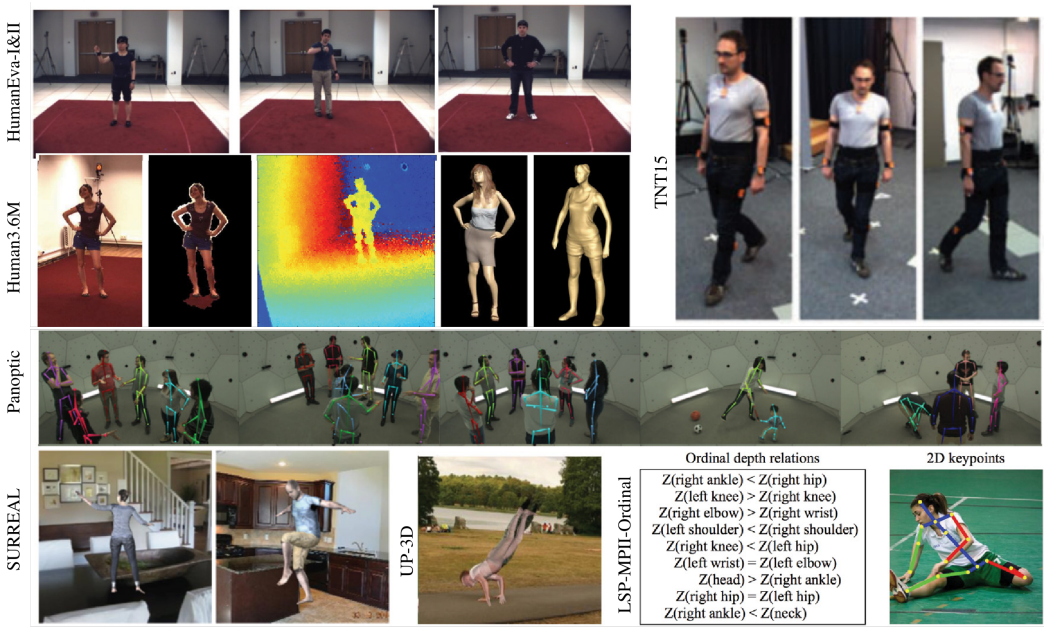


Figure 2.3: Some examples extracted from annotated 3D body pose datasets. Original image extracted from [17].

3D position of the body parts. To this end, methods like [42] propose a similar body structure but using 3D volumes instead of 2D patches. This kind of model can handle the same body motion constraints described for the kinematical model, but also includes extra information about the volume to extract texture information for the tracking process.

However, the appearance of the human body is not limited to the kinematical structure or the body pose. Volume-based parametric body models like SMPL [43] and Stitch [44] describe the pose and external appearance of the human body in a realistic manner, based on a learning process over a dataset of accurate 3D scans of real actors (see Figure 2.4).

SMPL [43] represents the external 3D shape and pose of the human body based on a set of parameterisable blendshapes. With a body pose as input, the model combines the learned blendshapes to generate a realistic 3D body representation for that pose.

Stitch [44] represents the external 3D body pose of the body dividing the body shape into smaller parts. Each part defines the deformability of

Figure 2.4: On the left, some examples of the body poses and shapes using SMPL body model. The image shows the body pose estimated using SMPL model configuration (brown) vs the ground truth 3D scan model (gray). On the right, some examples of the poses and shapes using Stitch body mode. Each colour represents a different deformable body part.

that region based in a PCA subspace and a fitting approach to defining the current surface appearance.

Both models have two types of parameters: the *shape parameters* adapt the external appearance of the model to the appearance of the tracked person and the *pose parameters* define the pose of the model.

Although the realism of the representation offers certain advantages, it must be kept in mind that the computational load of these models is significantly higher than the kinematic models.

2.1.2 Human face models

Structural face representations use 2D landmarks or 3D vertices to define a feature location [45], and the number and location of the features differ depending on the proposed method or reference dataset. A facial feature point is a dominant point describing a unique location of a facial component (e.g., eye corner or nose tip) or an interpolated point connecting those dominant points. However, some landmark definitions could be ambiguous in some cases, such as facial contour points, for example. The specific list of detected facial feature landmarks depends on the method and the implementation.

Figure 2.5 shows some examples of facial feature point lists used by structural face models found in different annotated facial databases. Usually, the representation includes some edges to group landmarks from the same facial element (e.g., eyes, lips or eyebrows) for a better visual interpretation.

For contour-based representations, the most straightforward 2D represen-

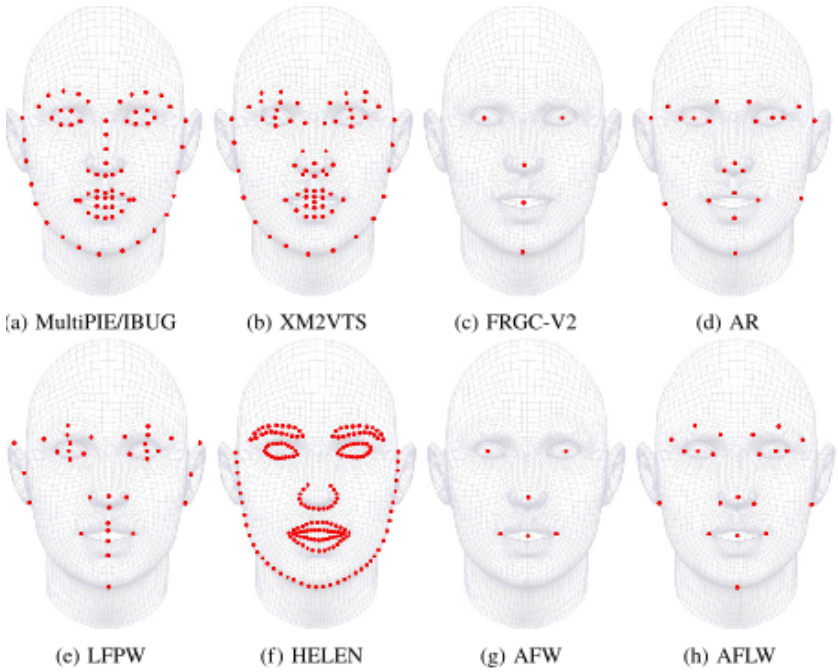


Figure 2.5: Face feature definition examples from different annotated facial image datasets: a) [46, 47] b) [48] c) [49] d) [50] e) [51] f) [52] g) [53] h) [54]. Image extracted from [47]

tation for a face is a rectangle shape representing the face texture boundaries in the image. Facial region detectors like [9, 55, 56] use contour-based representation. Some methods use this region information to perform a secondary analysis to also detect internal face structures like eyes and mouth using specific feature extractors [9, 55]. Other methods [34, 57] define a facial region silhouette and rely on the facial appearance information inside the silhouette to determine facial bidimensional deformation and gesture.

Volumetric face representations reconstruct denser facial representations than those generated by 3D structural models.

As an example of a simplified face pose estimation, the authors in [58] propose a 3D head model based on a single-cylinder and a tracking method to reconstruct the full movement of the head, but not the local face gesture.

To estimate the face gesture, the representation model needs more detailed information about the facial elements like the eyes and mouth. How-

ever, the level of detail and the deformation method must be well considered to maintain the right balance of representability and performance.

For example, the work presented in [59] proposes a facial model based on blendshape deformation [60] to define the face appearance and gesture. The model combines a set of 60 different facial gestures to adapt the original shape of the model to the appearance of the user's face.

Methods in [61, 62] generate a user-specific 3D face model using a user's video sequence as input. The generation process uses a generic blendshape based 3D model with a predefined gesture deformation space. The proposed method relies on an integrated PCA deformation space and a specific regression method to adapt the shape of the generic model to that of the users' face. Moreover, it includes an extra deformation layer to encode skin level deformations such as wrinkles.

In [63], the authors propose a 3D deformable face model with different levels of vertex densities (see Figure 2.6). This type of model distributes the computation required to fit the final dense model into different stages. The initial estimation uses the smallest representation level (with fewer vertices), and each stage updates the estimation of the previous one, having fewer parameters to adjust in each step.

In [64], the authors propose a fully convolutional deep neural network to estimate the dense facial shape by a set of dense template grids. Although the experimental results show the excellent accuracy of the reconstruction, the computational requirements are considerable, especially for environments with limitations.

2.2 3D object tracking

Model-based 3D tracking approaches fit a 3D deformable model to the object to be tracked, adapting the external shape or structure of the representation model to the current configuration of the tracked object.

Considering the possible types of object deformations, we can highlight two that are most representative:

- Rigid multibody deformations.
- The non-rigid deformations.

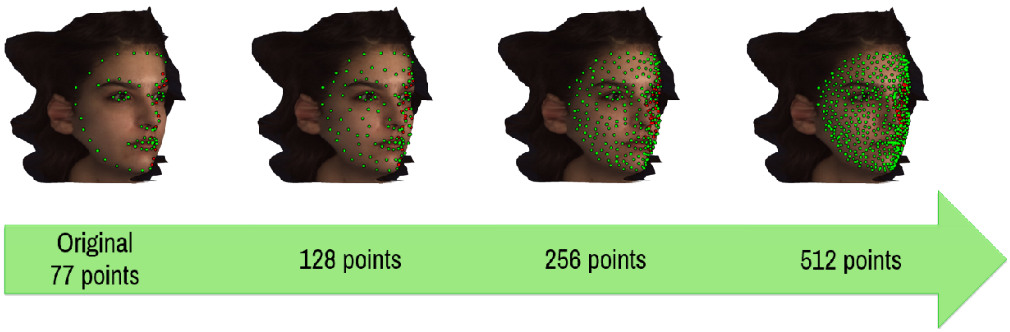


Figure 2.6: Different 3D vertex densities of the deformable face model proposed by [63].

2.2.1 Human body as a rigid multibody deformable object

An object with rigid multibody deformations behaves like a series of rigid elements interconnected by joints. The angle variations in the joints generate changes in the position and orientation of the component chain connected to that joint. The human body meets this definition since the bone structure of the human body defines a series of rigid elements that we can move to change the body pose. There are several examples of 3D body models [65] that vary depending on the needs of the method in question.

Many studies [10,20,36,66,67] use 3D kinematic body models to estimate the 3D human body pose. Some approaches [20, 66] separate the 3D estimation process in two different tasks. The first task uses a 2D body pose detector like [25, 68, 69] to estimate the body pose in the image, and the second task fits a 3D model using those 2D locations as reference.

The work presented in [66], computes the 3D model configuration using an expectation-maximisation algorithm to generate a sparse dictionary based on a set of heat-maps generated by the 2D joint locations from the first step.

In [20], the authors propose a pre-trained 3D pose estimator that generates a set of possible 3D body poses used to refine the given 2D landmark estimation and evaluate the depth of each landmark. However, these kinds of estimation methods must deal with the ambiguity of specific projections in the 3D fitting process.

An essential problem in the context of 3D body pose estimation is the

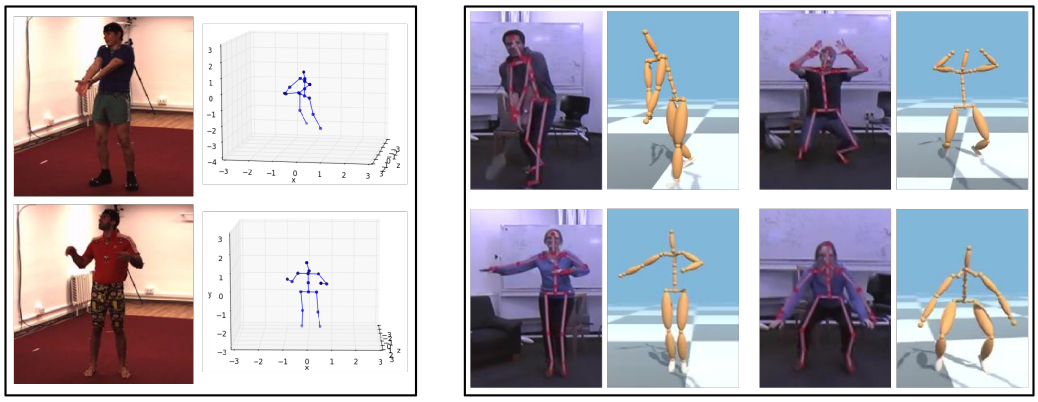


Figure 2.7: On the left, 3D body pose estimations using [36]. On the right, 3D body pose estimations using [10].

lack of in-the-wild 3D body pose datasets. The accurate capture of the 3D pose information requires complex setups and a controlled environment. In the case of two-dimensional images, it is likely to estimate person poses from monocular images taken in uncontrolled contexts. A human operator could do this task manually, for example. However, the interpretation of 3D human body poses in monocular images and the configuration of a 3D model to fit it is not such straight forward task, even for a human.

Trying to fill this gap, the work presented in [70] uses a 2D body pose estimation Deep Learning (DL) model as a reference and re-adapts it to generate the 3D pose estimation. This process combines the 2D estimations of the pre-trained model with a 3D human body model to generate the new inference model. The final model estimates the 2D and 3D body poses directly from a single image.

As another suggestion to fill the lack of annotated data, in [67], the authors propose an encoder-decoder based network to generate 3D human pose estimations from multi-view datasets in a semi-supervised way. The same article describes using the generated data to train a 3D human pose estimator for monocular images.

The work presented in [36] proposes an approach to improve the training process of 3D human pose estimators using a kinematic structure to constrain the used poses. The approach suggests inserting a kinematic 3D model of the object into the training phase of the CNN, which on inference estimates

the configuration of the model directly from a monocular image (Figure 2.7 left).

Similar to kinematic model-based methods, in [71], the authors propose fitting an SMPL model to a set of estimated 2D joint locations. This method uses an optimisation-based approach to recover only the SMPL pose parameters, leaving the shape parameters constant.

The estimation of pose and shape parameters for SMPL model extends the reconstruction complexity, and thus the computational needs. In response, some studies such as [23, 72, 73] use extra intermediate estimations to lessen this complexity. In [23], the authors propose using 2D joint estimations, and body silhouettes, the work presented in [72] uses a 2D body part segmentation map, and the method in [73] uses a combination of 2D body pose estimation, 2D body part segmentation, and 3D pose.

Furthermore, other methods [74, 75] combine different three-dimensional parametric models (i.e., structural and volumetric) to reduce the fitting complexity. Both propose using a realistic 3D model, rigged with a skeletal structure for the pose deformations. They use the skeletal structure as a kinematic representation to reduce the ambiguity of the pose, while in parallel, use the non-rigid deformations to reconstruct the non-rigid deformable elements like body shape and clothes (e.g., loose skirts and shirts).

Although the results using volumetric models are closer to the real appearance of the subject's body, the amount of extra computation needed to reconstruct the exterior appearance is considerable. Therefore, they are not suitable for environments with computational limitations.

2.2.2 Human face as a non-rigid deformable object

Objects with non-rigid deformations are those that can experience changes in appearance that do not follow a rigid pattern. The human face is a representative example of an object with this kind of deformation. Even if the face has a rigid base defined by the underlying bone structure (i.e., the skull and jaw), the facial muscles distort the appearance of the face without any rigid pattern (see top row of Figure 2.8). Moreover, the face shape changes significantly between individuals, even when they have similar characteristics like gender, age and ethnicity (see bottom row of Figure 2.8).

In order to accommodate this variability, methods such as [77, 78] directly record the user's appearance in an initialisation phase, fitting a 3D deformable object to the user's face. During the tracking sequence, it adapts

Figure 2.8: Examples of face deformations caused by facial cardinal expressions (top row) and examples of different human faces (bottom row). The facial expressions in the top row are (from left to right): neutral, happy, surprised, sad, afraid, disgusted and angry. Images extracted from KDEF dataset [76].

the model configuration to the next frame, updating the registered appearance with each step. This approach has the advantage of not needing a previous training phase (it is a learning-free method), but it is prone to losing the tracking due to changes in the user's appearance (e.g., if the user wears glasses during the tracking or light conditions change). This approach is computationally efficient but is not robust enough to apply in uncontrolled environments.

Face tracking methods like those proposed by [19] rely on structural 3D face models to estimate the face shape and gesture (see Figure 2.9 left). The method in [19] uses previously trained regressors to fit the projection of the 3D vertices of a facial model on the face image using image patches around the projections for the computation. These types of methods are computationally fast but represent only the essential features of the face, and they do not reconstruct facial details.

Other methods like [79] propose reconstructing the entire face shape using a dense 3D model and photometric information (see Figure 2.9 right). This method is generalisable to other deformable objects other than the face, but the optimisation and tracking system requires an expensive computation, and generated surface details are limited.

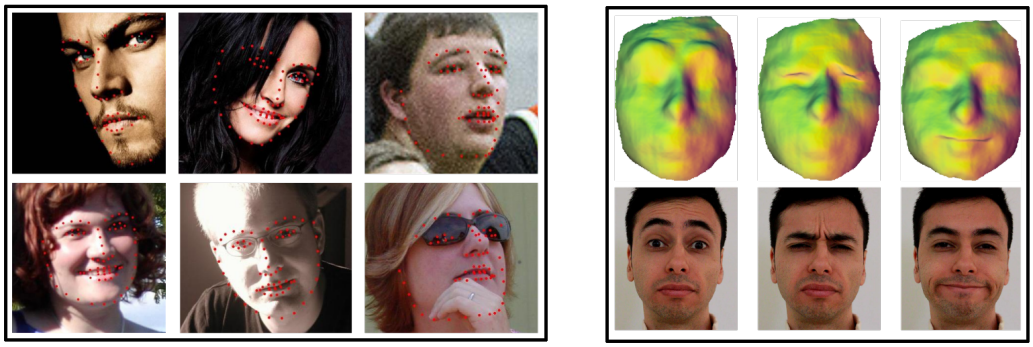


Figure 2.9: Left image shows some examples of the fitting results using [19]. Right image shows the reconstruction of the 3D shape of a face using [79].

Methods like [7, 59, 62, 63, 80, 81] present a better balance between reconstruction level and performance. What these methods have in common is the use of a first estimation of the 2D feature points of the face, followed by a second adjustment step of a deformable 3D model.

The method in [59] uses a user-specific 3D deformable face model, with a previously defined set of facial gesture deformations. This method requires a previous calibration and training step (around 40 minutes in total, according to the authors). In this step, the user reproduces each of the gestures in the model while a 2D feature point detector tracks the facial features. The training process relates the 2D features captured during the calibration with the gestures in the 3D model, building a specific regression model for the user. During the tracking, the method achieves real-time performance (∼24fps) in a desktop personal computer (PC). In [82] the authors adapt the method to run on a computationally limited device like a smartphone (Motorola MT788 cell phone), maintaining the real-time performance. In [83], the same authors modify the method in [59] to avoid the user-specific calibration and training step for new users. Instead of calibrating for each user, they use the calibrations of an initial user set and adapt their regression model to the new user, considering the differences in the 2D face feature capture step.

In [7, 63] the authors propose using a 3D model with a different level of vertex densities to reconstruct the face shape accurately. In the first instance, the detected 2D facial features help to adapt a coarse 3D deformable face model with a limited amount of 3D vertices. Later iterations increase the number of model vertices sequentially, increasing the level of detail in

facial reconstruction. Each iteration reduces the reconstruction uncertainty for the next, also reducing the computation for the iteration. The proposed experiments show excellent performance on a desktop PC for the shape reconstruction process. Additionally, the method in [7] also reconstructs the face representation texture to improve the final reconstruction, which adds critical computation times. However, both methods allow for the adaption of the density of the model and remove the texture generation step to improve the final tracking performance.

In [80], the authors propose an efficient method to fit and reconstruct a 3D face model using least-squares optimisation to reconstruct the shape, light direction, light strength and albedo. The approach decouples the geometric and photometric optimisations to simplify the optimisation and thus reduce computational needs. Although the method extracts the proposed features efficiently, the entire process is still computationally expensive, taking more than two seconds to process a single image. These computation results make the method unsuitable for real-time tracking.

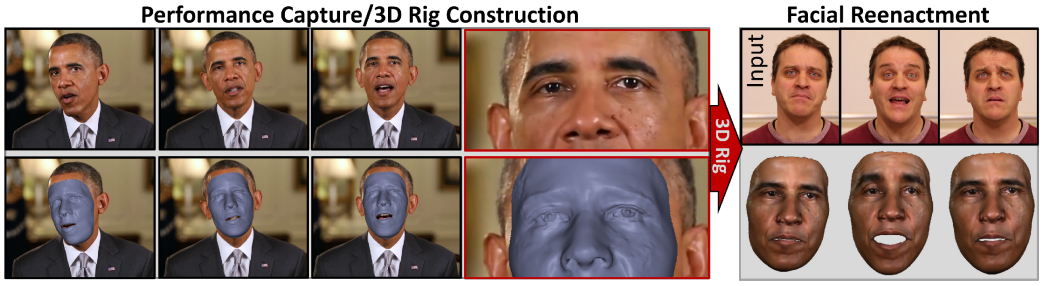


Figure 2.10: On the left, some examples of the 3D model generated by [62], and on the right, the results of using this model with a different actor.

Previously described methods rely on a generic deformable model, which adapts to the shape of the user's face but does not represent facial details. However, the method described in [62] proposes a method to extract the 3D face shape, including particular details like gesture wrinkles (see Figure 2.10). This method relies on a previously generated 3D model which encodes the user-specific shape, the user's expressions and expression related details such as wrinkles in a single blendshape based 3D face rig. The generation of this user-specific model is fully automatic, and only needs a monocular video sequence of the user as input without a specific expression sequence. However,

the high accuracy of the method requires a huge amount of computation during tracking, spending more than a minute for a single frame.

Similarly, and to reduce the computational cost, in [84] the authors propose a method with fewer facial details, but with real-time tracking performance. They suggest generating a user-specific face model with user-specific face deformations, and a dense photometric consistency measure to track the users face in real-time. Despite the real-time performance of the tracking, the user-specific model generation makes this approach non-viable in uncontrolled environments, and the photometric based tracking used by these methods can suffer tracking errors and artefacts in the presence of hard shadows.

More recently, the work presented in [85] proposes a Deep Neural Network (DNN) based method to reconstruct the dense shape of a face without a reference 3D deformable model, achieving real-time performance (9.8ms per image). This method gets impressive results reconstructing faces from monocular images even with occlusions and illumination changes. However, it needs a dedicated GPU to achieve real-time performance, which is not available in most cases.

As an example of a trade-off between reconstruction accuracy and real-time performance, in [81] the authors propose a generic region-based deformable 3D model and a stabilisation method to improve the stability during tracking (see Figure 2.11). This user agnostic method shows a stable tracking performance ($\sim$20 fps) on a smartphone (iPhone 7), maintaining a functional reconstruction of the face gesture and the facial boundaries. It relies on a set of detected 2D feature points and an optical-flow analysis of the pixel values during the video sequence, combined with a non-linear least-squares optimisation process. Additionally, it includes a stabilisation step to correctly adapt the pose estimation to extreme gestures or out-of-plane orientations.

Table 2.1 summarises the main face tracking methods presented in this section and the main features of each one.

2.2.3 Methods based on Deep Neural Networks

Analysing the methods listed in the previous sections, we can see that DNN-based methods have progressed in recent years. Surveys presented in [17] and [86] show the evolution of the human body pose tracking and human face tracking methods based on DNNs. Although the computational load of

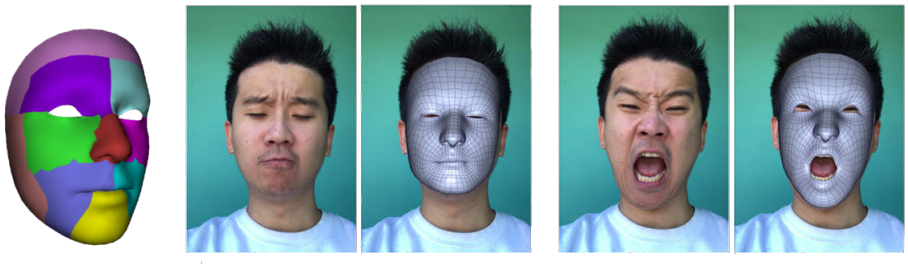


Figure 2.11: The left image shows the region-based deformable 3D model proposed in [81]. Right images show two examples of 3D model fitting on a face image.

this type of approach is still very high, we can find increasingly more efficient neural models and optimisation methods.

One of the elements that determines the performance of the final model in this type of method is the backbone, which defines the network architecture. Studies such as [87] show the relationship between performance and accuracy for a broad set of network architectures in different scenarios.

Table 2.2 summarizes the methods presented for the estimation of the body pose in previous sections. In this context, the most used backbones are ResNet and Hourglass [88]. In the case of face reconstruction, the backbones are more diverse. Table 2.3 shows the most notable methods in this field.

2.3 Facial gesture recognition

Both, the pose of the body and the facial gesture, represent a large part of non-verbal communication. In particular, the facial gesture represents most of our mental or emotional state. From the analysis of these gestures and emotions, we can estimate the state of inattention, level of liking or other interesting indicators for multiple applications. However, the great diversity of facial shapes and possible deformations make it difficult to define those facial gestures.

In the last decades, various notations have been proposed to define facial gestures. Summarising, we can point to three main notations: a) gestures based on micro-expressions (commonly called Action Units or AUs), b) gestures based on cardinal expressions (e.g., happiness, sadness, surprise, fear,

Table 2.1: The main features of the presented face tracking methods.

	Dornaika et al. 2006 [78]	Baltrusaitis et al. 2013 [19]	Cao et al. 2013 [59]	Garrido et al. 2013 [61]	Weng et al. 2013 [82]	Cao et al. 2014 [83]	Jeni et al. 2015 [63]	Thies et al. 2016 [84]	Huber et al. 2016 [7]	Garrido et al. 2016 [62]	Yu et al. 2016 [79]	Hu et al. 2017 [80]	Feng et al. 2018 [85]	Cao et al. 2018 [81]
Features	✓	✓	✓	✓	✓	✓		✓	✓			✓	✓	✓
Dense Geometry							✓		✓	✓	✓	✓	✓	✓
Dense Color				✓			✓	✓	✓	✓		✓	✓	
2 step method	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓		✓
1 step method											✓		✓	
Needs training		✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓
User specific	✓		✓	✓	✓			✓		✓				
User agnostic		✓				✓	✓		✓		✓	✓	✓	✓
Offline				✓					✓	✓	✓	✓		
Online	✓	✓	✓		✓	✓	✓	✓					✓	✓
Needs GPU								✓					✓	

anger and disgust), and c) gestures based on a combination of local expressions (e.g., smile, kiss, blink and brow rise).

The methods based on micro-expressions [91–94] combine elemental facial movements to generate more complex gestures (e.g., movements of several muscles of the mouth to define the smile gesture). P. Ekman, W. Friesen and H. Hager [95] proposed the Facial Action Coding System (or FACS) as a naming system to define the gestures and facial movements, based on what they call Action Units. Figure 2.12 shows some examples. The AUs are minimal facial expressions based on individual muscle movements defined as qualitative values (i.e., if there is muscle contraction or not). Each AU has limited expressiveness, but the combination of AUs represents complex

Table 2.2: Main studies based on Deep Learning approach for human 3D body pose reconstruction and tracking.

Method		**Output**	**Backbone**
Bogo et al.	[71]	SMPL model pose configuration	DeepCut [89]
Zhou et al.	[36]	Kinematic model pose configuration	Hourglass
Tome et al.	[20]	Pre-trained body pose model configuration	CPM
Mehta et al.	[70]	Kinematic model pose configuration	ResNet
Pavlakos et al.	[23]	SMPL model pose and shape configuration	Hourglass
Omran et al.	[72]	SMPL model pose configuration	RefineNet
Varol et al.	[73]	SMPL model pose and shape configuration	Hourglass
Rhodin et al.	[67]	Pre-trained body pose model configuration	Hourglass

Table 2.3: Main studies based on DNN methods for face tracking.

Method		**Output**	**Backbone**
Zhang et al.	[90]	Pose, shape, expression	TCDCN
Deng et al.	[9]	Pose, shape, expression, texture, illumination	ResNet
Feng et al.	[85]	pose, shape	Custom CNN

gestures.

The methods based on emotion or cardinal expressions [96–98] use a set of six complex emotional gestures - happiness, surprise, sadness, fear, disgust and anger- proposed by P. Ekman, W. Friesen and H. Hager [95] to define a wide range of facial gestures (see top row of Figure 2.8). Although nowadays, the detected emotion types are richer, these are the most widely used.

The methods based on face region expressions can be seen as a middle ground between the other two. Such methods estimate local zone-specific gestures and combine them to generate the global face estimation as a complete expression [24,90,99]. For example, they extract the local gesture of the mouth from the mouth region, and the eyebrow gestures from the eyebrow

Figure 2.12: Action Units #2 (left) and #10 (right). Top row shows the rest state of the action, while the bottom row shows the active state.

region. Then, combining all the extracted gestures, define a full-face gesture representation. This approach reduces the gesture estimation complexity to a specific facial area, reducing the complexity and narrowing the dependence with the rest of the facial elements.

Some methods improve the limited gesture extraction by using only feature point movements, including the analysis of face texture using the face feature points as a reference in some cases.

The method proposed in [100] analyses face texture using a set of previously detected facial feature points and a Histograms of Oriented Gradients (HoG) analysis to extract the gesture features. With both type of features as input, the authors propose training a specific AU regressor to estimate the gestures. Experiments show real-time performance on a desktop PC, but there are no measurements on computationally constrained devices.

In [96], the authors propose a specific Convolutional Neural Network (CNN) model to extract the facial gesture directly from the face image (without facial feature points) and apply it directly to a 3D virtual avatar with gestures designed by a 3D artist. The method proposes training two CNN models, one to extract the gesture of the actor and a second one to represent the expressiveness of the virtual avatar. With both models, they suggest a transfer learning technique to directly define the motion transference between the user's gesture model and the avatars deformation model. This strategy improves the expressiveness of the avatar. However, it requires a training step for each generated avatar in order to get the specific deformation model for the character and a new motion transference model.

In addition, the method in [93] proposes first to detect the facial feature landmarks of the users' face and then estimate the facial expression using the face texture with the landmark location as reference. In this second stage, a set of Gabor filters analyse the area around the feature points. This method uses information that feature point detection is not using (like skin wrinkles, for example).

In [90], the authors outline a method to combine landmark detection and expression estimation in a single step using a multi-task learning approach. This method takes advantage of the correlation between facial landmark positions and facial expressions, improving the estimation of each one in the process. They demonstrate the improvements given by their approach, but the number of expressions extracted is limited.

2.4 Computationally limited systems

We define a computationally limited system as a hardware device with constrained data processing capabilities which potentially misses requirements of the assigned tasks. This definition includes hardware elements that, although they may seem powerful (e.g., desktop PCs or computation servers), the assigned tasks exceed their capabilities. As part of its limitations, there are also restrictions in scaling the processing capabilities to fit the criteria. Thus, the efficiency of the assigned tasks is something to consider.

Two representative examples of a limited system are the onboard computing hardware used in the automotive industry and broadly used smartphones. The automotive area is quickly evolving and including more complex onboard Advanced Driver Assistance Systems (ADAS) with a high degree of computation. ADAS involves the human behaviour analysis inside and outside the cabin [101], for example. Furthermore, in the age of autonomous driving, Driver Monitoring Systems (DMS) need to analyse the driver's behaviour to decide if it is safe to transfer driving control to the user as shown in [102].

In the case of smartphones, current devices include complex face identification [103, 104] and tracking [81, 82, 105] features to enhance security, human-machine-interaction and several multimedia applications. Some of those systems include computationally demanding methods to run in devices with limitations in computation and power supply.

A typical example of a computationally limited system is an embedded system. An embedded system is a computer system designed for a dedicated

function inside a more extensive system. Originally each embedded device was specifically designed and implemented for a particular task, but nowadays there are generic hardware components that can be adapted to a specific task using specific software.

We can categorise the most typical embedded systems in various groups:

- Custom boards
- Specialised PCs
- Single-board computers (SBC)
- Computer on module (CoM)
- System on module (SoM)

Custom board systems are circuits explicitly designed for a specific task. They can contain a complete computer system similar to a PC or just the elements necessary for the intended job. These are fitter systems, but as they need specific manufacturing processes, the production price is something to consider.

Specialised PCs are computers with distinctive elements (e.g., case, refrigeration, enclosure) for the final application environment. For example, ruggedised PCs are designed to operate reliably in harsh environments, such as dusty places, in high temperatures, or wet areas. They are commonly adopted by police, firefighters, as well as military and some industrial applications. They can include standard hardware pieces, which reduce production costs, but they are bigger and have higher power consumption.

A single-board computer (SBC) is a generic computing system that includes all the necessary computing elements (i.e., CPU, RAM or storage), connection controllers and standard connectivity ports (e.g., USB, HDMI, Ethernet) in a single circuit board. Typical SBC implementation like that shown in the left image of Figure 2.13 integrates all the computing elements and connection controllers in a single circuit called system-on-chip (SoC). SoCs are more energy-efficient than multi-chip implementations with equivalent functionality, and they are very common in the industry for smartphones and edge computing devices.

A computer-on-module (CoM) is a subtype of SBCs system which integrates all the elements in an SBC except the standard input/output connectors. They include a connection bus which combines all the connectivity

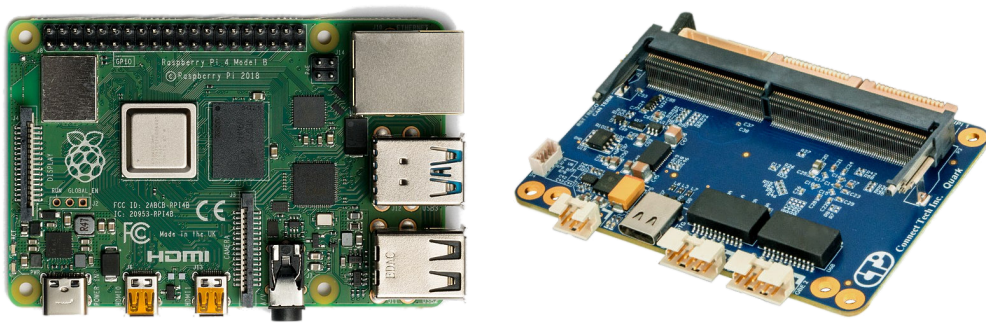


Figure 2.13: On the left, a Raspberry Pi 4 Model B with a Broadcom BCM2711 SoC. On the right, a carrier board designed by "Connect Tech" for the Jetson Xavier NX from Nvidia.

elements of the board (e.g., USB, Ethernet), and usually needs a carrier board (similar to the example shown in the right image of Figure 2.13) to split the bus into standard connectors or to interact with the rest of the system.

A system-on-module (SoM) is very similar to a CoM, including the same or similar elements. However, it has a stronger encapsulation to reduce power consumption and heat emission. SoMs rely on SoC and similar components to increase the encapsulation. SoMs are the most common hardware structure used for onboard computation and edge computing. Some examples are Nvidia's Jetson Xavier [1] (Figure 2.14 left) or Google's Coral [2] (Figure 2.14 right), which also include specific hardware to speed up neural network processing.

Limitations can be internal or external. Internal limitations refer to the inherent capabilities of the hardware itself. The lack of internal CPU frequency, constrained system memory size, or network bandwidth limitations are some of the possible factors that limit computation.

In the case of machine learning computations, specific computing hardware can reduce bottlenecks. Elements like Tensor Processing Units (TPU)

[1]Nvidia Jetson Xavier NX SoM official site (11th of May 2020): https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-nx/

[2]Google Coral SoM official site (11th of May 2020): https://www.coral.ai/products/som/

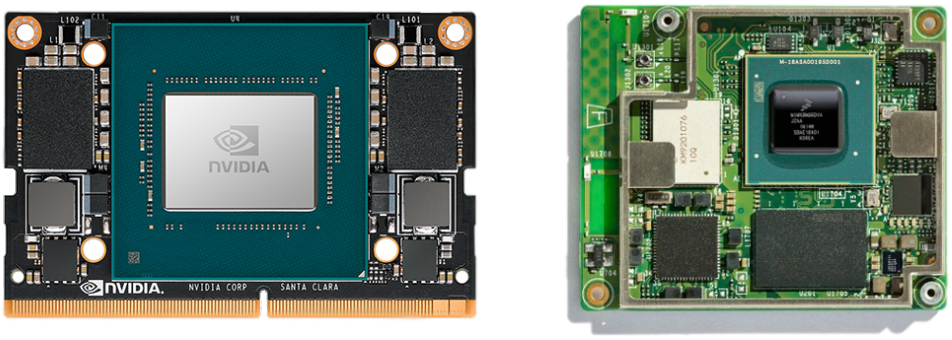


Figure 2.14: On the left the Jetson Xavier NX SoM from Nvidia. On the right the Coral SoM from Google.

included in Google Coral (see Figure 2.14 (right)), Vision Processing Units (VPU) included in Mustang form Intel or Graphics Processing Units (GPU) included in Jetson device series (see Figure 2.14 (left)) for example, can perform some computation tasks related to DNNs more efficiently. Furthermore, some commercial implementations of the named elements include standard interfaces like USB (see Figure 2.15) or PCI connections to expand embedded devices.

Figure 2.15: On the left, Movidius Myriad-2 USB AI-accelerator (VPU) from Intel. On the right, Coral USB AI-accelerator (TPU) from Google.

External limitations refer to the constraints imposed by the scope of the final application. Some representative examples of external constraints are power supply limitations, heat radiation thresholds and production cost limits, among others. Power Usage effectiveness is a critical element of autonomous environments (e.g., automotive onboard systems or smartphones) because significant consumption rates drastically reduce the autonomy of the batteries. In some safety-critical system environments like aerospace control systems, computing elements have a strict heat emission threshold.

Wide-scale adoption of these kinds of devices by the industry and research communities encourages performance analysis of different inference techniques in several scopes and hardware architectures. Benchmark suits like [87] allow for the comparison of the performance of machine-learning models on different hardware setups.

2.5 Image capture and 3D reconstruction

A critical point for monocular camera-based methods is the projection scheme adopted by the approach. In a real scenario, a monocular camera projects the captured scene data (i.e., captured light colour) into a single plane, losing depth information in the process.

The pinhole camera model describes this projection mathematically, being close to the physical behaviour of the capture process. In this model, the light rays that are reflected in the object pass through a single point (i.e., a hole) before being projected in the projection plane. Figure 2.16 shows a simplified representation of this model.

The pinhole model describes a *perspective* projection of a 3D point in the real scenario (P) to a 2D point in the image plane (p). Essential elements to compute the projection are the intrinsic camera parameters and the extrinsic camera parameters, mathematically described as a matrix in both cases [5]. The intrinsic camera matrix describes the geometric property of the camera. This matrix includes the focal length (f), which is the distance between the camera centre and the principal point (i.e., the centre of the projection plane), and the principal point offset (x_0, y_0), which defines the size of the projection image plane. The extrinsic camera matrix describes the 3D location and 3D orientation of the camera for the reference coordinate system of the 3D scene. This information is essential in understanding the scene since it is necessary to differentiate whether a particular pose is being performed with

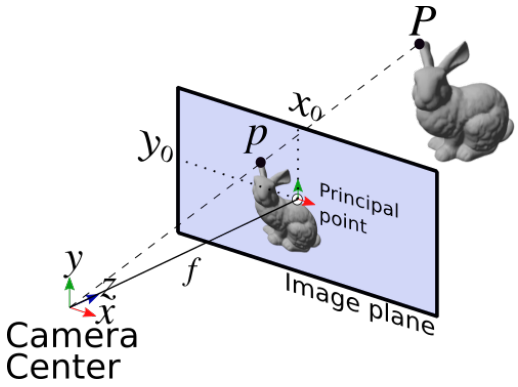


Figure 2.16: Simplified representation of the pinhole camera model.

the individual standing or lying down, for example.

The appearance of a projected element using *perspective* projection varies, mainly depending on the value of the focal length (f) as shown in Figure 2.17. To have a reference within the spectrum of possible focal lengths for *perspective* projection, the focal length of the cameras mounted on Iphones 7+, 8+ and X would be equivalent to a 28 mm full-frame camera, like those represented in Figure 2.17.

Other projection approaches, like the *orthographic* projection, assumes that all light rays coming from the object are parallel to each other and perpendicular to the image plane. Following the model proposed in Figure 2.16, this could be interpreted as a focal length with infinite value.

An *orthographic* projection represents the three-dimensional nature of the object. However, it does not represent the object as it would be perceived by a human observer or a monocular camera. Considering the infinite focal length value as a reference for orthographic projection, and the examples shown in Figure 2.17, it can be seen that the appearance of the captured face varies significantly between both types of projection.

Moreover, consisting of all the captured parallel rays, the object projection has the same size independent of the distance of the object with respect to the camera. Trying to introduce a distance representation, *weak-perspective* projection combines an *orthographic* projection with a scale factor to change the size of the projection. A *weak-perspective* projection assumes

[3]Image taken from (18/5/2020): http://stepheneastwood.com/tutorials/lensdistortion/strippage.htm

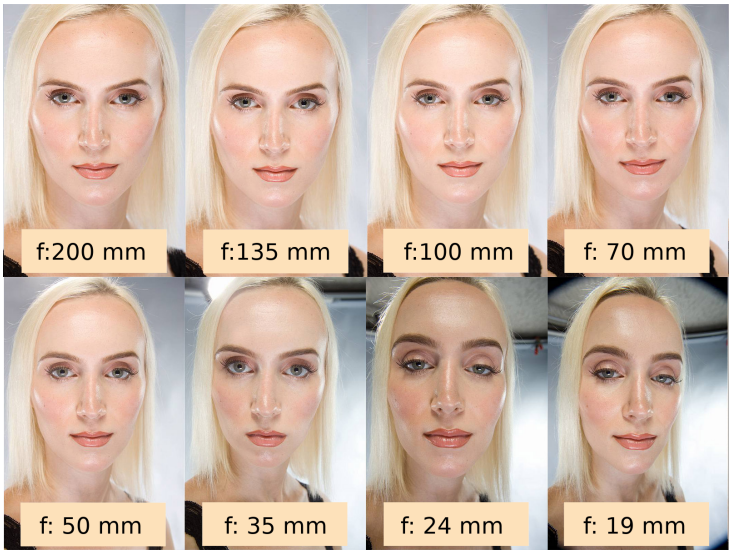


Figure 2.17: Different images of the same face taken with a full-frame camera and using objectives with different focal lengths. Original image by Stephen Eastwood[3]

that the depth values of the object are significantly smaller than the distance between the camera and the object, and thus, negligible. However, this assumption is not right in many scenarios, like face tracking applications used in smartphones, where the object (i.e., the face) is too close to the camera. Even this approach is closer to a *perspective* projection; the appearance of the projected element does not represent the object as perceived by an observer because the focal length of the capture is not considered.

Face images in Figure 2.17 shows how the appearance of a captured element could vary depending on the focal length. Some 3D model reconstruction methods use weak-perspective (e.g., [61, 106]) or normalized projection spaces (e.g., [38, 63, 64]) with a fixed focal length value. Even if these reconstructions do not estimate the position of the object with respect to the camera, the initial reconstruction can be placed in the 3D space using methods like [107], similar to that proposed by [6]. However, the initial reconstruction could not fit correctly with the object due to perspective deformation, causing fitting errors and tracking artefacts.

2.6 Discussion

The examples presented in this chapter show that many methods have been proposed to solve the problem of tracking deformable objects. Depending on the final application, this tracking can be performed on the image plane as a two-dimensional object, in a reduced three-dimensional space, or a fully three-dimensional space. Some of the methods track a series of points (e.g., facial landmarks, body pose structure) in two or three-dimensional space. In contrast, others can extract details from objects, such as the volume of body parts or even wrinkles caused by certain facial gestures.

Additionally, this chapter described methods dedicated to the estimation of the facial gesture, detecting facial gestures belonging to one of the proposed categorisations (i.e., micro-expressions, cardinal expressions or a mixture of both). This information improves the understanding of the scene, providing an emotional context to the captured body or face movements, and therefore, making its interpretation deeper.

However, within all these methods, few studies consider the possible limitations related to the integration of these methods in final real-world setups. Limitations in critical aspects such as energy consumption, computing power or the size of usable memory are elements to consider when using a method in a real scenario. These limitations can make a tracking method unfeasible in specific situations or contexts.

This study focuses on solving the problem of tracking three-dimensional deformable objects in a three-dimensional environment, keeping computational consumption as low as possible. In addition, it includes various experiments to ensure the viability of the proposed methods in setups with limitations such as those discussed in this chapter.

Chapter 3

Feature point detection

The first step in the proposed pipeline for 3D object tracking is the extraction of feature points of the object, either face or body. Further steps strongly rely on these feature points, so their precision and accuracy determine the performance of the end result.

Although this step is common for both types of objects, this section will focus on the detection of the facial feature points. Same or similar methods can also be applied to the scope of body pose detection. This chapter introduces two computationally efficient methods for extracting the facial feature points from monocular images.

The first method is a learning-free method based on a local image gradient analysis. This approach focuses on scenarios with extreme memory and computation limitations, like those where even loading a statistical or regression model in memory could be a problem. It analyses the image gradients on each internal face element region (i.e., eyes, nose and mouth) to determine the location of the feature points. Even if other methods based on sophisticated features and regressions show better accuracy than the presented approach, the extreme computational lightness of the presented method could make it the only choice in most restricted scenarios.

The second proposed method is a multi-level, multi-modal and multi-task method (named M3) based on a DNN approach which results in the detection of facial feature points and the estimation of local face gestures (e.g., smiling, kissing or frowning). A set of neural networks, divided into various estimation levels, combine different data inputs (image and head orientation) for the estimation using a multi-task strategy. The computation levels reduce the estimation space to smaller facial zones thereby reducing

variability and solving the estimation with specialised zone-specific models. Each model takes advantage of synergies between estimated attributes (i.e., landmarks and gestures) to improve the final performance of both elements using a multi-task strategy. The combination of the models provides a robust and lightweight facial point detection system able to run on computationally limited systems such as smartphones with real-time performance (∼30 fps).

3.1 Related work

Within the scope of automatic detection of facial feature points, there are many and diverse methods to detect characteristic points in monocular images [45, 108].

A widely adopted method was presented in [109]. This method proposed iteratively improving an initial rough estimation of the face, analysing the region around each landmark to improve the previous location with each repetition. On each iteration, a two-dimensional structural model constrains the new locations considering all the landmark set to avoid unnatural or impossible point distributions.

Later, studies [18, 110] suggest using specialised regressors to detect the face landmarks, dramatically improving the performance of the process. They both assume that the regression method includes the global facial shape constraints for the entire facial structure, avoiding the use of a shape verification process. However, in both cases, accuracy suffers in the presence of out-of-plane head rotations.

To include the nature of the three-dimensional faces in the estimation process, other methods like [38,111] propose to use a three-dimensional point distribution model.

In [38], the authors propose a method to estimate all facial landmarks using a DeepLearning (DL) based regression model. To improve computational performance, they adopt a binarization strategy, which reduces the computation time at the cost of some precision. Despite the binarization process, the fitting method is still computationally expensive for computationally limited hardware.

Baltruslaitis et al. [111] propose a method based on [109], using a specialised DL based regressor to analyse the image patch of each landmark. It also includes a 3D shape model to constrain the global point distribution, achieving real-time performance (∼30 fps) on a desktop PC.

As a real-time DL method, study [90] outlines the extraction of facial features and a set of facial gestures using the same computation adopting a multi-task strategy. Based on a direct relationship between the landmark locations and the face gesture, the proposed method uses the facial gestures to improve the landmark estimation, and the landmarks to estimate the face gestures. This method achieves real-time performance on a desktop PC with a limited set of facial gestures.

Other proposals like [9] include landmark detection as part of the face region detection process in the entire image. This strategy reduces the time needed to detect landmarks because region detection includes part of the feature extraction computation. However, the detection step is inseparable, leaving out approaches based on tracking to reduce the significant computation needed by the detection task.

3.2 Face landmark detection by image gradient analysis

In this section, we propose a learning-free approach for detecting facial features that can retain the advantages of both learning-free and learning-based approaches. In particular, the advantages of learning-based approaches (i.e., rich set of facial features, real-time detection, accurate localization) will be retained in our proposed approach. In addition, the proposed approach will have the two advantages that are associated with learning free approaches. First, there is no tedious learning phase. Second, unlike many learning approaches whose performance can downgrade if imaging conditions change, our proposed approach is training free and hence independent of training conditions.

3.2.1 Lightweight facial feature detection

This method separates the facial region given by a face detector into smaller facial zones and estimates the landmarks in each zone separately. Thus, the proposed approach requires an initial stage of face detection, which, depending on the approach taken, might also require a facial training stage, such as [112, 113]. Nevertheless, these face detection techniques do not limit the search as much as the facial feature detection methods would do under different lighting conditions, and with different facial expressions and appearances;

therefore, we consider them acceptable for our purpose. Furthermore, these approaches have proved to be robust and do not need any retraining. We can also apply the same detection techniques (i.e., [112, 113]) for localizing facial parts such as the eyes, nose and mouth. However, we do not consider their detection as a *strict* requirement because we also consider low-resolution facial images or partially occluded ones, which would prevent the detectors from finding them correctly. However, we include these as potential reinforcing checks since they can locate the facial parts with higher precision in more favourable circumstances.

Figures 3.1 and 3.2 and Algorithm 1 show the whole fitting process step by step, where the term *ROI* refers to a *region of interest* (the sought region) and *SROI* to a *search ROI*. The input data related to the eyes, nose and mouth can be *ROI* or *SROI*, depending on whether the corresponding object detector has already detected them or not, as mentioned above. Algorithm 1 aims to detect 32 facial points in any input image (Figure 3.3). Their 2D positions are fixed within their corresponding regions, taking into account the scale of the found local regions and the in-plane rotation of the face (roll angle). Thus, by finding the *ROI* of a face part as well as the roll angle, the 2D points of that face part will be automatically and rapidly estimated. This process is good enough to allow an OAM tracker such as [78] to fit a 3D model on subsequent frames with a visually alike correlation between the model and the face images (Figure 3.2-(6)). This is especially the case of contour points, which help in the initialization but do not match with real landmarks, which cannot be determined with a high degree of certainty on a face image even by trained observers. Once a face region has been located on an image (e.g., using [112, 113]), all 32 point positions are always estimated, even if they are not visible in the image, due to occlusions.

First, the eye points are estimated, then the eyebrows, followed by the mouth, the nose and finally the contour points. The search regions derive from the detected face and eye regions (Figure 3.1). In case *eyeROIs* have not been detected by an external detector (i.e., they have not been input to Algorithm 1), Algorithms 2, 3 and 4 are applied to estimate their boundaries[1]. Then, we determine the eyepoint positions and the face *roll* angle θ, derived proportionally and fixedly from the geometry of those *ROIs*. Figure 3.4 shows that the eye centre positions correspond to those of the *eyeROI* centres, that

[1]Note that algorithms 2, 3 and 4 are also used for estimating the ROI boundaries of the eyebrows, nose and mouth. Algorithm 2 invokes both algorithms 3 and 4.

Algorithm 1: Lightweight facial feature detection algorithm

Input: ROI of different facial regions (*faceROI*, *lEye(S)ROI*, *rEye(S)ROI*, *nose(S)ROI*, *mouth(S)ROI*), peak values (p_x, p_y), contour point threshold (*binThresh*)

Output: The detected landmarks (*eyePoints*, *eyebrowPoints*, *mouthPoints*, *nosePoints*, *contourPoints*)

---

for ***each*** *eye* **do**
  if ¬ eyeROI **then**
    Detect *eyeROI* in *eyeSROI* (*eyeSROI*, p_x, p_y) → *eyeROI*
  end
end
Estimate *roll* rotation angle derived from *eyeROIs* → θ
Estimate eye point positions in a fixed way derived from (*eyeROIs*, θ) → *eyePoints*
for ***each*** *eyebrow* **do**
  Get the eyebrow search region derived from (*faceROI* and *eyeROI*) and rotate it ($-\theta$) → *rotEyebrowSROI*
  ROIBoundDetection(*rotEyebrowSROI*, not_used, p_y) → *rotEyebrowROI*
  Estimate eyebrow point positions in a fixed way derived from *rotEyebrowROI*
  Apply θ rotation and transform to global image coordinates → *eyebrowPoints*
end
for *mouth and nose* **do**
  if ¬ partROI **then**
    Rotate *partSROI* ($-\theta$) → *rotPartSROI*
    ROIBoundDetection(*rotPartSROI*, p_x, p_y) → *rotPartROI* (Alg. 2)
  else
    Rotate *partROI* ($-\theta$) → *rotPartROI*
  end
  Estimate part point positions in a fixed way derived from *rotPartROI*
  Apply θ rotation and transform to global image coordinates → *partPoints*
end
ContourPointDetection(*faceROI*, *eyeCenters*, *lEyeLCorner*, *rEyeRCorner*, *mouthCorners*, *binThresh*) → *contourPoints* (Alg. 5)

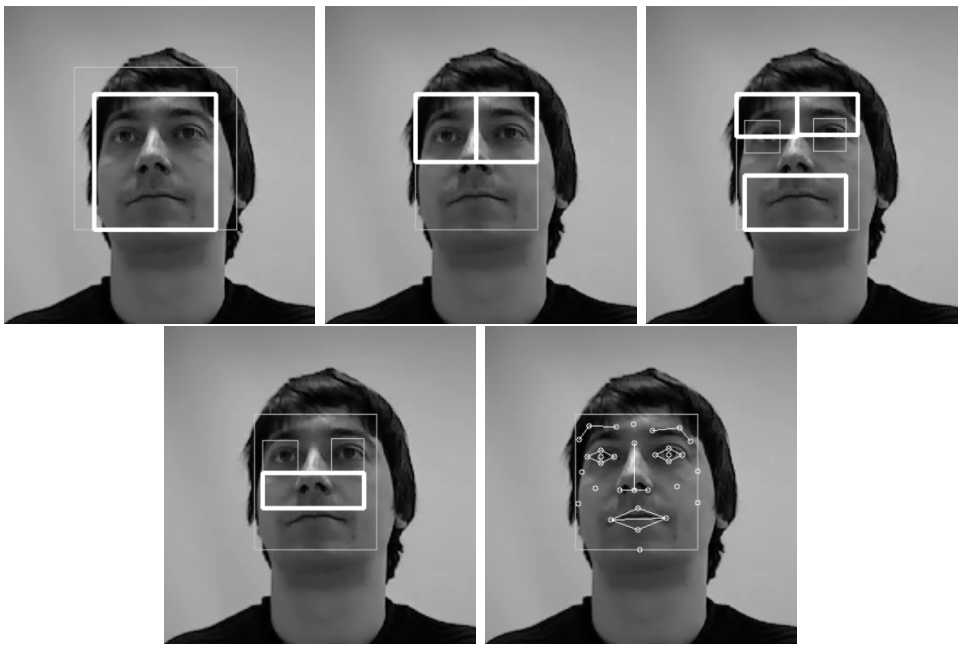


Figure 3.1: Proposed fitting approach. From left to right and top to bottom: (1) The detected face region and the *faceROI* derived from it (thicker line), (2) *faceROI* and the *eyeSROIs* derived from it (thicker line), (3) *faceROI*, the estimated *eyeROIs* and the *eyebrowSROIs* and *mouthSROI* derived from them (thicker lines), (4) *faceROI*, the estimated *eyeROIs* and the *noseSROI* derived from them (thicker line) and (5) the detected facial features.

eye widths and heights are equal in both sides with a proportion obtained from the mean *ROI* sizes, where θ is measured, and how the rest of eye points are located. As we rely on face detectors, the *roll* angle variation has a limited range, and therefore the eyes have well-defined search regions. Thanks to the eyes displaying approximate radial symmetry, we do not need the *roll* estimation for their localization.

For the estimation of the additional facial feature points in the eyebrows, mouth and nose, their corresponding ROI boundaries are used as a reference, also in a fixed way. These boundaries are also obtained through Algorithms 2, 3 and 4, considering the influence of the *roll* angle θ. In the specific case of eyebrows, as some people have bangs occluding them, or even no eyebrows, we do not calculate the boundaries in the X direction but fix them according to the search region width and the expected eyebrow geometry in

Figure 3.2: Facial feature detection procedure steps. From left to right and top down: (1) eye point detection, (2) eyebrow point detection, (3) mouth point detection, (4) nose point detection, (5) contour point detection and (6) OAM tracker model initialization example.

the 3D model. The parameters p_x and p_y are thresholds for the normalized gradient maps for detecting the horizontal and vertical boundaries. In our experiments we use $p_x = 20$ and $p_y = 50$ in all cases.

The double sigmoidal filtering applied to the search regions (Algorithm 2) allows us to reduce the influence of directional illumination, while the squared sigmoidal gradient calculation accentuates the likely edges, and neglects the edge direction information, and considers only the edge strength [114]. The estimation of the contour point positions is done in a fixed way too, taking into account the eye and mouth positions. Algorithm 5 returns eight contour points: the forehead centre, the left and right cheeks, the four facial corners and the chin bottom point. Even though none of these points is a fiducial point, they are useful for 3D model fitting and tracking. In the case of the facial side corner estimation, the image region that goes from the facial region boundary to its corresponding mouth corner is analyzed, assuming that a noticeable X gradient appears in that region in one of the sides but not in the other when the subject exhibits a non-frontal pose, which corresponds to the face side boundary (e.g., see Figure 3.2-(5)). For this, we calculate the squared sigmoidal gradient in X and assume that those side points lie on it. These side points subsequently allow us to better estimate the *pitch*

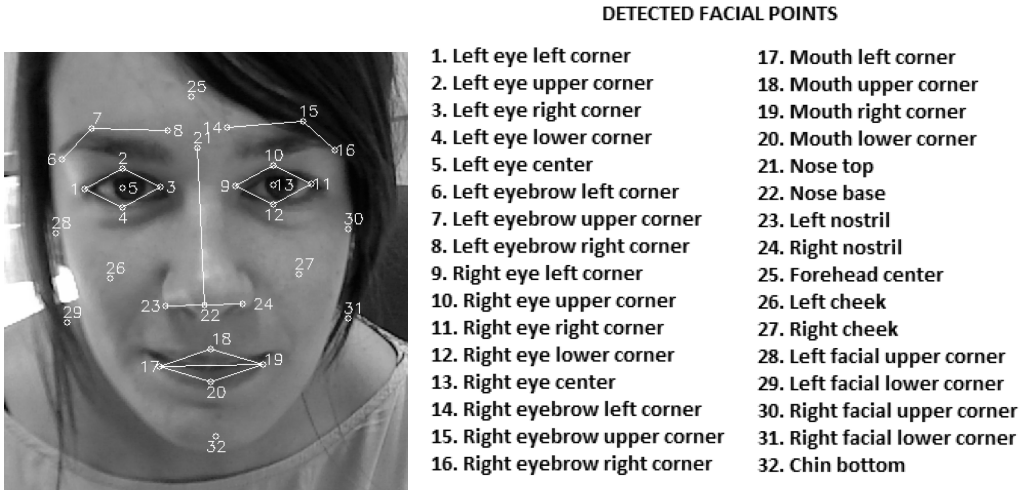


Figure 3.3: The detected 32 facial points. Note that the words *left* and *right* are relative to the observer rather than the subject.

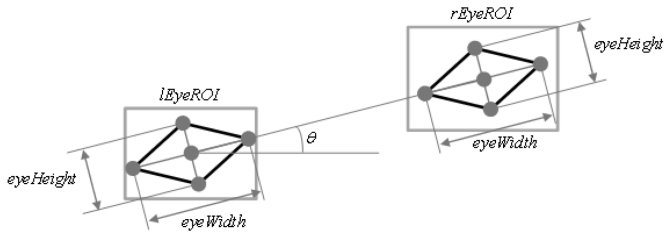


Figure 3.4: Eye points geometry derived in a fixed way from the estimated *eyeROIs*.

angle of the face. However, there might be cases in which both sides have a noticeable gradient in X, which may correspond not only to the face side boundary but also to other features such as beard, or local shadows. In order to filter these cases, we assume that the side that should take into account the gradient to estimate the X positions is that in which the mean positions are closer to the face region boundary. In contrast, for the other side, the X positions are those of the boundary itself (see Figure 3.2). The parameter $binThresh$ is the binarization threshold for the normalized gradient map in X. In our experiments we use $binThresh = 150$.

Algorithm 2: ROI boundary detection algorithm

Input: Face search ROI ($SROI$), peak values (p_x, p_y)
Output: Boundary values $leftX$, $rightX$, $bottomY$ and $topY$

Apply double sigmoidal filter to $SROI \rightarrow dsSROI$
Apply squared sigmoidal Y gradient to $dsSROI \rightarrow ssySROI$
YBoundDetection($ssySROI$, p_y) $\rightarrow$ ($bottomY$ and $topY$) (alg. 3)
XBoundDetection($ssySROI$, p_x, $bottomY$, $topY$) $\rightarrow$ ($leftX$ and $rightX$) (alg. 4)

Algorithm 3: ROI Y boundary detection algorithm

Input: Y sigmoidal gradient ROI($ssySROI$), Y peak value(p_y)
Output: Boundary values on Y axis $bottomY$ and $topY$

for ***each*** row *in* ssySROI **do**
  $(ssySROI_{height}/2 - |ssySROI_{height}/2 - y|) \cdot p_y \rightarrow w$
  $(w \cdot \sum_{x=1}^{width} ssySROI_x) \rightarrow wVertProj_{row}$
end
Normalize $wVertProj$ values from 0 to 100
Locate the local maximum above p_y with the lowest position in $wVertProj \rightarrow maxLowY$
$(maxLowY + ssySROI_{height}/4) \rightarrow topY$
$(maxLowY - ssySROI_{height}/4) \rightarrow bottomY$

Algorithm 4: ROI X boundary detection algorithm

Input: X sigmoidal gradient ROI($ssySROI$), X peak value(p_x)
Output: Boundary values on X axis $leftX$ and $rightX$

for ***each*** col *in* ssySROI **do**
  $(ssySROI_{width}/2 - |ssySROI_{width}/2 - x|) \cdot p_x \rightarrow w$
  $(w \cdot \sum_{y=bottomY}^{topY} ssySROI_y) \rightarrow wHorProj_{col}$
end
Normalize $wHorProj$ values from 0 to 100
Locate the first value above p_x starting from the left and right sides in $wHorProj$
 $\rightarrow$ ($leftX$ and $rightX$)

Algorithm 5: Contour feature detection algorithm

Input: Face data *faceROI*, *eyeCenters*, *lEyeLCorner*, *rEyeRCorner*, *mouthCorners* and *binThresh*

Output: Contour features (*foreheadCenter*,*lCheek*, *rCheek*, facialCorners and chinBottom)

$faceVector \leftarrow (lEyeCenter + rEyeCenter - mouthLCorner - mouthRCorner)/2$

$foreheadCenter \leftarrow (lEyeCenter + rEyeCenter + faceVector)/2$

$lCheek \leftarrow (lEyeLCorner + lEyeCenter - faceVector)/2$

$rCheek \leftarrow (rEyeRCorner + rEyeCenter - faceVector)/2$

ssxFaceROI ← Apply squared sigmoidal X gradient to *faceROI* and normalize between 0 and 255

for ***each*** *facial side* **do**

- *ssxFacialCornerROI* ← Get region between *mouthCorner* and *faceROI* outer boundary
- *binFacialCornerROI* ← Binarize *ssxFacialCornerROI* with *binThresh* and remove clusters (obtained through [115]) with $area < 0.8 \cdot ssxFacialCornerROI_{height}$
- $facialUCorner_y \leftarrow 0.75 \cdot ssxFacialCornerROI_{height}$
- $facialUCorner_x \leftarrow$ Get X centroid of white pixels at $facialUCorner_y$ in *binFacialCornerROI*
- $facialLCorner_y \leftarrow 0.25 \cdot ssxFacialCornerROI_{height}$
- $facialUCorner_x \leftarrow$ Get X centroid of white pixels at $facialUCorner_y$ in *binFacialCornerROI*
- *facialCorners* ← Transform to global image coordinates

end

facialCorners ← Check which side from *facialCorners* mean X position is further from its corresponding face region boundary, and then set their X positions in the boundary

chinBottom ← Calculate the intersection between the bottom of *faceROI* and the line traced by *faceVector*

3.2.2 Experimental results

To demonstrate the performance of the proposed method, we tested it on an iPad 2. For the test, we used images taken from the CMU Pose, Illumination and Expression (PIE) [116] database. Specifically, images with the face in front orientation and with a neutral gesture were used. In total, 7134 images were used. In this test, the landmark detection process needed an average time of 22 milliseconds to detect the entire set of landmarks, using the CPU (1 GHz dual-core 32-bit Cortex-A9) of the iPad 2. As a reference, Constrained Local Model (CLM) method needs an average time of 88 milliseconds to estimate the landmarks in each image during a tracking sequence, while the first image of the sequence needed nearly 250 milliseconds.

3.2.3 Conclusions

This section proposed a very efficient method to detect a set of facial feature points in monocular images. As it is not based on a pre-trained statistical model, it does not require a previous training phase. Thus, it is not biased by the variability of samples in the training dataset.

Unfortunately, this approach has some limitations. For example, it assumes that the user's face is near frontal, which limits the out-of-plane rotation of the face. Moreover, the mouth region landmarks only include the outer boundary of the mouth. Thus, it cannot estimate if the mouth is open.

Despite these limitations, the capabilities of the proposed method are sufficient for many applications in controlled or semi-controlled scenarios. Moreover, the computational efficiency of the feature extraction makes it suitable for very constrained hardware devices, such as an iPad 2.

3.3 Face landmark and gesture detection by DNN regression

Depending on the final application, a facial analysis method may need to identify different kinds of facial attributes. The list of possible attributes includes such diverse elements as the facial feature points, the orientation of the head, the eye-gaze and the local gestures, among others.

The extraction of each of the named attributes is a challenging task. However, there are synergies between some attributes that could be harnessed to

improve the performance of the final estimation. For instance, facial feature points and the estimation of the facial gesture are strictly related attributes, and it is possible to combine them taking advantage of this relationship to improve the accuracy of both estimations.

Based on the approach presented in [90], this section proposes an efficient method (named *M3*) to improve the landmark and gesture estimations, as well as other attributes like eye gaze and head orientation. It divides the process into two levels. The first level detects a small sub-set of feature points for the facial elements (i.e., face contour, eyes, nose and mouth) and estimates the head orientation. The feature points extracted in the first level define the facial element regions in the face image. The second level combines the head orientation estimation and the image patch of each facial region and estimates the specific landmarks and attributes for each region. The separation of the estimations reduces the complexity of the estimation and improves local expressiveness, maintaining efficiency.

3.3.1 Multi-level approach

The process has two levels of analysis (see Figure 3.5): a) holistic face attribute detection and b) facial zone-specific attribute estimation.

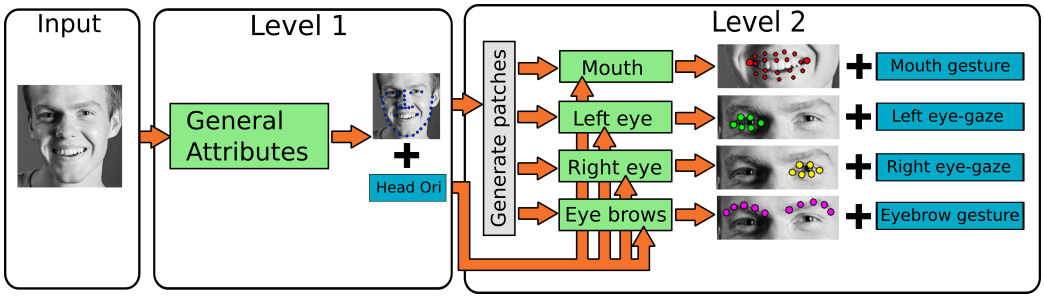


Figure 3.5: Flowchart of the proposed multi-level analysis pipeline.

The first level extracts general facial attributes. It takes the entire face in to consideration in order to estimate an initial set of landmarks and the head pose (i.e., the face yaw orientation).

The second level uses the output of the first level as input - it extracts the attributes of each facial zone (i.e., landmarks, gestures or gaze). First, it generates a set of normalised facial zone patches, one for each face region (i.e., mouth, eyebrows and each eye). Then, it estimates the zone-specific

landmarks and gestures using the zone patches and the orientation of the head.

The final output merges all the landmarks from both levels into a single set of 68 landmarks. Figure 3.6 shows all the landmarks detected on each level and the final set of landmarks for a face image.

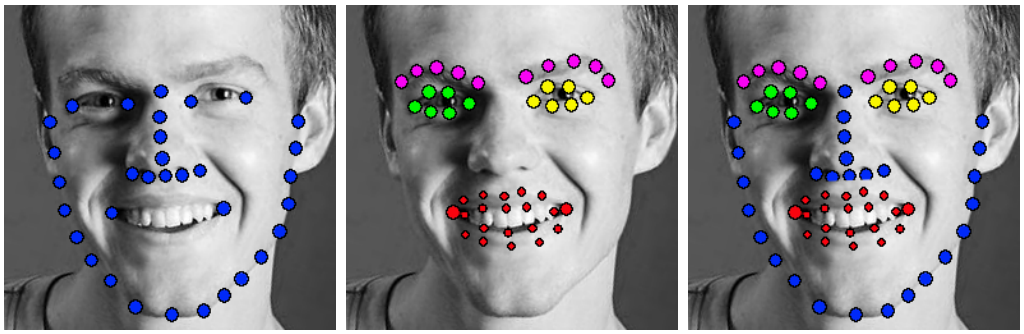


Figure 3.6: From left to right, a representation of the 32 landmarks detected in the first level, a representation of the landmarks detected in the second level and a combined representation of all the detected features. Each colour represents the landmarks for each face zone.

The zone patch generation process uses two *reference landmarks* from the landmark set extracted in the first step. Then, it applies a warping procedure to align the *reference landmarks* with two *target locations* in a previously defined zone patch template. The warping procedure normalises the orientation and size of the zone image region. Note that this process could add some unwanted distortion in patch regions occluded by a large out of plane head rotation.

The reference feature points used to generate the patches are those that maintain their relative location during gestures. Considering the high deformability of the mouth region, we chose feature points out of the gesture influence, like the bottom of the nose and the chin. We selected those landmarks because the influence of the evaluated mouth gestures in those locations is small enough to assume that they are stable or relatively static. Likewise, the reference points for each eye region are the eye corner landmarks, and the reference points for the eyebrow region are the left eye left corner and the right eye right corner landmarks, including both eyebrows in the same image patch. Figure 3.7 shows some examples of normalised face patches, including the location of the reference landmarks.

Figure 3.7: Examples of (from left to right) an eyebrow patch, a right eye patch and a mouth patch using the image shown in Figure 3.6. The red cross represents the *target location* of the *reference landmarks* for each patch. Note that the reference point of the mouth patch is out of the patch region.

To enhance the zone patches for the inference step, we apply a bilateral filtering process to reduce the possible noise in the warped image. At the same time, it preserves the information of edges and general shape. In the case of the eye regions, the warping process smooths the edge information because the size of this region tends to be small compared with the entire face region. An image intensity histogram equalisation compensates the sharpness reduction increasing the contrast and normalising the brightness for the eye patches.

Finally, each zone-specific neural network extracts the landmarks and attributes of each zone patch, also considering the estimated head orientation. The inclusion of the head orientation in the estimation process helps to ignore or reduce distortions caused by large head rotations during the warping procedure, making the estimation more reliable.

3.3.2 Facial attribute definition

The final estimation includes different facial attributes: a) the orientation of the head, b) the gesture of the mouth, c) the gesture of the eyebrows and d) the eye gaze of each eye.

The head pose (i.e., face yaw orientation) is defined as a classification between 5 possible classes *-60* (for angles <-60), *-30* (for angles from -60 to -30), *0* (for angles from -30 to 30), *30* (for angles from 30 to 60) and *60* (for angles >60).

The included zone-specific gestures are: smile, mouth frown and kiss gestures for the mouth region and inner eyebrow raise and inner eyebrow

frown for the eyebrow region. The gestures in each region are mutually exclusive, so only one can be active in a frame. Other gestures (e.g., mouth open) can be directly extracted from the landmark positions, measuring the distance between inner mouth landmarks, for example.

Gesture estimation is a classification process which estimates the probability distribution of the group of gestures for each face element. Moreover, we include an additional neutral class for each region to represent the absence of all gestures. The final probability estimation of the gestures in a face region is normalised using a softmax function.

Eye gaze estimation defines a gaze vector for each eye as a normalised vector of three values.

3.3.3 Network structure

Each network relies on a Multi-Task Learning (MTL) approach for the estimation process. It defines a CNN model to generate a list of outputs of different types and meanings using the same computation, similar to the method proposed by Zhang et al. [90]. Conventional MTL approaches try to optimise all tasks at once using a combined loss function. In our case, the loss function combines the loss of each task, assigning different priorities to each one. This priority designation defines the landmark detection as the primary task, setting the gesture estimation as an auxiliary task. Even if the landmarks have higher priority than the gestures, the gesture information improves the accuracy of the landmark detection due to the direct correlation between the position of the landmarks and the definition of each gesture.

The structure of each network has two parts, a) the trunk which performs the general feature extraction, and b) the leaves in charge of the final task estimations. The trunk of the network estimates all the features needed by the leaves to generate their estimations. The trunk shares its output with all the leaves, which use this information to estimate the landmarks and gestures. During training, the process updates the weights of the trunk and leaves combining all the outputs of the network. Hence, all the estimation tasks influence the training of the feature detection, enhancing their results through the synergies between them. See Figure 3.8 for a visual example of a multi-task network.

The trunk is a concatenation of four convolutional layers, including a pooling step after each convolution, and a final fully connected layer. Each convolution layer includes a *Rectified Linear Unit* (ReLU) activation func-

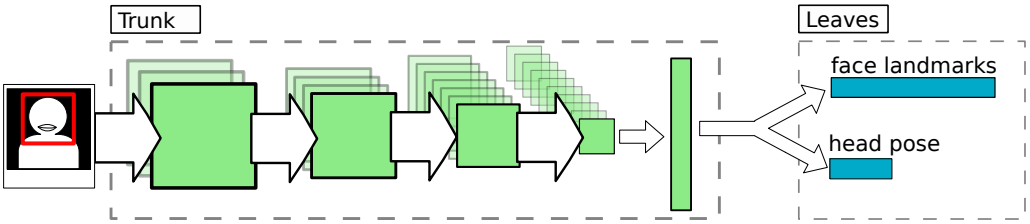


Figure 3.8: The general structure of the multi-task network designed for level 1. The layers in the trunk generate all the features for the leave layers, sharing the output values with both leaves. The leave layers are the output of the network.

tion. This network configuration is similar for all the cases, with small variation in layer sizes. Figure 3.9 shows the detailed configuration of each layer.

The input of the network in the first layer is only the cropped image of the face. However, the networks in the second level have two inputs: the normalised face zone patch and the orientation of the head estimated in layer one. To include the orientation of the head in the estimation process, the networks in the second level include an extra fully connected layer. This layer includes the head orientation values as concatenated elements (see Figure 3.9). Then, the final fully connected layer integrates all the elements into a single output vector.

Figure 3.9 includes the details and configuration of the proposed networks, and how the output of level 1 interacts with the estimation process of level 2.

3.3.4 Learning process

Due to the multilayer structure of the method, the learning process has two phases. In the first phase, we train the neural network defined for level 1. Then, in a second phase, we use the model trained for level 1 to generate the input data to train the models in level 2.

In all the cases, we define the landmarks $\mathbf{y}_i$ as a list of x and y coordinate values in the image $\mathbf{I}_i$, ordered as a unique list $\mathbf{y}_i = \{x_i^0, y_i^0, x_i^1, y_i^1, ..., x_i^n, y_i^n\}$. The error function for the landmarks is set as a least square problem,

$$L(\mathbf{y}, \mathbf{I}, \mathbf{W}) = \frac{1}{2} \sum_{i=1}^{N} \|\mathbf{y}_i - f(\mathbf{I}_i; \mathbf{W})\|^2 \tag{3.1}$$

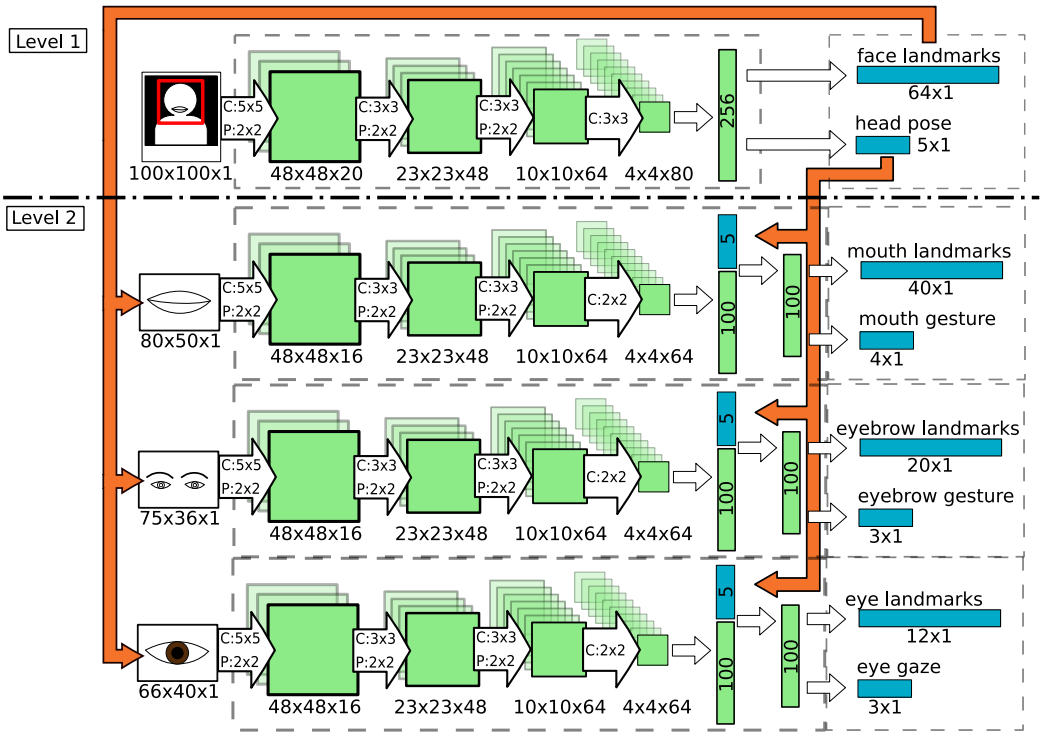


Figure 3.9: The neural network configuration for the attribute estimation process. The first row shows the network configuration for the first estimation level, including the input and the output elements. The rows below show the network configuration for each estimation model in the second level. A single eye model estimates the eye attributes for both eyes, using symmetry to estimate the attributes of the second eye. For each model, 'C' represents the kernel size of the convolutional layer and the 'P' represents the configuration of the pooling layer.

where f is the model function, N is the number of images in the batch and $\mathbf{W}$ is a list of weights that encapsulates the trained parameters of the neural network.

As mentioned previously, the model in level 1 classifies the head pose values (i.e., head yaw orientation) in five classes. Each head orientation class takes a probability value in the range $[0, 1]$. For the ground truth, a class has the value 1 if the head pose matches the class or 0 if it does not. As an orientation can only belong to one of the classes, they are mutually exclusive.

To compute the head's pose probability, the method adopts a softmax function P, which models the class posterior probability. P combines all orientation classes in a list of probabilities $\mathbf{h} = \{h_0, h_1, ..., h_4\}$. Equation (3.2) shows the cross-entropy loss function used for the head orientation error during the training process for a list of A classes. In this equation, $h_{i,a}$ represents the ground truth value of the orientation class a for the image i.

$$H(\mathbf{h}, \mathbf{I}, \mathbf{W}) = -\sum_{i=1}^{N}\sum_{a=1}^{A} h_{i,a} \log(P(h_{i,a}|\mathbf{I}_i; \mathbf{W})) \tag{3.2}$$

For the final error function, we combine equations (3.1) and (3.2) in a single expression to produce a combined output. The final loss function (3.3) includes a regularisation term to penalise high layer weight values, similar to the function proposed in [90]. To maintain the landmark estimation as the primary task, the final loss function includes a priority factor λ to manage the influence of the orientation class estimation error in the learning process. We define the λ value experimentally to correctly balance the landmark and gesture estimations.

$$\arg\min_{\mathbf{W}} \left\{ L(\mathbf{y}, \mathbf{I}, \mathbf{W}) + \lambda H(\mathbf{h}, \mathbf{I}, \mathbf{W}) + \|\mathbf{W}\|_2^2 \right\} \tag{3.3}$$

For models in level 2, the learning process follows a similar approach. However, the data used as ground truth for the training of the models in level 2 needs to be generated using the model trained for level 1. Thus, we processed the training dataset using the model trained for level 1, storing the output data for the posterior level 2 training phase.

For the mouth and eyebrow regions, each gesture g gets a value in the range $[0, 1]$, 1 being if the gesture is present and 0 if the gesture is not present. For the training process, it uses a loss function similar to (3.3) to determine the gesture probability model.

The softmax function P models the class posterior probability. To simplify the output and assuming that the gestures in the training dataset are mutually exclusive, it combines all gestures for a facial zone in a list of gesture probabilities $\mathbf{g} = \{g_0, g_1, ..., g_n\}$. For example, for the gestures in the mouth region (smile, mouth frown and kiss) the gesture probability distribution would be $\mathbf{g} = \{g_{smile}, g_{frown}, g_{kiss}, g_{neutral}\}$. The last value ($g_{neutral}$) defines the absence of any gesture, having the value of 1 when none of the defined gestures is present. Equation (3.4) shows the cross-entropy loss function used for the gesture error during the training process for a list of B gestures (including the neutral gesture). In this equation, $g_{i,b}$ represents the ground-truth value of the gesture b for the image i.

$$G(\mathbf{g}, \mathbf{I}, \mathbf{W}) = -\sum_{i=1}^{N}\sum_{b=1}^{B} g_{i,b} \log(P(g_{i,b}|\mathbf{I}_i; \mathbf{W})) \tag{3.4}$$

For the landmarks in the mouth and eyebrow regions, the learning process uses the same loss function presented in Equation (3.1). Both loss functions (L and G) are combined in a single expression (equation (3.5)) to produce a combined output. The equation also includes the regularisation term and the balance modifier λ.

$$\arg\min_{\mathbf{W}} \left\{ L(\mathbf{y}, \mathbf{I}, \mathbf{W}) + \lambda G(\mathbf{g}, \mathbf{I}, \mathbf{W}) + \|\mathbf{W}\|_2^2 \right\} \tag{3.5}$$

For the attributes in the eye region, the learning process replaces the classification loss function with a regression function similar to that presented in Equation (3.1). In Equation (3.6), g is the model function, the vector v_i represents the ground truth values for the eye gaze vector on image i, and $\mathbf{v}$ lists the eye gaze vector values for all the images.

$$V(\mathbf{v}, \mathbf{I}, \mathbf{W}) = \frac{1}{2}\sum_{i=1}^{N} \|v_i - g(I_i; \mathbf{W})\|^2 \tag{3.6}$$

Similar to previous examples, the final loss function for the eye region (Equation (3.7)) combines the error of the landmarks and the eye gaze estimation, with the additional λ factor to manage the influence of the secondary task.

$$\arg\min_{\mathbf{W}} \left\{ L(\mathbf{y}, \mathbf{I}, \mathbf{W}) + \lambda V(\mathbf{v}, \mathbf{I}, \mathbf{W}) + \|\mathbf{W}\|_2^2 \right\} \tag{3.7}$$

3.3.5 Experimental results

To evaluate the general performance of M3, this section compares the computation time of its implementation with two real-time methods proposed in the literature: Kazemi and Sullivan [18] and Baltrusaitis et al. [111]. Although the method by [111] is also capable of estimating certain gestures, the experimental comparison focuses only on the result of the main task, which is the extraction of facial feature points.

The authors in [18] propose a fast landmark detection method, able to estimate 68 face landmarks in a single millisecond. It relies on linear regressors, using binary pixel comparisons as a feature.

The work presented in [111] combines the AU detector presented in [100] with efficient methods to extract facial feature landmarks [117], head orientation estimation, and facial texture generation in a single analysis tool.

Model training

For the training phase, we used a subset of the images from the Multi-Task Facial Landmark (MTFL) dataset presented in [90]. However, we manually re-annotated the entire dataset with 68 facial landmarks, the gestures of the mouth (smile, mouth frown and kiss) and the gestures of the eyebrows (inner eyebrow raise and inner eyebrow frown).

First, we generated a mirrored copy of each image, doubling the number of samples. Then, we used an automatic landmark detection based on [18] to get a first estimation of the facial landmarks. Finally, we manually checked the results to discard the images with significant landmark estimation errors. The final dataset had 18720 images in total.

To increase the number of samples in the training dataset and increase the robustness of the estimation in different image capture conditions, we augmented the number of samples using different image perturbations (e.g., random noise, affine transformations and occlusions).

In addition, we balanced the dataset for the classification. Thus, the final training dataset for each classification process (i.e., the models for level 1, eyebrows and mouth) had the same number of samples for each of the classes to classify. For example, for the model in level 1, we generated a different number of augmentation images so that the number of final samples was the same for each orientation. We followed the same approach to create the training data for mouth and eyebrow models in level 2. Figure 3.10 shows

Figure 3.10: Some examples of training patches generated with the described perturbations. From top to bottom, the original face images, the patches for model in level 1, patches for eyebrow model and patches for mouth model.

some examples of different training patches.

For the eye regions, we used the samples created using a synthetic eye image generator based on the approach proposed in [118]. It creates a set of photo-realistic 3D eye renders, simulating eyes from different persons by variations in shape, skin colour and skin texture, among others. It also varies the eye-gaze and the orientation of the head with respect to the camera randomly. The left column of Figure 3.11 shows some examples of the output given by the previously mentioned method. As the image generation relies on a 3D eye model, the output information includes an accurate eye gaze vector and eye landmarks. With the set of eye images and landmarks, we generated the eye patches for training. We augmented the generated crops with a list of perturbations, including random noise, partial occlusions and transparent white boxes simulating reflections from glasses. Figure 3.11 shows some examples of the eye region patches generated for training.

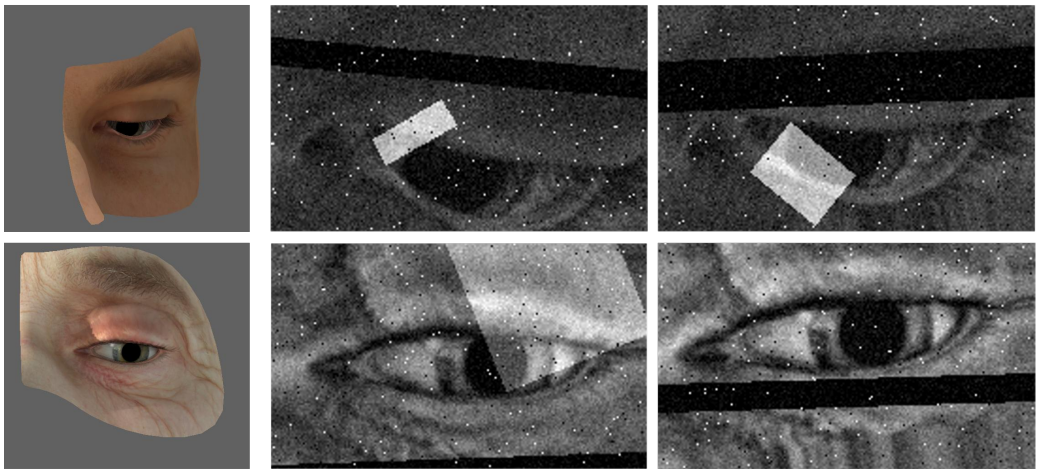


Figure 3.11: Two examples of the synthetic eye render (left column) and some of the eye region training patches generated with those images. The eye patches include the augmentation perturbations.

Performance comparison

For the comparison, we used a desktop PC to measure the processing time of each method. The PC included an Intel i5-9400F (@ 2.90GHz) as the main CPU, and 16 gigabytes of RAM, using an Ubuntu 18.04 as the operating system. No hardware acceleration was used in any of the cases.

The database used to make the comparison is the one presented by Sagonas et al. [47]. This database includes 600 images of different resolutions annotated manually with 68 facial feature points that include points for the eyes, eyebrows, nose, mouth and the contour of the face. Since the three compared methods needed a facial region to initialise the estimation process, the bounding boxes of the landmarks defined in the database were used as input for each evaluated method.

The point detection error is the mean square error (MSE) using the 68 points on the face and averaging across all images in the dataset. We normalised the estimation of each image using the interocular distance. This normalisation avoided biases caused by different image sizes in the dataset.

The measurement of computing time in each case included only the time necessary for the calculation of the feature points, excluding any image handling process (like loading or resizing processes) or the initialisation of the models.

Qualitative results

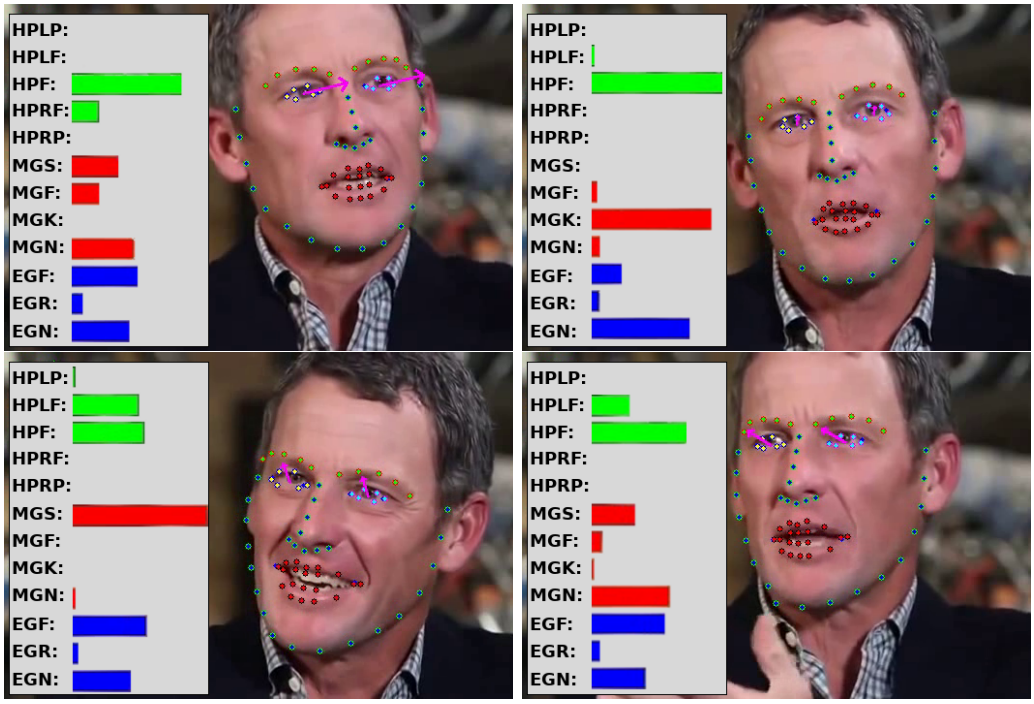


Figure 3.12: Each image shows the estimated gesture probabilities as bars (left), the tracked facial feature points (dots) and the estimated eye gaze vector (purple arrows). The gesture definitions are, from top to bottom, head pose left profile (HPLP), head pose left-front (HPLF), head pose front (HPF), head pose right-front (HPRF), head pose right profile (HPRP), mouth gesture smile (MGS), mouth gesture frown (MGF), mouth gesture kiss (MGK), mouth gesture neutral (MGN), eyebrow gesture frown (EGF), eyebrow gesture raise (EGR) and eyebrow gesture neutral (EGN).

During tracking, the method extracted the landmark positions and gesture probabilities from the users face. Figure 3.12 shows some examples of attributes extracted from a monocular video sequence.

Moreover, in the experiments using embedded devices, the implementation reached real-time performance (> 25 fps) using the embedded camera as image source. The hardware used for the experiments was the Iphone SE, which has an A9 (ARMv8-A dual-core @ 1.85 GHz) central processor.

Quantitative results

Taking all this into account, Table 3.1 presents the data extracted from the comparison. The table includes the landmark estimation error and the average time for their calculation.

Table 3.1: Results of the landmark estimation time measurement.

Method	**MSE**	**Time** (seconds)
Kazemi and Sullivan [18]	0.0872	0.0018
Baltrusaitis et al. [111]	0.0184	0.0389
M3 (Ours)	0.0227	0.0074

The method proposed in [111] is more accurate than the other methods, but the computation time is also significantly higher. In the case of [18], it estimates the feature points very efficiently, but at the cost of significantly higher error.

To see the relationship between the MSE and the computation time, Figure 3.13 shows the values in a visual representation. The image shows how the proposed method improves the CPU time and error compared to the other two.

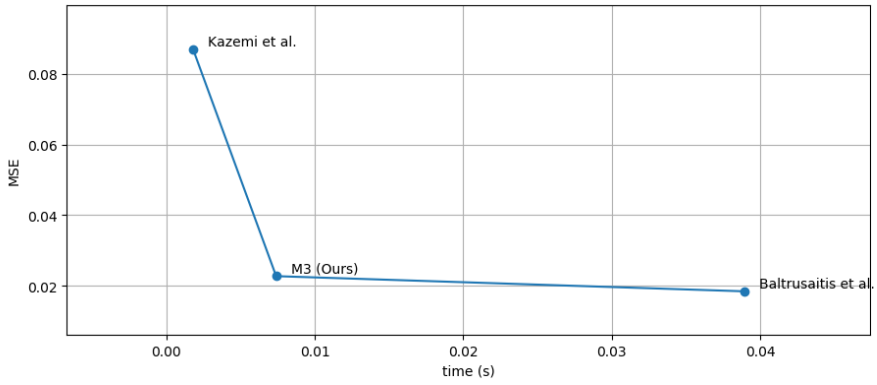


Figure 3.13: Visual representation of the MSE and time values for Kazemi and Sullivan [18], Baltrusaitis et al. [111] and the proposed method (M3).

M3 presents a better balance between computational load and precision, extracting data from the face gesture in the process.

3.3.6 Conclusions

This section presented a computationally efficient method to estimate the facial feature points as well as several additional attributes (i.e., head orientation, facial gestures and eye gaze orientation) in a single computation. Experiments show that it improves the balance of precision vs computation time compared to other methods proposed in state-of-the-art methods.

Given the computation time required for the estimation of landmarks, the method is a clear candidate to be used in systems with computational limitations, allowing for additional computations such as 3D estimation of facial attributes.

Chapter 4

Face fitting with non-rigid deformations

In the previous chapter, we analysed various techniques for detecting the facial feature points in an image. However, to move from an image's two-dimensional coordinates to a scene's three-dimensional coordinates, we need to add depth to those landmarks and determine the spatial relationship between the object (in this case, the face) and the camera.

In this chapter, we are going to analyse how we can make this transition using a deformable 3D model as a reference. This scheme follows the steps presented in the introductory chapter, in which we presented a tracking system based on three stages. Figure 4.1 reviews this scheme, adding certain specific elements of the facial tracking environment.

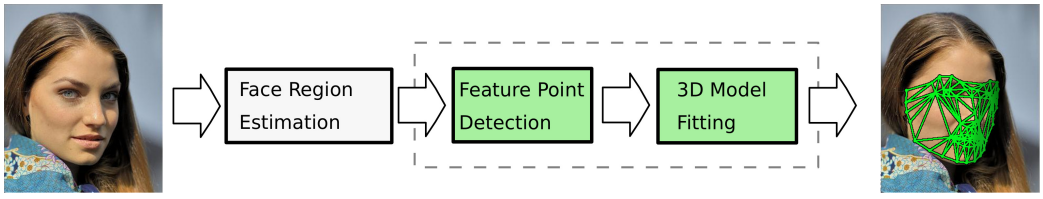


Figure 4.1: An adaptation of the graphical representation of the process to fit the 3D model presented in section 1.2.

During the research process for this doctoral thesis, we proposed different versions of this fitting approach. This chapter presents two of those versions. The first lays the foundations of the method (called Facial Feature

Back-Projection or FFBP). The second version (called Multi-Stage Back-Projection or MSBP) reformulates the problem to resolve deficiencies in FFBP, improving the quality of the final result.

This fitting approach configures a 3D deformable model using as input the feature points detected using an automatic feature detector (like one of the methods presented in chapter 3) and the camera parameters.

The proposed 3D deformable model alignment method is based on a two-step approach (see Figure 4.2). In the first step, a set of 2D fiducial face landmarks are located in the image, representing key points on the face of the user. In the second step, a deformable 3D model is deformed and adapted to those landmarks.

During the tracking, the proposed approach fits a deformable 3D model with the tracked face onto each of the images. This section introduces the fitting process for a single image of the sequence. Further sections will explain how to use this method to track the face in the entire video efficiently.

4.1 Facial Feature Back-Projection (FFBP)

The baseline version of the face model-fitting method estimates the rigid parameters (position and orientation) and gesture parameters in a single optimisation step. To simplify the projection computation, it assumes a weak perspective projection to estimate the error function.

4.1.1 3D deformable model definition

The 3D generic face model we adopt is a modified version of Candide-3 [119]. We will refer to this as Candide-3m. The modifications consist primarily in simplifying and streamlining the model in order to enhance the fitting and tracking capabilities of the original. Figure 4.3 shows the Candide-3m geometry compared to the original model. Figure A.1 shows the added and modified shape units (SUs) and animation units (AUs) for Candide-3.

The Candide-3m model has the following modifications for Candide-3:

- The geometry around the eyes has been simplified by removing the vertices that form the eyelids.
- The triangular mesh around the eyes and mouth has been tweaked, to make the mesh density more uniform in those areas and to fit the new

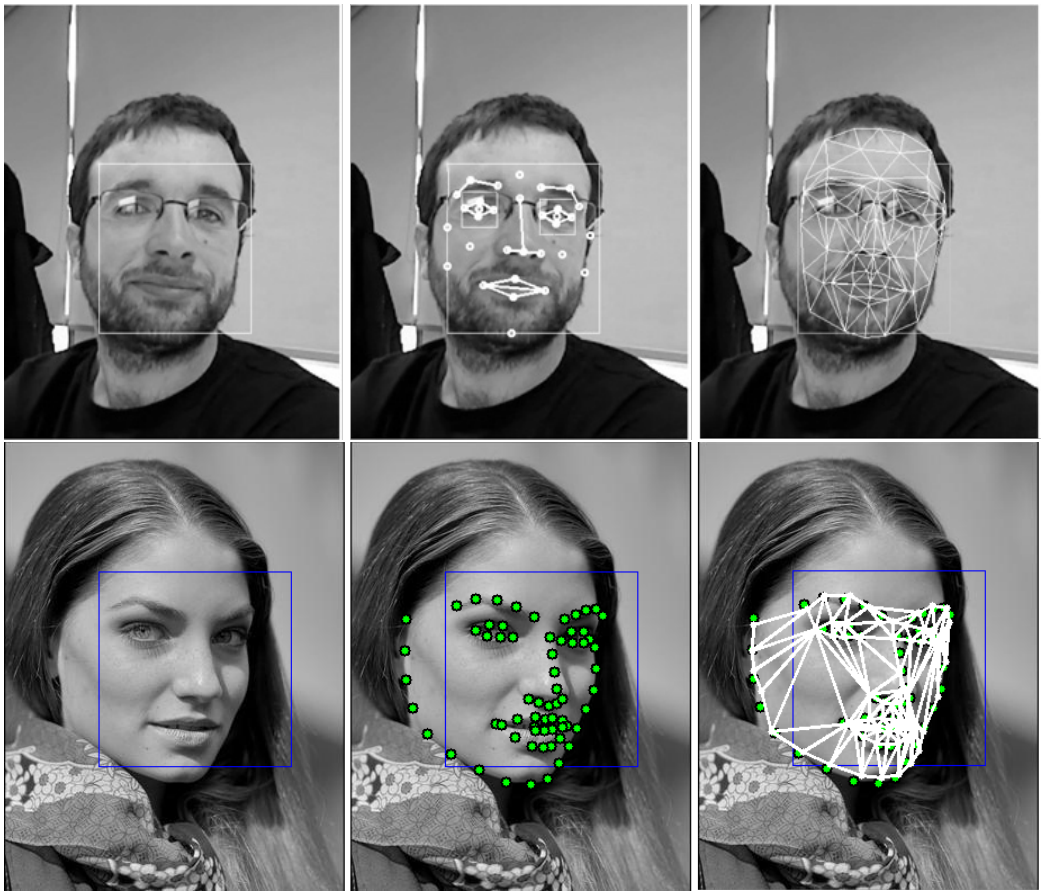


Figure 4.2: Face model fitting steps for FFBP (top row) and MSBP methods (bottom row). The left image shows the automatically detected face region. The central image shows the same face with the 2D landmarks overlapped. The right image shows the result of the 3D model deformation and alignment.

vertex list.

- The SUs have been changed in order to make them more appropriate for the initialisation procedure proposed in this study: (1) *Cheeks Z*, *Chin Width* and *Eyes Vertical Difference* SUs have been removed, maintaining the rest, and (2) three more have been added, called *Eyebrow Width*, *Eyebrow Separation* and *Nose Width*.

- The AUs have also been changed in order to allow for more expres-

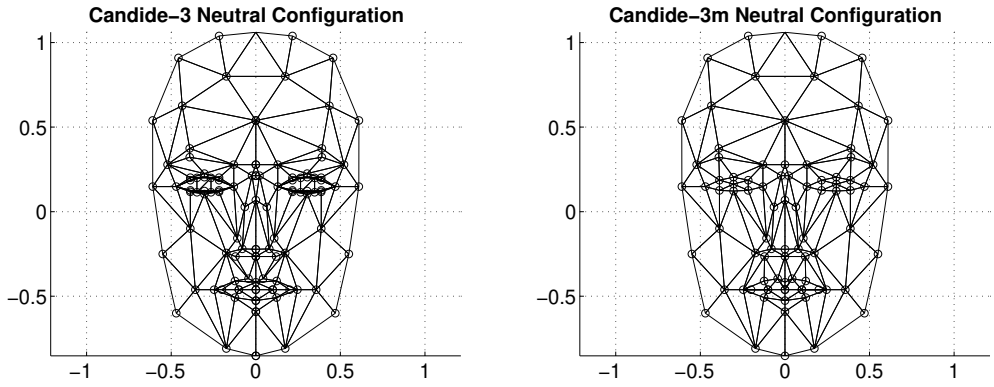


Figure 4.3: The geometries of the Candide-3 and the Candide-3m face models.

siveness: (1) All MPEG-4 FAPs have been removed, (2) the *Upper Lip Raiser*, *Lid Tightener*, *Nose Wrinkler*, *Lip Presser* and *Upper Lid Raiser* animation unit vectors (AUVs) have been removed, and (3) the *Outer Brow Raiser* AUV has been split into left and right AUVs, and (4) the *Eyes Closed* AUV has been split into left and right AUVs and reorganised according to the new vertex list.

4.1.2 Baseline fitting method

Once the facial features have been located in the image, the next stage is to determine which position, orientation, shape units (SUs) and animation units (AUs) provide for the best fit possible.

In the original experimental work [26], we used the 32 facial features detected using the landmark detection method explained in section 3.2, and we related them to a subset of vertices in the Candide-3m face model. However, any set of facial landmarks could be used to fit the model. In the forthcoming experimental section, for instance, we use a set of 68 landmarks to fit the Candide-3m model.

In any case, we use the existing correspondence between the 3D model points and the 2D facial features to make the face model fitting more efficient. The 3D face model is given by the 3D coordinates of its vertices $\mathbf{P}_i$, $i = 1, ..., n$, where n is the number of vertices. Thus, the shape, up to a global scale, can be fully described by a $3n$-vector $\mathbf{g}$, the concatenation of the 3D

coordinates of all vertices (equation (4.1)), where $\overline{\mathbf{g}}$ is the standard shape of the model, the columns of $\mathbf{S}$ and $\mathbf{A}$ are the shape and animation units, and $\boldsymbol{\tau}_{su} \in \mathbb{R}^m$ and $\boldsymbol{\tau}_{au} \in \mathbb{R}^k$, are the shape and animation control vectors, respectively.

The configuration of the 3D generic model is given by the 3D face pose parameters (rotations and translations in the three axes) and the shape and animation control vectors, $\boldsymbol{\tau}_{su}$ and $\boldsymbol{\tau}_{au}$. These define the parameter vector $\mathbf{b}$ (equation (4.2)).

$$\mathbf{g} = \overline{\mathbf{g}} + \mathbf{S}\boldsymbol{\tau}_{su} + \mathbf{A}\boldsymbol{\tau}_{au} \tag{4.1}$$

$$\boldsymbol{b}_t = [t_x, t_y, t_z, \theta_x, \theta_y, \theta_z, \boldsymbol{\tau}_{su}, \boldsymbol{\tau}_{au}] \tag{4.2}$$

A shape unit provides a way to deform the 3D model in order to adapt inter-person parameters such as the eye width and the eye separation distance. (see 4.1.1). Thus, the term $\mathbf{S}\boldsymbol{\tau}_{su}$ accounts for the shape or inter-person variability, while the term $\mathbf{A}\boldsymbol{\tau}_{au}$ accounts for the facial or intra-person animation. Hence, in theory, for face tracking, the shape units would remain constant, while the animation units could vary. However, it is challenging to perfectly separate both kinds of variability defining the generic face model such as they would fit any kind of human face. This challenge is due to the neutral facial expression, which varies significantly from person to person. Therefore, in our initialisation process, we must consider both the shape and animation units, without an explicit distinction between them. Only after the initialisation can we assume that the shape units remain constant during the tracking stage. Furthermore, we consider a subset of the animation units in order to reduce the computational load [78].

In equation (4.1), the 3D shape is expressed in a local coordinate system. However, one should relate the 3D coordinates to the 2D image coordinate system. To this end, we adopt the *weak perspective* projection model. We neglect the perspective effects since the depth variation of the face can be considered small, when compared to its absolute depth from the camera. Therefore, the mapping between the 3D face model and the image is given by a 2×4 matrix $\mathbf{M}$, encapsulating both the 3D face pose and the camera parameters. Thus, a 3D vertex $\mathbf{P}_i = [X_i, Y_i, Z_i]^T \subset \mathbf{g}$ will be projected onto the image point $\mathbf{p}_i = [u_i, v_i]^T \subset \mathbf{I}$ (where $\mathbf{I}$ refers to the image), as defined in equation (4.3).

$$\mathbf{p}_i = [u_i, v_i]^T = \mathbf{M}[X_i, Y_i, Z_i, 1]^T \tag{4.3}$$

The projection matrix $\mathbf{M}$ is given by equation (4.4), where α_u and α_v are the camera focal length expressed in vertical and horizontal pixels, respectively. (u_c, v_c) denote the 2D coordinates of the principle point, $\mathbf{r_1^T}$ and $\mathbf{r_2^T}$ are the first two rows of the 3D rotation matrix, and s is a global scale (the Candide model is given up to a scale).

$$\mathbf{M} = \begin{bmatrix} \frac{\alpha_u}{t_z}\, s\, \mathbf{r_1^T} & \alpha_u \frac{t_x}{t_z} + u_c \\ \frac{\alpha_v}{t_z}\, s\, \mathbf{r_2^T} & \alpha_v \frac{t_y}{t_z} + v_c \end{bmatrix} \tag{4.4}$$

The central idea behind this approach for deformable 3D model fitting is to estimate the 3D model configuration by minimizing the distances between the detected facial points and their counterparts in the projected model, with:

$$\mathbf{d}_j = [x_j, y_j]^T \subset \mathbf{I} \tag{4.5}$$

where q is the number of detected facial points, $j = 1, ..., q$ and $q \leq n$. Algorithm 6 shows the proposed method (FFBP), as we are inferring the configuration of a 3D deformable model from sparse data corresponding to one of its 2D projections. The more data we detect on the image, the more shape and animation units we will be able to vary in the model. The minimum condition to be ensured is that the points to be matched should not lie on the same plane. Thus, our objective is to minimize equation (4.6), where $\mathbf{p}_j$ is the 2D projection of the 3D point $\mathbf{P}_j$. Its 2D coordinates depend on the model parameters (encapsulated in $\mathbf{b}$). These coordinates are obtained via equations (4.1) and (4.3). The weight elements w_j refer to confidence values ($0 \leq w_j \leq 1$) for their corresponding $\mathbf{d}_j$. This confidence depends on the method used for the detection of points.

For this approach, we recommend setting the higher weights (e.g., 1) to eye points, mouth points, nose top and base points, and the forehead centre point; in a second level (e.g., 0.8) the eyebrow points and the rest of the contour points; and finally in a third level (e.g., 0.2) the left and right nostrils. We apply the POS algorithm [1], in order to get an initial guess of the

[1]POS is a pose solver based on a linearization of the perspective projection equations, which corresponds to a single iteration of POSIT [120].

position and orientation of the face object, before the optimisation procedure starts.

$$\mathbf{b}^* = arg \min_{\mathbf{b}} \sum_{j=1}^{q} w_j \cdot [(\{\mathbf{d}_j\}_x - \{\mathbf{p}_j(\mathbf{b})\}_x)^2 + (\{\mathbf{d}_j\}_y - \{\mathbf{p}_j(\mathbf{b})\}_y)^2] \quad (4.6)$$

The degrees of freedom from the Candide model (to be optimised) are set in a normalised framework so that their variations are not biased towards any in particular. Empirically, we found that it was better to keep the translation estimated by POS algorithm constant since the sensitivity of LM is very high to these global parameters. For this reason, we keep constant the position obtained through POS and optimise the rest of parameters, which can better accomplish this requirement.

Algorithm 6: Facial feature back-projection (FFBP) algorithm

Input: Standard shape of the 3D model $\overline{\mathbf{g}}$, confidence values $\mathbf{w}$, shape units$\mathbf{S}$, animation units$\mathbf{A}$ and detected landmark list $\mathbf{d}$

Output: The final configuration $\boldsymbol{b}$ of the aligned 3D model

Apply POS algorithm [120] to $\overline{\mathbf{g}}$ with $\mathbf{d} \rightarrow (\theta_x^0$ and θ_y^0 and θ_z^0 and t_x^0 and t_y^0 and $t_z^0)$

Minimize equation (4.6) through the LM algorithm [121], taking into account equations (4.1) and (4.3) for the update in the iterative optimization process. The position is kept constant $(t_x = t_x^0, t_y = t_y^0, t_z = t_z^0) \rightarrow \boldsymbol{b}$

4.2 Multi-Stage Back-Projection (MSBP)

After settling the foundations of the method, we identified two fundamental improvements. The first improvement relies on changing the assumption of a weak perspective projection model to a more realistic projection (i.e., full perspective projection). This change should improve the estimation of 3D positioning in the 3D scene.

The second improvement focuses on separating the optimisation process into different cost functions, estimating the entire configuration of the model into separated steps. Consequently, we first estimate the model parameters that have the most significant influence on the cost function (e.g., position and orientation), the remaining parameters with less influence (e.g., SU and

AU control parameters) are computed in a second optimisation. This change should improve the accuracy of SU and AU parameters.

Additionally, we revisited the deformable 3D model. The model used for the MSBP method focuses on reducing the distances in the image plane between the detected 2D landmarks and the perspective projection of the 3D model.

Further sections outline in detail the implementation of the improved MSBP fitting method.

4.2.1 Deformable 3D face model

The proposed deformable model is a parametric face mask with a defined set of 3D vertices and triangular surfaces (see Figure 4.4). It has a set of previously defined deformation parameters divided into two groups: 11 Shape Units (SUs) and 6 Animation Units (AUs). Table 4.1 shows the list of SU and AU names.

Table 4.1: List of shape and action units included in the deformable 3D model.

Shape Units	**Action Units**
Eyes Height	Open Mouth
Eyes Separation Distance	Raise Left Outer Eyebrow
Eyes Vertical Position	Raise Nose Bridge
Eyes Width	Raise Right Outer Eyebrow
Face Length	Smile
Face Roundness	Stretch Lips
Lip Thickness	
Mouth Vertical Position	
Mouth Width	
Nose Width	
Nose Z Extension	

While previous studies describe similar 3D models [119], we have generated an entirely new 3D model for this study. The motivation is to provide a deformable 3D model whose vertices match easily with the 2D landmarks provided by the landmark detector.

Indeed, the 3D model must be related to the result of the landmark detector to align the 3D model. For this reason, we define a strict relationship between the 3D vertices and the 2D landmarks. We established this relationship previously, considering the projection of the model vertices (without deformation) in the image plane. After manually aligning the projection of the vertices with the 2D landmarks, we associate each landmark with the closest vertex projection. These relationships are stored as landmark-vertex couples to be used later in the optimisation step.

The deformation parameters are used to adapt the form of the mesh to the different possible face shape and gestures. The SU parameters modify the shape of the mesh to adjust it to the specific face shape of the user (e.g., *vertical mouth position* or *face width*) and are assumed constant for the face of a particular individual. The AU parameters modify the mesh to adapt the model to the specific gesture of the tracked face (e.g., smiling, blinking or mouth opening) and they can change during the tracking in a video stream. Each AU parameter controls a particular deformation (e.g., *left eyebrow-raiser* or *mouth stretcher*) as can be seen in 4.5.

Hence, the entire configuration of the 3D model, $\boldsymbol{b}_t$, for a certain 3D face pose is composed by: 1) the rigid configuration parameters (position t_x, t_y, t_z and orientation $\theta_x, \theta_y, \theta_z$) of the model with respect to the camera, 2) the list of SU parameters $\boldsymbol{\tau}_{su}$, and 3) the list of AU parameters $\boldsymbol{\tau}_{au}$. Equation (4.7) shows a vector with the entire set of the model parameters.

$$\boldsymbol{b}_t = [t_x, t_y, t_z, \theta_x, \theta_y, \theta_z, \boldsymbol{\tau}_{su}, \boldsymbol{\tau}_{au}] \tag{4.7}$$

4.2.2 3D face model alignment

The 3D mesh is aligned with the detected 2D landmarks through three separate non-linear optimization steps:

1. Estimate the rigid parameters $(t_x, t_y, t_z, \theta_x, \theta_y, \theta_z)$ of the 3D model.
2. Estimate all the shape deformation parameters $\boldsymbol{\tau}_{su}$ in a single optimization.
3. Estimate each animation deformation parameter $\boldsymbol{\tau}_{au}$ sequentially.

Splitting the unknowns into three parts leads to better efficiency and stability. The first step infers the global pose even when the AU and SU do

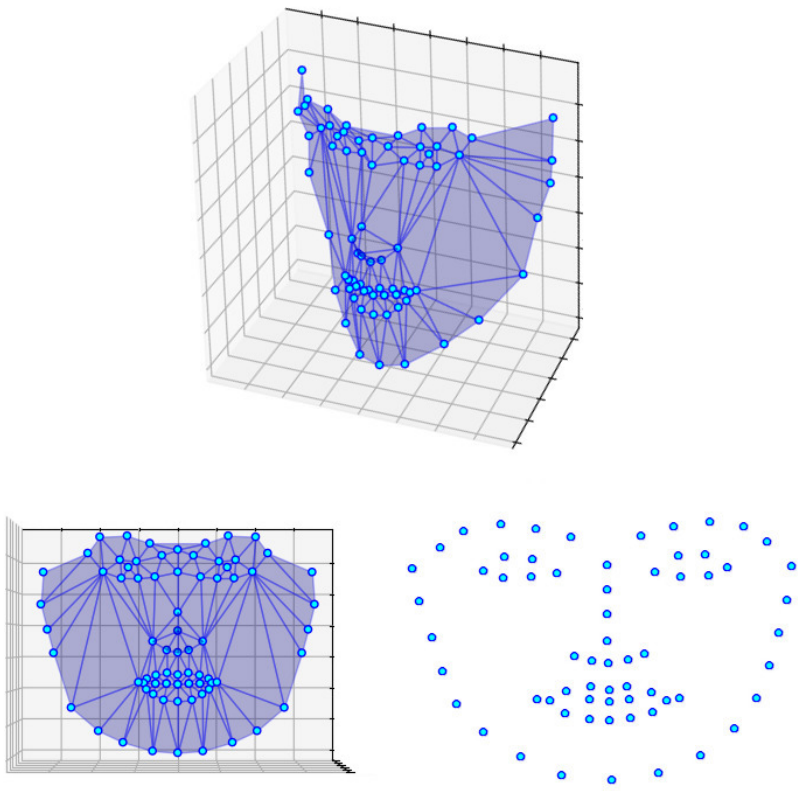


Figure 4.4: The upper image in the figure shows the shape of the deformable 3D model used in this study. The lower row shows the frontal view of the same deformable 3D model (left) and the mean landmark shape of the used 2D landmark detector model (right). In the current representation, all the 3D deformation parameters (SUs and AUs) are set to zero value.

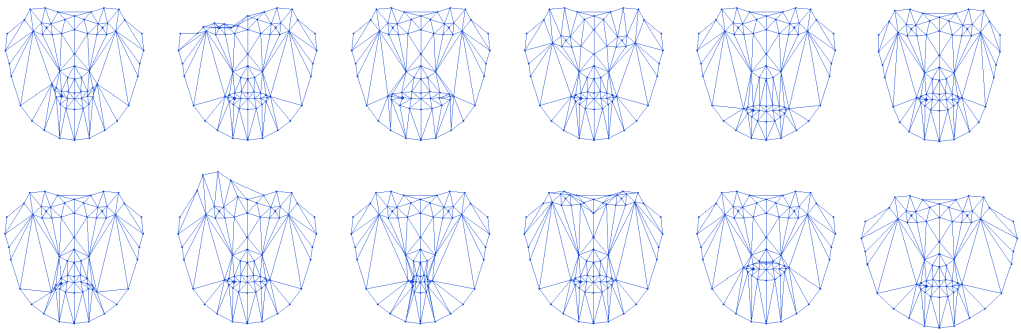


Figure 4.5: Example of different deformations of the model. Each column shows the same deformation with positive deformation values (top row) and negative deformation values (bottom row). The shown deformations correspond to (from left to right) lip corner depressor (AU), left eyebrow raiser (AU), mouth stretcher (AU), eye vertical position (SU), mouth vertical position (SU) and face width (SU).

not correspond to the real ones[2], giving a better initial position to estimate the shape and pose. The second step relocates the face structure adapting the entire facial shape, and the third step refines the local gestures relying on the estimations of previous operations.

Figure 4.6 shows an example of the alignment. Each optimization process tries to find the parameters $\boldsymbol{b}$ which minimize the sum of Euclidean distances (L_2) between the landmark positions $\boldsymbol{y}$ and the projection of the related model vertices $P(\boldsymbol{b}, c_x, c_y, f_x, f_y)$.

This projection function adopts the pinhole (perspective) camera model [5] (ignoring the lens distortion). c_x, c_y, f_x and f_y denote the intrinsic camera parameters (sensor centre in x, sensor centre in y, focal length in x and focal length in y respectively). If the camera parameters are unknown for a certain device, the system assumes a set of previously defined values. This simple camera model provides good realism for full perspective projections.

Equation (4.8) describes the generic loss function:

$$e = \underset{\boldsymbol{b}}{\operatorname{argmin}} \sum_{i=1}^{n} \|\boldsymbol{y}_i - P(\boldsymbol{b}, c_x, c_y, f_x, f_y)\|^2 \tag{4.8}$$

[2]This is motivated by the fact that the 3D pose parameters benefit from the global rigidity of the head as seen in the set of the 2D points.

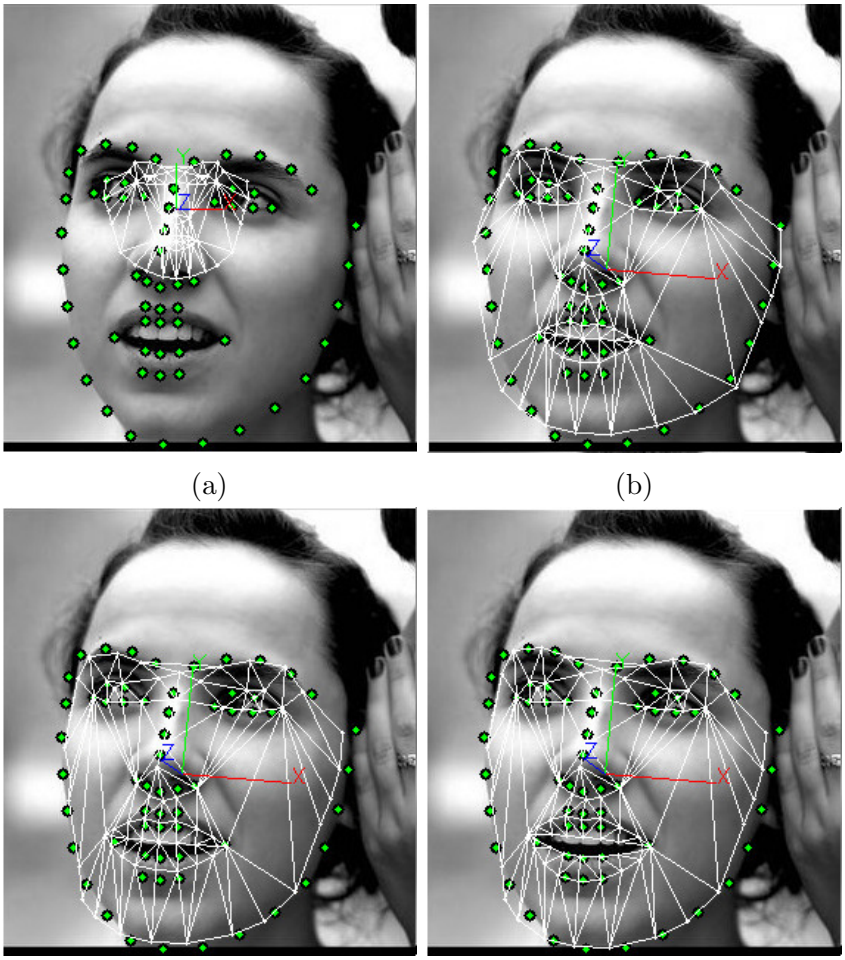


Figure 4.6: Example of the optimisation steps for the 3D model alignment: a) the initialisation of the fitting in the centre of the space, b) the raw estimation of the position and rotation of the 3D object, c) the shape parameter optimisation, and d) the animation parameter optimisation.

where n is the number of 2D landmarks and e the fitting error value. The loss function is the same in all the optimisation phases, but at each stage, the error value is computed using a different list of landmark-vertex couples. Thus, the first step uses all the landmark-vertex couples; the second step only uses the landmark-vertex couples related to the shape unit parameters

of the 3D model, and the third step uses the landmark-vertex couples related to the animation unit analysed in each case.

Before the fitting process, the 3D model is initialized with a neutral configuration:

$$\hat{\boldsymbol{b}} = [0, 0, Z_0, 0, 0, 0, \mathbf{0}, \mathbf{0}] \tag{4.9}$$

$$Z_0 = \frac{1}{2}(Z_{near} + Z_{far}) \tag{4.10}$$

Thus, when the location of the model is unknown, it is initialized in the centre of the tracking 3D space, the corresponding rotation values are set to zero (in a vertical position and faced to the camera) and all the deformation parameters are set to zero as shown in the first image of Figure 4.6[3]. The position of the 3D model is on the optical axis of the camera (the vertical and horizontal location values since t_y and t_x are set to zero) and the depth value t_z is set as Z_0 the centre point of the space between previously defined Z_{near} and Z_{far} 3D planes, as shown in (4.10).

If we know the configuration of the model in the previous frame ($\boldsymbol{b}_{t-1}$), the same position and orientation parameters are used as the initial guess. This initialisation reduces the iterations needed to estimate the current pose.

After initializing the model, the first optimization step estimates the pose parameters $(t_x, t_y, t_z, \theta_x, \theta_y, \theta_z)$ with a Levenber-Marquardt (LM) non-linear optimization method. The error computation includes all the landmark-vertex couples.

Then, the second optimisation step computes all the shape unit parameters ($\boldsymbol{\tau}_{su}$) to adapt the form of the 3D model to the face shape of the user. This step does not change the pose parameters (position and orientation) computed in the first step, keeping them constant. We use a Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm for non-linear optimisation. This computation only uses the landmark-vertex couples involved in the shape deformation to reduce the calculation.

The third optimisation step estimates each animation unit parameter ($\boldsymbol{\tau}_{au}$) one by one to adapt the shape of the 3D model to the gesture of the face. As in the previous step, we are using a BFGS non-linear optimisation method. This step only uses the landmark-vertex couples related to the calculated AU deformation to reduce the computation.

[3]The projection of the 3D model is not centred in the image because the face region was cropped to improve the visibility of the face region in this text. However, the projection of the object corresponds to the centre of the original image.

Regarding the pose parameter computation (face model position and orientation), LM shows good results with low computation times. However, when we use LM for deformation parameter estimation (second and third optimisation steps), the results are not so good. To improve the optimisation results given by LM estimating the deformation parameters, we used the BFGS optimisation method instead, maintaining the LM method for the rigid parameters. BFGS is more robust than LM for general optimisation problems. For example, if the second derivative of the function to be optimised is negative definite, the iterations needed by the LM method for convergence increase dramatically. In contrast, BFGS needs more computation than LM, and thus, more processing time.

The heterogeneous nature of the parameters used for the model and the full perspective projection needs a normalisation procedure to ensure a nonbiased result towards one or some of the parameters. The objective of the normalization step is to maintain the values of each pose parameter $(t_x, t_y, t_z, \theta_x, \theta_y, \theta_z)$ between the values -1 and +1. We use (4.11) to transform the value space of the model parameters from the real value space to the normalised value space.

$$\begin{aligned} t_x^{'} &= t_x/Z_0 \\ t_y^{'} &= t_y/Z_0 \\ t_z^{'} &= 2 * (t_z - Z_{near})/(Z_{near} - Z_{far}) + 1 \\ \theta_x^{'} &= \theta_x/\pi \\ \theta_y^{'} &= \theta_y/\pi \\ \theta_z^{'} &= \theta_z/\pi \end{aligned} \tag{4.11}$$

We remember that the value Z_0 represents the centre point of the space between the 3D planes Z_{near} and Z_{far}, which are previously defined.

After the first optimization step has estimated the pose parameters of the model using the normalized values $(t_x^{'}, t_y^{'}, t_z^{'}, \theta_x^{'}, \theta_y^{'}, \theta_z^{'})$, they need to be transformed again to the original value space. This transformation is computed using (4.12) (inverse of (4.11)):

$$
\begin{aligned}
t_x &= t_x^{'} * Z_0 \\
t_y &= t_y^{'} * Z_0 \\
t_z &= \frac{1}{2} * (t_z^{'} - 1) * (Z_{near} - Z_{far}) + Z_{near} \\
\theta_x &= \theta_x^{'} * \pi \\
\theta_y &= \theta_y^{'} * \pi \\
\theta_z &= \theta_z^{'} * \pi
\end{aligned}
\tag{4.12}
$$

Algorithm 7 summarises the entire fitting process, including the model initialisation and normalisation calls.

Algorithm 7: Multi-Stage back-projection (MSBP).

Input: Set of 2D face landmarks $\boldsymbol{l}$, landmark and vertex relation list $\boldsymbol{R}$, previous frame configuration $\boldsymbol{b}_{t-1}$

Output: The configuration $\boldsymbol{b}_t$ of the aligned 3D model

if *Previous configuration* $\boldsymbol{b}_{t-1}$ *defined* **then**
  Initialize the model with previous frame configuration $\boldsymbol{b}_t = \boldsymbol{b}_{t-1}$
else
  Initialize the model with neutral configuration $\boldsymbol{b}_t = \hat{\boldsymbol{b}}$
end
Set deformation values $(\boldsymbol{\tau}_{su}, \boldsymbol{\tau}_{au})$ to zero in $\boldsymbol{b}_t$ Change from configuration values to normalized value ranges (equation (4.11)) $(\boldsymbol{b}_t) \rightarrow \boldsymbol{b}_t^{'}$
Optimize position and orientation using LM $(\boldsymbol{b}_t^{'}, \boldsymbol{l}, \boldsymbol{R}) \rightarrow (\theta_x^{'}, \theta_y^{'}, \theta_z^{'}, t_x^{'}, t_y^{'}, t_z^{'})$
Optimize shape deformation using BFGS $(\boldsymbol{b}_t^{'}, \boldsymbol{l}, \boldsymbol{R}) \rightarrow \boldsymbol{\tau}^{'}{}_{su}$
for $au_j \in \boldsymbol{\tau}^{'}{}_{au}$ **do**
  Optimize gesture deformation using BFGS $(\boldsymbol{b}_t^{'}, \boldsymbol{l}, \boldsymbol{R}) \rightarrow au_j^{'}$
end
Change from normalized values to configuration value ranges (equation (4.12)) $(\boldsymbol{b}_t^{'}) \rightarrow \boldsymbol{b}_t$

4.3 Face tracking in a Video Sequence

Video sequences have important differences when compared to single images. We can consider a video sequence as a series of concatenated single images. However, there is a direct relationship between the content of all the frames in the sequence. The tracker can take advantage of this association to improve performance.

This section has three parts. First, we expose how a baseline tracker works, considering each image as an independent frame. Then, we show how the relationship between the images can improve the performance of the tracking. Finally, we show an efficient filtering approach to reduce landmark detection jittering during the tracking.

4.3.1 Baseline tracker

The basic 3D tracking flow in a video sequence $\boldsymbol{I}$ processes each frame I_i in three steps: 1) the detection of the face region r, 2) the detection of the feature points in the image and 3) the 3D model fitting.

The first step estimates the face region in the image using an automatic face detection method [122–125]. Then, the second step detects the 2D feature points of the face in the detected region. Finally, the third step computes the configuration $\boldsymbol{b}$ for the 3D deformable model using the detected 2D landmarks $\boldsymbol{l}$ as explained in section 4.2.2.

If the detector does not detect any face in the image, the system continues to the next frame ignoring the point detection and the 3D fitting (steps 2 and 3). In the next frame, the face detector tries again to find a face in the image.

4.3.2 Fast tracker

The baseline tracker needs to call on the face detector on each frame. The face detector needs to analyse the entire image, a computationally expensive process. To make the tracking more efficient, we propose a method to reduce the computational needs of the facial region re-detection between consecutive images using the information of the previous frame.

In an image sequence captured in real-time ($\backsim$30fps), we can assume that the facial appearance of the user has no significant changes from one frame to the next. We can use the face appearance information in a frame to determine the new face location in the next frame, using a template-matching approach [126]. Moreover, we restrict the template search area to the region around the face in the previous frame. These two modifications significantly reduce the re-detection computation compared to the original automatic face detector.

However, this tracking approach still needs an automatic face detector to locate the face in the first image of the sequence. In addition, the appearance

of the face could change significantly in some cases (e.g., partial occlusions or re-detection errors) causing the tracking to fail. In those cases, the automatic face detector restarts the tracking, locating the face again in the next frame. Thus, we combine the automatic face detection and the template-matching based re-detection in the same pipeline, keeping the computational cost to a minimum.

To avoid tracking errors caused by global changes in the image (e.g., global illumination changes, automatic gain corrections), we normalise the cross-correlation value of the similarity measurement during the template-matching analysis. This normalisation reduces the influence of global changes in the template comparison.

Regular execution of the tracking algorithm works as follows.

First, an automatic face detector initialises the tracking sequence of a user. This first step finds a face in the entire image using a previously trained face model. Using the face region defined in subsection 4.3.1, the feature points are detected and fit to the 3D model following the steps in Algorithm 7.

The tracker projects a set of vertices of the fitted 3D model onto the image to determine the face image region for the template-matching.

In the next frame, the tracker finds the most similar image area using as template the region it learned from in the previous frame. Using the new region, steps 2 and 3 find the new feature points and fit them to the 3D model. Once again, it projects the 3D model vertices to determine the new face region for the next frame.

Figure 4.7 shows an example re-detection step using a video sequence.

Note that since the detected 2D landmarks rely on local facial features in the image, they compensate for small errors in the location of the facial region. Then, the 3D model fitting process uses the 2D landmarks to acquire the 3D model configuration, ignoring the detected area for that task. This way of generating the facial template reduces the error accumulation that is usually associated with an iterative template-matching process.

However, there are some cases where the template-matching based tracking could fail. For example, an error in the 2D landmark detection could make the tracker estimate an incorrect face template, degrading the tracking even when the template was correctly found. To detect those cases, the tracker checks the learned patch using an MMOD [122] face classifier to ensure that the section contains a valid face. The MMOD method was initially designed to detect faces in images by checking all the possible image regions,

Figure 4.7: Representation of the face location update between the frames #24 and #25 of the sequence *410* of the '300VW challenge' dataset [127]. The tracker extracts the face template (middle) from the frame #24 using the projection of the 3D vertices. Then, it finds the template in the red area in the frame #25.

changing the location and size of the tested area. In the case of the template validation process, MMOD only checks a single space: the entire template patch. This restriction reduces the computation significantly, achieving a lightweight validation approach.

If the result of the validation is negative, the tracker discards the fitting process, and it calls on the automatic face detector in the next frame, to restart the tracking sequence.

4.3.3 Fast tracker with points filtering

The 2D landmark extraction process can be noisy in some cases, affecting the robustness of the tracking process.

To reduce the impact of noisy detections, we include a filtering step based on an optical flow analysis [128] of the image. This analysis method reduces the jittering in the landmark detection without the delays added by other filtering methods like moving average filter or Kalman filter.

The optical flow analysis method takes two consecutive frames and a list of pixel locations (i.e., a list of 2D landmarks) in the first frame. Next, it analyses the pixel value changes between two consecutive images and around

the original locations, estimating the position of the points in the second image.

Different factors cause the optical flow analysis to deteriorate over time (e.g., error accumulation, wrong evaluations, regions with uniform colour). To limit the effect of the deterioration, we define a maximum distance (e.g., 5-pixel diameter) between the detected landmark location and the estimation given by optical flow analysis. If the range exceeds the specified threshold, the tracker uses the detected landmark for the 3D model fitting and the further optical flow estimations for that landmark. If the distance is less than the threshold, then it uses the optical flow prediction as the current landmark location.

This filtering method reduces the noise efficiently in the detection without perceptible delays in the tracking. Algorithm 8 shows the filtering process. Moreover, Algorithm 9 combines the fast-tracking approach and the filtering process in the same tracking pipeline.

Algorithm 8: Optical flow-based filtering method.

Input: Current face image I_t, Previous face image I_{t-1}, Previous landmark locations $\boldsymbol{l}_{t-1}$, Current landmark detections $\boldsymbol{l}^d$

Output: List of filtered landmark positions $\boldsymbol{l}$

```
threshold = 5 pixels
if First frame of tracking sequence then
    Use detected landmark l^d → l
else
    Estimate optical flow (I_t, I_{t-1}, l_{t-1}) → l^o for l_n^o ∈ l^o do
        Compute euclidean distance (l_n^d, l_n^o) → d
        if d > threshold then
            Use detected landmark l_n^d → l_n
        else
            Use filter estimation l_n^o → l_n
        end
    end
end
```

4.4 Experimental results

In this section, we present the experimental results and the used setup.

Algorithm 9: Final face tracking approach.

Input: Face image sequence $\boldsymbol{I}$, landmark and vertex relation list $\boldsymbol{R}$

Output: The configuration $\boldsymbol{b}$ of the fitted 3D model

```
redetect = True
for I_t ∈ I do
    if redetect then
        Use a face detector to estimate face region → r  redetect = False
    else
        Use template-matching to detect a face region (I_t, A_{t-1}) → r
        Validate face region (r)
        if Validation negative then
            redetect = True
        end
    end
    if Not redetect then
        Detect face landmarks (r, I_t) → l^d
        Filter detected landmarks (I_t, I_{t-1}, l_{t-1}, l^d) → l_t (Algorithm 8)
        Fit 3D model with 2D landmarks (l_t, R) → b_t (Algorithm 7)
        Store face region appearance (b_t, I_t) → A_t
    end
end
```

4.4.1 Experimental setup

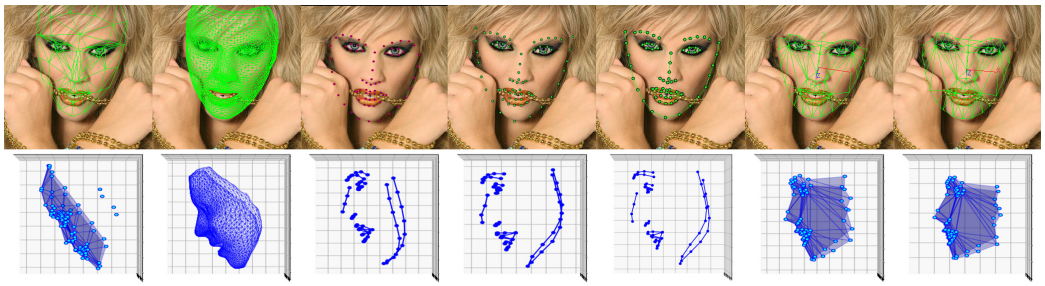


Figure 4.8: Fitting example for each fitting method using the same image. The top row shows the projection of each model on the image. The bottom row shows the side view of the same fitting. Each column represents a method. From left to right FFBP, Huber et al. [7], Baltrusaitis et al. [6], Bulat et al. [8], Feng et al. [85], Kazemi and Sullivan [18] + MSBP and Baltrusaitis et al. [6] + MSBP.

First, we compared the accuracy of the different selected methods using

two measurements: 1) the 3D orientation difference between the ground truth and the estimated values, and 2) the distance between the ground truth 3D vertices and the predicted vertices (expressed in the same coordinate system). Then, we compared the performance between the tested methods and the proposed tracking method in different ARM devices with similar hardware setup but different operating systems. For all operating systems, we used a native implementation of the design. In some instances (on iOS and Android), we had to use the primary language of the system (Objective-C and Java respectively) for the deployment, camera image acquisition and method execution calls.

We compared the accuracy of the proposed solutions with four different approaches to face fitting [6–8, 85]. The list included two different two-step methods for 3D alignment [6, 7] for desktop PC and two single-step 3D landmark detection methods [8, 85] for desktop PC (in [38] the same authors propose an efficient method for landmark detection, but this method only detects the 2D positions).

For the comparison, we included the baseline fitting method FFBP using the learning-free landmark detection method exposed in section 3.2 (FFBP), following the original approach presented in [26]. In addition, we combined the MSBP method with the estimation given by two different 2D landmark detection methods (Kazemi and Sullivan [18] and Baltrusaitis et al. [6]). An example result of each method can be seen in Figure 4.8.

The dataset adopted as ground truth in the accuracy comparison experiments is called AFLW2000-3D and was generated using an automatic 3D face fitting method proposed by Xiangyu et al. [129]. It includes the first 2000 images from the AFLW dataset [130] with a list of 68 annotated 3D points following the same distribution (see Figure 4.2) for each image. Moreover, as all the methods in the comparison experiments need an initial face region detection, the entire dataset was processed using the MMOD face detection method. We included five different MMOD models (front view, full-profile view and half-profile view) for different face out of plane orientations. After removing the images with extreme out of plane rotations and the images without face detection, the final dataset included 1357 images of the original dataset.

For the face tracking, we used *300VW challenge* dataset [127]. Each video sequence in this dataset shows a human talking and moving in front of the camera in non-controlled environments. The dataset included videos with different frame sizes, from 1280x720 to 480x360, and with a frame-rate close

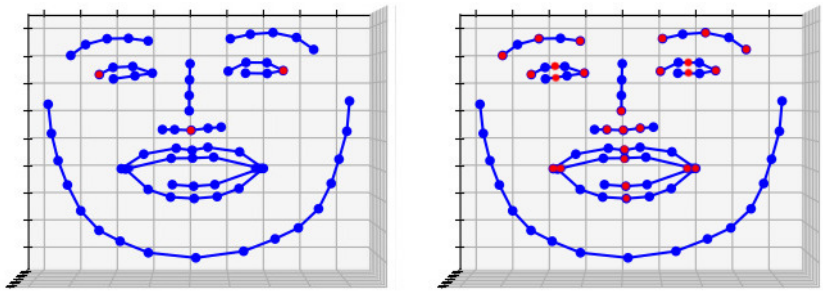
to 30 frames per second.

4.4.2 3D orientation comparison

A widely adopted approach to measure the accuracy of a fitting method is to compare the re-projection of the fitted model with a set of ground truth 2D landmarks. However, this measure does not consider the orientation of the 3D reconstruction. As shown in Figure 4.8, the 3D orientation of the face model can differ between methods even though the projection of the model is apparently correct in all of them. This experiment compared the orientation estimation of the different methods.

The ground truth dataset does not include an explicit 3D orientation value for each image, and some of the tested methods do not estimate this information. Thus, the orientation is measured using three representative face feature points. The same strategy is used to extract the face orientation given by the tested methods to avoid possible biases introduced by the differences in the definition of the 3D model in each technique.

As all the methods proposed for the accuracy comparison experiments (including the ground truth dataset) have different 3D model definitions (see Figure 4.8), a unified face model is proposed. We define the unified model by a subset of vertices representing Face Animation Parameter (FAP) defined by MPEG-4 international standard [131] and mostly present in the models of all methods.



(a) (b)

Figure 4.9: Example of an annotation of the AFLW2000-3D dataset with the ground truth vertices in blue. Each image shows in red the FAP locations for: a) the orientation estimation and b) the shape estimation comparison.

For the orientation computation, a set of three points are selected using

the same MPEG-4 international standard [131]. Figure 4.9 (a) shows a visual representation of the selected feature points. We have selected three points considering these requirements:

1. The points are present in all models used by the tested methods.
2. The relative 3D position of the points is not significantly changed by face gesture deformations.
3. The points have enough information to estimate the face orientation.

From this definition, we have selected the FAPs defined for the left eye left corner (FAP 3.12), right eye right corner (FAP 3.7) and the base point of the nose (FAP 9.15).

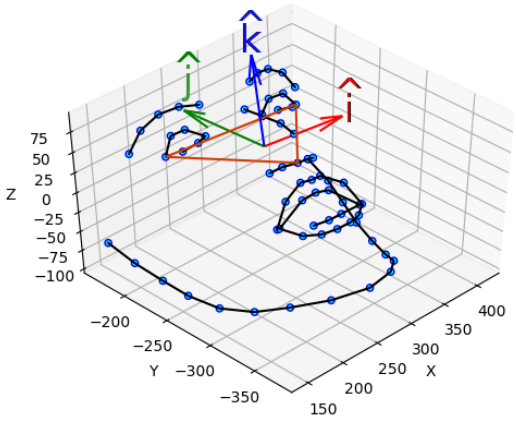


Figure 4.10: Graphical representation of the face orientation, including the reference plane.

Once a face pose estimation is given, the orientation is measured as follows. First, we transform all the model points to the same coordinate system, using the ground truth dataset as reference. Then, we define a plane using the three reference points, as shown in Figure 4.10. The 3×3 rotation matrix R_m is computed using the unit vector described by the points FAP 3.12 and FAP 3.7 as $\hat{i}$ axis, the normal vector of the plane as $\hat{k}$ axis, and the cross product between $\hat{i}$ and $\hat{k}$ as the $\hat{j}$ axis.

Using this process, we compute the R_m rotation matrices for all face pose estimations given by each evaluated method. We also calculate the rotation

matrices for each of the estimates in the ground truth dataset, which we distinguish by calling it R_g.

Based on the ground truth rotation matrix R_g and the estimated rotation matrix R_m the orientation error is computed using (4.13). If both orientations are equal, the result of the multiplication is the identity matrix, meaning that the orientation error is zero. To reduce the measurement to a single value, we transform the error matrix R_e into an *axis-rotation* representation using the Rodrigues formula, and we only consider the angle value.

$$R_e = R_g R_m^T \tag{4.13}$$

The results of the orientation comparison are listed in Table 4.2. As the mean rotation error over 1357 images (in radians) shows, the accuracy of MSBP method improves the results of most of the compared methods -those proposed by Huber et al. [7] and Feng et al. [85] being the only methods with better orientation accuracy. Note that even though the reference 3D landmarks for the measurements are the same in all cases (shown in Figure 4.10), different 3D models are used in some instanced (shown in Figure 4.8). These 3D models are deformed by the applied approach based on their shape and deformability. In the case of [6], its projections rely on weak-perspective instead of full-perspective (the one considered in our case). Therefore, some assumptions are required that might not be valid in some applications (e.g., if the face is quite close to the camera viewpoint, and the depth of the facial parts cannot be considered to be in the same plane).

In these experiments, [6] obtains slightly better results than [6] + MSBP fitting. However, taking into account these considerations, we believe that is because the 3D model used in [6] for its own fitting process is more detailed than the one we have used for our fitting. Nevertheless, the results from [6] vs [6] + MSBP fitting are similar to each other, with the remarkable difference that [6] requires weak-perspective assumptions, and that [18] + MSBP fitting obtains the best results of all. It is also relevant to see the differences between [18] + MSBP fitting and [6] + MSBP fitting, both using exactly the same 3D model, to better understand how the fitting approach improves accuracy when more accurate 2D points are provided even though the 3D model is not as detailed as that of [6].

Table 4.2: Results of the Orientation Estimation Comparison.

Method	**Mean Rotation Error** (radians)	**Error Standard Deviation**
FFBP	0.2299	0.1152
Huber et al. [7]	0.1634	0.1112
Baltrusaitis et al. [6]	0.2112	0.1952
Bulat et al. [8]	0.2088	0.1082
Feng et al. [85]	0.1132	0.1179
MSBP (points by Kazemi and Sullivan [18])	0.1875	0.1185
MSBP (points by Baltrusaitis et al. [6])	0.2694	0.1912

4.4.3 3D Shape Estimation Comparison

We used the 3D shape of the models as a measurement of accuracy because of the lack of ground truth data based on model parameters. Moreover, as different approaches use different parameters for their models, a comparison in the parametric space is not possible. Note that accurate 3D shape estimation means that the algorithm accurately recovers both types of shape parameters: the SUs and the AUs.

As in the case of the orientation, the shape estimation comparison needs a unified model definition for all the tested methods. Similarly, we made the unified model by using a subset of vertices representing FAPs defined by MPEG-4 international standard [131] and mostly present in the models of all methods. Some of the points are not present in all, but can be easily computed using a combination of two other vertices of the model. The selected 26 FAPs are 4.6, 4.4, 4.2, 4.1, 4.3, 4.5, 3.12, 3.14, 3.8, 3.10, 3.11, 3.13, 3.7, 3.9, 9.3, 9.4, 9.15, 9.5, 8.4, 8.1, 8.3, 8.2, 2.5, 2.2, 2.4, and 2.3.

Figure 4.9 (b) shows an example of this model, represented by red dots. In this example, it can be seen that, while the top and bottom FAPs of each eye are not present, they can be easily computed as the centre position of the vertices on both sides of those FAPs.

To measure the shape reconstruction error between the ground truth and the estimation given by a tested method, the defined subset of vertices is extracted from each model. Both subsets are aligned using the Procrustes analysis to scale, translate and rotate them. Then, we measure the shape

error as the sum of the squares of the pointwise differences (L_2 distance) between both vertex sets.

Table 4.3: Results of the Shape Reconstruction Comparison.

Method	**Mean Shape Error**	**Mean Fitting Time** (secs)	**Fitting Time Deviation**
FFBP	0.0182	0.0017	0.0006
Huber et al. [7]	0.0114	0.0471	0.0134
Baltrusaitis et al. [6]	0.0136	0.0116	0.0040
Bulat et al. [8] (GPU)	0.0072	0.0358	0.0053
Feng et al. [85] (CPU)	0.0052	0.2291	0.0266
MSBP (points by Kazemi and Sullivan [18])	0.0117	0.0054	0.0003
MSBP (points by Baltrusaitis et al. [6])	0.0145	0.0128	0.0041

Table 4.3 shows the mean shape error across all the images in the used dataset (1357 images). The methods proposed by Bulat et al. [8] and Feng et al. [85] have the best results. Note that part of the AFLW2000-3D dataset was used to generate the model used by Bulat et al. [8]. In the case of Feng et al. [85], the good accuracy of reconstruction requires significant computation time compared with the other methods. The MSBP approach shows good performance, coming close to the method with the second best accuracy-time results [7]. Note that in the comparison, for [6] + MSBP fitting, we only used the 2D landmark positions provided from [6], ignoring the 3D deformable model that it uses internally, but which was also computed, as it is part of the process to get the 2D landmarks. This explains why [6] + MSBP fitting is slightly slower than [6] alone.

4.4.4 Computation Time Comparison

In this section, we compare the mean computation time of each method while they try to estimate the 3D face shape in the previously defined 1357 images of the AFLW2000-3D dataset. We used a desktop PC to measure

the performance because the implementation of most of the tested methods was not focused on ARM architectures. The computer used had an Intel i7-4770K CPU (@ 3.50GHz), 16 GB of physical memory and Ubuntu 17.10 as OS.

For the method described in [8], an extra GPU was used because of the special needs of the approach.

The time measure only includes the fitting time of the image, excluding the time needed to load the model, image acquisition, face region detection or any other pre-processing task.

Table 4.3 shows the mean processing time (in seconds) for each tested method and Figure 4.8 shows an example of each fitting. The method presented in [26] was the fastest in the experiments, but it also had the worst results in the accuracy testing (see sections 4.4.3 and 4.4.2). The method proposed (Kazemi and Sullivan [18] + MSBP) showed the second shortest processing time. It outperformed the rest of the tested methods, while the accuracy was similar to most of the other approaches. Huber et al. [7], Bulat et al. [8] and Feng et al. [85] have better reconstruction accuracy, but they also needed more time.

For a better visual comparison between the tested methods, Figure 4.11 shows all the values in the same plot, normalised on a unified scale. We exclude Feng et al. [85] in this figure because the required computation time makes it unsuitable for real-time tracking systems, making it difficult to compare to the other methods.

4.4.5 Performance Analysis on ARM Architectures

In this section, the performance of one of the proposed methods is tested using different ARM devices with similar computational power. We chose our proposed MSBP fitting method that relies on the face points provided by the Kazemi and Sullivan [18] method because it has the smallest computation time measured on a PC.

The ARM devices used in the experiment are: 1) iPhone SE (using iOS 10.11), with an A9 (ARMv8-A dual-core @ 1.85 GHz) processor as CPU, 2) a Pixel-C (using Android 6.0), with a Tegra X1(ARMv8-A quad-core @ 1.9 GHz) as CPU, and 3) a JetsonTX1 (using Ubuntu 16.04), with an A57 (ARMv8-A quad-core) processor as CPU. The hardware specification in all the cases is similar. Even the JetsonTX1 device is ready to use the integrated Nvidia GPU units; the GPU was excluded from the experiment, using only

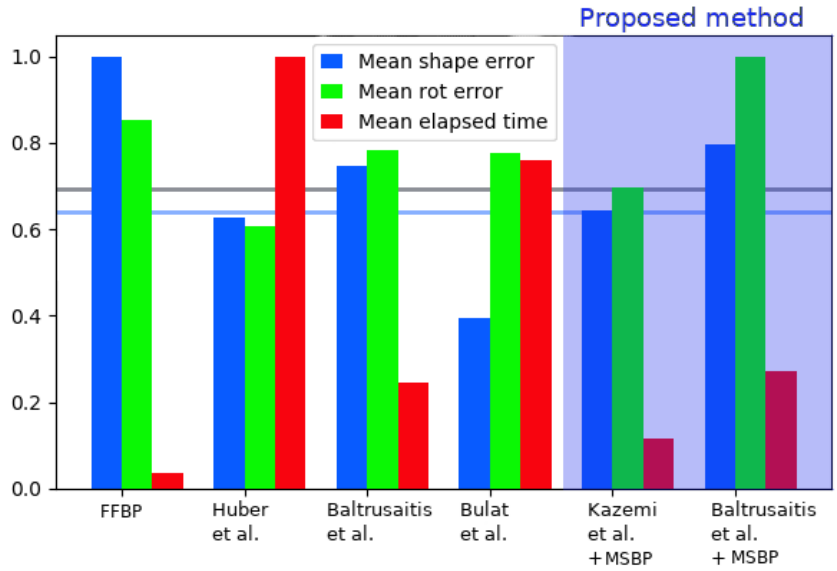


Figure 4.11: Mean shape error, mean rotation error and mean elapsed time measurements for each of the tested methods, normalised in a range from zero to one. The highest value for each measurement type is set to one while the rest of the values are scaled accordingly. The two horizontal lines show the accuracy values of our fastest method against the rest. In all cases, smaller is better.

the installed CPU. The PC uses Ubuntu 17.10 as operative system.

First, we measured the time of the different stages of the proposed method (face detection, facial landmark detection, and 3D model fitting) on each device separately. This experiment did not rely on a tracking method, so it detected the face using the face detection methods for each frame. This test shows the different computation times needed for each part of the method, allowing us to compare the performance of the ARM platforms and PC architecture.

For this test, we used the AFLW2000-3D dataset as in previous sections, with images of 450x450 pixels. Figure 4.12 shows a visual representation of the results. Table 4.4 presents the mean computation times for each platform and task. From the original 2000 images in the dataset 1490 faces were detected, so the landmark detection and the fitting method were applied to 1490 faces (including faces not labelled in the original dataset).

The full-face detection step is the most computationally demanding of

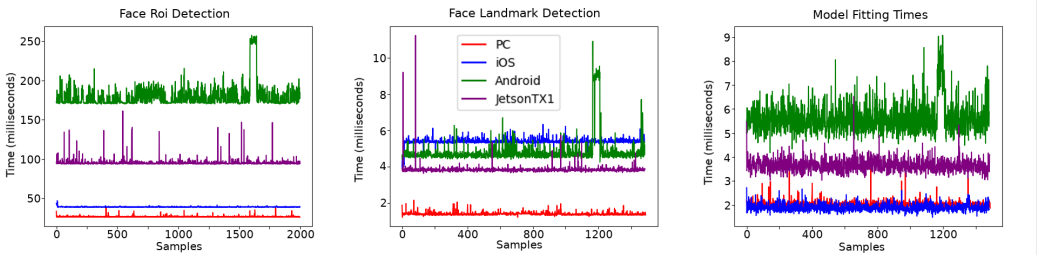


Figure 4.12: Times needed for each device to detect the face in the image(left), to detect the facial landmarks in the detected region (center) and to fit the 3D model using the detected landmarks (right). The vertical axis shows the time (in milliseconds) and the horizontal axis the sample index.

all platforms, but Android and JetsonTX1 need significantly more time than iOS and PC to achieve the same task. For the landmark detection, the ranking changes, the iOS device being the slowest, but with just a minute difference considering the case of face region detection. For the 3D model fitting, Android is again the slowest, and iOS device and the PC showing very close performance.

Table 4.4: Mean Computation Times for each Device and Task.

Device	**Arch.**	**OS**	**Face Region**	**Face Landmarks**	**Model Fitting**
PC	x86-64	Ubuntu 17.10	26.24ms	1.4ms	2.02ms
iPhoneSE	ARM	iOS 10.11	38.77ms	5.39ms	1.89ms
Pixel-C	ARM	Android 6.0	178.92ms	4.84ms	5.61ms
JetsonTX1	ARM	Ubuntu 16.04	95.06ms	3.84ms	3.65ms

Additionally, we tested the tracking method on each device to check the suitability of an entire tracking scenario. This experiment measured the time needed to process each frame of a series of video sequences. The video dataset used in the analysis was the *300VW challenge* dataset [127]. All the video files were processed twice on each device; the first time using the face detection method on each image and then fitting the 3D face model as described in section 4.2.2, and the second time using the tracking method

proposed in section 4.3.

Those two scenarios show the performance of the tracking method on each ARM device, and the performance difference between the tracking method and the continuous re-detection approach. Moreover, the correctly tracked frames are also counted to show the number of frames successfully processed with each technique. The detection method used in both scenarios (tracking and re-detection) is MMOD because of its robustness and high detection rate.

The time measure only includes the processing time of the image, excluding the time needed to load the model, image acquisition or any other pre-processing task. However, the face detection and filtering computation times varies depending on the image size, so the expected processing time was higher than in the previous experiment. We only used the CPU for the calculus, excluding any extra hardware component like GPUs.

Table 4.5: Processing Time Comparison Between Devices.

Device	**Method**	**Avg. Time** (ms)	**Fps**	**Correct Frames**	**Correct Frames** %
iPhoneSE	Tracking	19.6	51.0	186466	91.77
iPhoneSE	Re-detect	525.6	1.9	178332	87.92
Pixel-C	Tracking	57.5	17.4	184550	91.49
Pixel-C	Re-detect	799.0	1.25	177486	87.99
JetsonTX1	Tracking	33.5	29.8	185110	92.44
JetsonTX1	Re-detect	287.4	3.47	177342	88.41

Table 4.5 shows the experimental results for each device using the 201,705 frames of the *300VW challenge* dataset [127]. It lists both test cases (tracking and continuous re-detection). The estimated frames per second values are calculated using only the time needed for the frame computation. A frame is set as correct if the mean square error (MSE) between the estimated landmarks and the ground truth landmarks (both normalised) is less than an experimentally defined threshold (0.03 in this case).

Compared with the re-detection scenario, the tracking method had better results on performance and robustness, being faster and more reliable.

Moreover, the tracking method could track the face in more frames than using the face detector in each frame. The tracking method reached real-time performance ($\backsim$30fps) on all devices.

An interesting point here is that, apparently, the hardware used is not the only element to take into account. The Android device shows worse results than others with similar specifications.

4.4.6 Qualitative results

This section presents some qualitative results of the proposed method.

Regarding the fitting method presented in section 4.2, and beyond the quantitative experiments, Figure 4.13 shows some examples of the final fitting results. The projection of the final model on different in-the-wild face captures can be seen, including partial occlusions (i.e., hair and glasses), different illumination and various facial orientations. Despite these variations, the results are visually accurate - it is possible to reconstruct the direction and gesture of each face.

Figure 4.13: Some results of the fitting process using the AFLW2000-3D dataset.

In the case of the fast-tracking approach, the proposed method achieves real-time performance tracking on smart-phones using the embedded camera

for the capture. Figure 4.14 shows the proposed method running in two of the devices used in the experiments. In both cases, the implementation runs in real-time (29-30 fps on iPhoneSE and 26-27fps on Pixel-C).

Figure 4.14: Proposed method running on an iPhoneSE (left) and a Pixel-C(right) using images captured with the integrated front camera in real-time.

With regard to the other methods tested in ARM devices, in Cao et al. [81] the authors analyse the performance of the method using an iPhone7 for the experiments. Their method focuses on virtual avatar puppetry and virtual makeup and includes a stabilisation system to increase the fitting accuracy and robustness. However, it requires more computation, achieving a running speed of 20 frames per second. The MSBP method achieves 30 frames per second on an iPhone SE, which is one generation older, and thus, has lower computational power. Although the method proposed by Cao et al. [81] is more robust and precise, its computing needs are higher, making it unfeasible in computationally constrained devices.

In Deng et al. [9], the authors test the proposed method on different kinds of devices including an ARM device with an ARM-RK3399 processor, which includes a dual-core Cortex-A72 and a quad-core Cortex-A53. Their approach achieves a processing speed of 16 frames per second, ignoring the time cost of the included dense regression step. Compared with the device used in our experiments, the ARM-RK3399 processor has significantly higher performance, especially if the separate NEON processor included in ARM-RK3399 chip is in use during the tests.

4.5 Conclusions

This section presented a method to efficiently align deformable 3D face models to faces in images and track faces in video sequences. The computational

needs of the entire tracking pipeline make it fast enough to use it in limited environments such as on-board systems and smart-phones.

The experimental results show how the reconstruction and estimation accuracy is close to other state-of-the-art methods, while maintaining a fast processing speed. The method reconstructs face gesture, shape and orientation even with significant out-of-plane orientations, different illumination and partial occlusions.

We tested the approach on different devices with limited computational power, including smart-phones and small embedded systems oriented to the automotive industry. The results show how it correctly tracks a user with real-time performance in several video sequences and using real-time captures.

The experimental results also show how the operating system used for the implementation has a significant impact on the final performance of the application.

Chapter 5

Body pose estimation with multi-body deformations

Following the same scheme presented in previous chapters, we can fit multi-body based deformable 3D models to a set of 2D feature landmarks in an image. Considering the camera configuration and adopting a full-perspective projection model, the final result represents the 3D pose and location of the body pose in the real scene. Figure 5.1 shows this scheme, adding certain specific elements of the body pose reconstruction case.

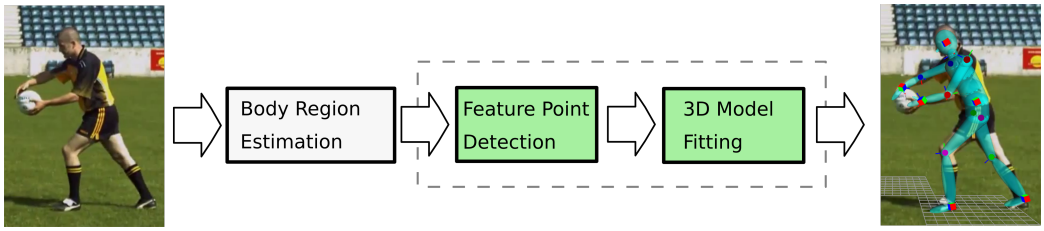


Figure 5.1: An adaptation of the graphical representation of the process to fit the 3D model presented in section 1.2.

This chapter presents a hierarchical optimisation procedure that adjusts a parametrised multi-body 3D kinematic model, based on the combination of efficient perspective-n-point camera pose estimation and constrained viewpoint-dependent inverse kinematics. It explicitly considers and preserves the relations between the 3D subject's overall scale; the level of relevance between the overall scale and the specific body part sizes; its depth with respect to the camera; and its configuration related to the reference floor

of a predefined 3D world. The proposed method, which will be referred as *Contextualized Learning-Free Body Adjustment* (CLFBA), works efficiently, without the need of learning 2D/3D mapping models from the training data. Therefore, it is not affected by data characteristic differences between training and deployment stages.

We consider using automatic body feature detectors like those presented in [25, 68, 69] to detect the 2D body features in the image.

5.1 Method

Figure 5.2 shows the components of the CLFBA method. First, the camera configuration is done by adjusting the reference floor of the predefined 3D world to four key-points in the image. For many human body tracking applications, it is enough to use a static camera without zooming operations, so in such cases, the camera should be configured only once, during the installation stage. Otherwise, the reference floor must be readjusted each time the camera is moved or zoomed, e.g., manually or by applying visual SLAM techniques [132]. Once the reference floor is adjusted to the observed scene, the body pose estimation procedure can be applied under a tracking-by-detection scope.

5.1.1 Camera calibration

The camera calibration process is based on the approach presented in [133]. This method estimates the intrinsic and extrinsic parameters of the camera by adjusting a planar rectangle representing the floor in the 3D space being observed, which will be shared by all entities in the scene. For this adjustment, the user manually annotates the 2D positions of the four corners of the rectangle on the image and assigns metric measurements to its two sides. These data are then used to compute the homography between the image and the floor planes using the DLT (Direct Linear Transform) algorithm [5]. Once the homography has been calculated, the rotation and translation of the camera are extracted from the resulting matrix. Finally, the re-projection error over the set of camera parameters is optimised using the Levenberg-Marquardt (LM) non-linear optimisation method [134,135]. This approach is valid for cameras without considerable distortion, which is the case of typical video cameras used for broadcasting and surveillance. This optimization ap-

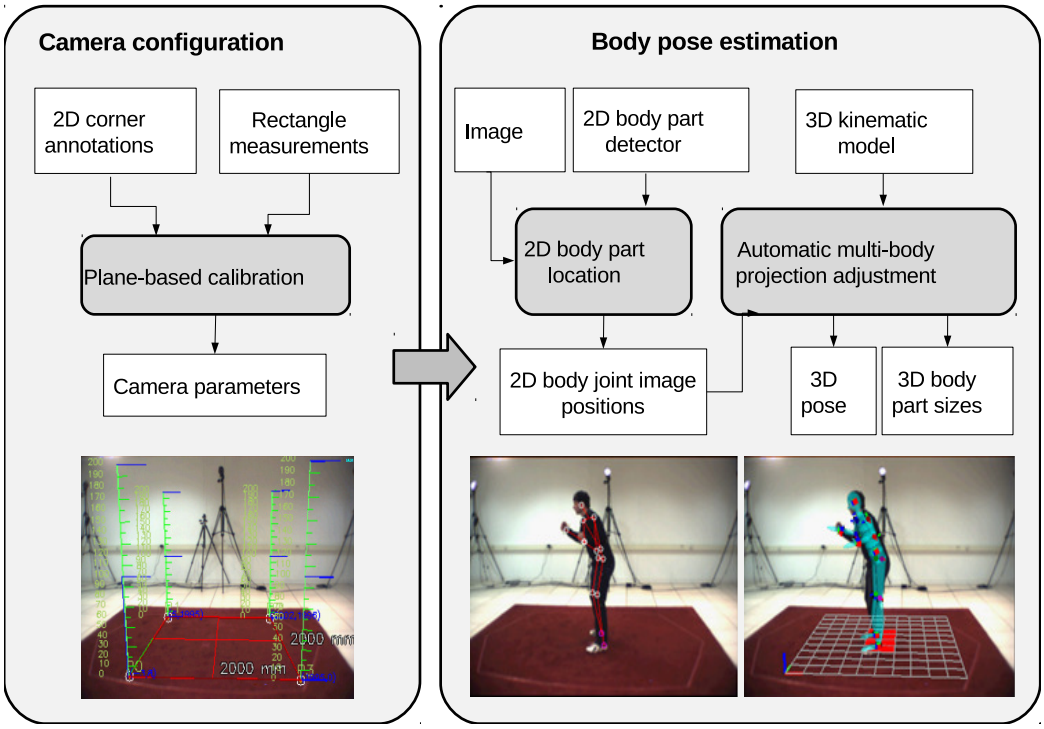


Figure 5.2: The CLFBA method components with their corresponding inputs and outputs.

proach runs efficiently, and therefore, while the user is setting the corner 2D positions and the side sizes, the tracking system can show augmented reality feedback about the 3D placement of the reference floor, in real-time. More specifically, lines perpendicular to the plane are drawn at each corner, with their estimated metric measurements, so that users can check the suitability of the obtained results during the camera configuration (left image in Figure 5.2).

5.1.2 Body pose estimation

Once the camera has been calibrated, the system continuously processes images from the video stream. First, the 2D body joint positions are located in the input image by state-of-the-art body part detectors such as [25, 68, 69]. Then, the 3D kinematic model is adjusted to the 2D joint positions, also considering its possible interaction with the floor reference. During this

adjustment process, the parameters of the body are estimated in order to optimise the overlap of its projection with respect to the 2D positions located in the image. This is an ill-posed problem as the kinematic model has many parameters that cannot be obtained directly from the image data due to non-measurable depth variations, self-occlusions, and the irreversible perspective projection. To solve this problem under a learning-free scope, a set of constraints are imposed, such as the biomechanical rotation limits of joints, body symmetry, multi-body kinematic motion constraints, or the relation of the body joints with respect to the reference floor. The relevance of the parameter variations is taken into account hierarchically during the projection overlapping process.

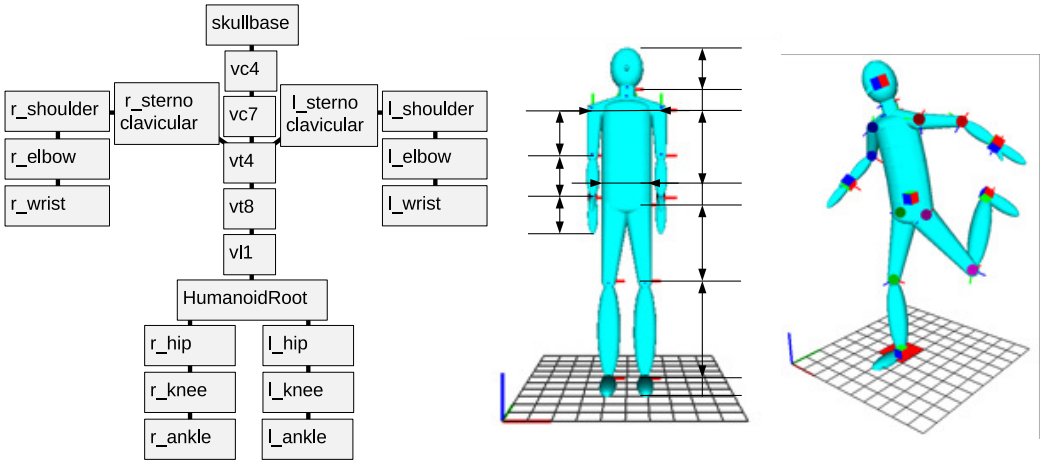


Figure 5.3: (a) The 3D kinematic model's hierarchical structure (based on H-Anim [136]), (b) its body dimension parameters and (c) its posing features for IK control (located at end-effectors and intermediate upper and lower limb joints).

A key component of the method is the parametrised multi-body 3D kinematic model, with a posing mechanism in the 3D metric space based on constrained viewpoint-dependent inverse kinematic (IK) (i.e., IK that considers the in-plane motion of posing features and body motion constraints, such as body joint rotation constraints and collisions in the 3D world). Figure 5.3 shows the hierarchical structure of joints, which follows the H-Anim standard [136]; the parameters to control the lengths of the body parts; and the posing features used for IK. The selected kinematic model has 21 joints (listed in Figure 5.3 left), with 54 degrees-of-freedom (DoF); six for the root

joint and three for the rest, except for sternoclavicular joints (no twist), elbows (only flexion and twist), wrists (no twist) and knees (only flexion). Nevertheless, the method supports a different number of joints in the spine (if required). There are twelve parameters that control the sizes of the body parts, and two parameters that control the total length of the back and neck (d6 and d8), which distribute the lengths of each body segment inside them proportionally. The sizes of the graphical objects that represent the body segments vary proportionally to their related body part lengths in order to adapt to different scales appropriately.

In addition, there are fourteen posing features located at key joints which allow the model to be posed through IK. The posing features at the end-effectors (represented as cubes in Figure 5.3-c) allow the variation of their positions and orientations, while the rest (represented as spheres in Figure 5.3-c) only their positions. The IK strategy varies the configuration of the body differently, depending on which posing feature is moved. Then, the motion of each body limb (arms and legs) and the central part of the body (pelvis, trunk, neck, head) are distinguished separately.

In the case of the arms and legs, moving their posing features only affects their corresponding body limb, but not the rest of the body:

- When the end-effector posing feature (wrist/ankle) is moved, the two segments of the limb are moved.
- When the intermediate posing feature (elbow/knee) is moved, the limb rotates around the axis between the limb's root (shoulder/hip) and the end-effector (wrist/ankle).
- When the posing features at the sternoclavicular joints are moved, the shoulders and their corresponding full arms are moved as rigid bodies, rotating around the sternoclavicular joints.

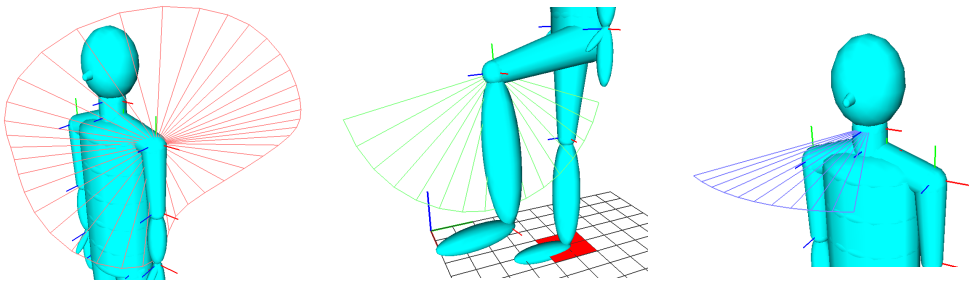
In the case of the central part of the body, the motion of its posing features (at the root joint, head and hips) affects the configuration of the whole structure:

- When the posing feature at the root joint is moved, the whole body is moved as a rigid solid in the 3D space.
- When the posing features at the hips are moved, the pelvis is rotated around the root joint, while the rest of the posing features are kept

in place. Therefore, the IK procedure makes the body adapt to the relative change of the posing features in order to match them.

- When the posing feature at the head is moved, the spine joints are bent, giving major preference to the neck, and allowing the translation of the root in the spine's axial direction. Additionally, the arm and leg end-effectors are kept in place, and therefore the IK procedure makes arms and legs adapt to them, as shoulders and hips move.

This IK strategy has proved to be adequate for the posing of the kinematic model with viewpoint constraints. In this study, this process is automated, as explained in the following section. The motion of the joints is constrained by biomechanical rotation limits that prevent unfeasible poses (Figure 5.4). These joint rotation limits are parametrised with circumduction-swing-twist rotation angles in order to model complex biomechanical rotation limits. This is a more appropriate strategy compared to other alternatives such as constraining Euler angles or exponential maps [27], as the workspaces are closer to the real behaviour of human body joints. Finally, posing features are prevented from surpassing the floor, which is also considered by the IK procedure to correct the body poses accordingly.



(a) (b) (c)

Figure 5.4: Examples of biomechanical rotation limits of body joints: (a) shoulder swing limits, (b) knee flexion limits and (c) a cervical vertebra twist limit.

5.2 Automatic multi-body projection adjustment process

This process corresponds to the automation of the hierarchical constrained 3D body adjustment shown in Figure 5.2 and Figure 5.5. The process aims to minimize the overall projection error between the 2D joint positions automatically obtained by the body part detector and the image projections of their corresponding joints in the 3D kinematic model, taking into consideration the kinematic and floor-contact constraints and the hierarchical relevance of the parameter variations. Thus, the objective function is the following:

$$e = argmin \frac{1}{n} \sum_{j=1}^{n} (p_j - w_j)^2 \tag{5.1}$$

where $\mathbf{p} = \{p_1, p_2, p_3, ...\}$ are the detected/annotated 2D joint positions, which include the following posing features: root, hips, knees, ankles, shoulders, elbows, wrists and head; $\mathbf{w} = \{w_1, w_2, w_3, ...\}$ are the 2D projections of the corresponding 3D body posing features; n is the number of (annotated/detected) 2D joint positions; and e is the residual error.

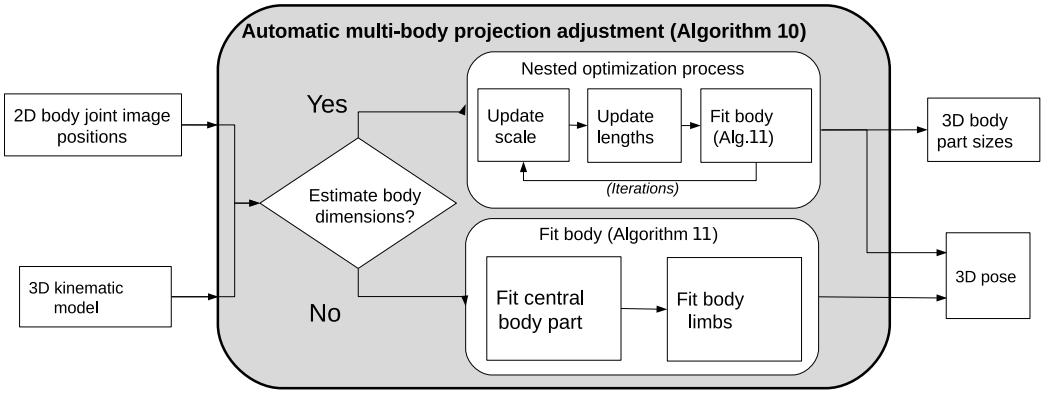


Figure 5.5: Flowchart of the automatic multi-body projection adjustment process.

Algorithm 10 shows the entire proposed method to automatically obtain the 3D pose and body dimension parameters from the image observations, while Algorithm 11 shows the procedure to fit a kinematic 3D model with

specific dimensions to 2D annotations, which is done during the optimization. There is a nested loop of two layers in the optimisation procedure, in which both try to minimise the objective function (5.1), with independent termination criteria (maximum residual error value, minimum residual error variation value, the maximum number of iterations), but hierarchically sharing the parameter values. In the outer loop, a coarse variation of body dimensions is considered in order to change the body scale. In contrast, refined variations are done for each body dimension separately in the inner loop. This way, a major priority is given to the scale variations which have a larger influence in the correct posing than those of each body part separately. The outer loop optimisation is solved with the method [137], while the inner one is solved with the method [138]. The former, known as DIRECT-L algorithm, is the *locally biased* variant of DIRECT (DIviding RECTangles algorithm), which is a deterministic-search algorithm based on systematic division of the search domain into smaller and smaller hyper-rectangles (see [137] for details). The latter, known as COBYLA (Constrained Optimisation BY Linear Approximations) algorithm, is a derivative-free optimisation procedure with non-linear inequality and equality constraints. It constructs successive linear approximations of the objective function and constraints via a simplex of $n+1$ points (in n dimensions). It optimises these approximations in a trust region at each step (see [138] for details).

The notation in the algorithms is the following:

- $\boldsymbol{K(X_0, d_0, b)}$: 3D kinematic model with initial default parameter values for $\mathbf{X}$ and $\mathbf{d}$.
- $\mathbf{X}$: 3D pose parameters.
- $\mathbf{d}$: body dimension parameters.
- $\mathbf{b}$: constraints related to biomechanical rotation limits of joints, body symmetry and multi-body kinematic motion constraints.
- $\mathbf{C} = f_x$, f_y, c_x, c_y, $\boldsymbol{t}_{xyz}$, $\boldsymbol{R}_{xyz}$: camera calibration parameters.
- $\mathbf{f}$: reference floor plane annotated in the calibration stage.
- s: body scale.
- δ: body dimension parameter variations.

- ε: body scale variation.

Algorithm 10: 3D body pose and body part lengths estimation from 2D image data.

Input: 3D kinematic model ($\mathbf{K}$), landmarks ($\mathbf{p}$), camera parameters ($\mathbf{C}$), body scale (s), body dimension parameters ($\mathbf{d}_0$) and floor plane ($\mathbf{f}$)
Output: Final pose parameters ($\mathbf{X}$), final body dimensions ($\mathbf{d}$)

```
d_0 → d
if Face region available then
    while not termination condition 1: e_1 > 5^2(pixels^2)|δe_1 < 0.01|iterations_1 > 50 [137] do
        s + ε → s
        update d * s → d
        while not termination condition 2:
          e_2 > 5^2(pixels^2)|δe < 0.01|iterations_2 > 100 [138] do
            d + δ → d
            fit human body(K, p, d, C, f) → X (Algorithm 11)
            get projection error (X, C, p) → e_2
        end
        get projection error (X, C, p) → e_1
        if floor cross then
          e_1 + BIG_NUMBER → e_1
        end
    end
    update pose with x
else
    fit human body(K, p, d, C, f) → X (Algorithm 11)
end
```

In Algorithm 11, the human body fitting from 2D annotations to 3D pose is completed through two different stages. First, the central part of the body is placed by fitting the coordinates of the 3D model's shoulders, hips and body root joints to their corresponding 2D target locations through EPnP (Efficient PnP) [107]. PnP stands for *perspective-n-point* problem and refers to the problem of estimating the pose of a calibrated camera from n 3D-to-2D point correspondences. EPnP expresses the n 3D points of an object (n $\geq$ 4 for both planar and non-planar configurations) as a weighted sum of four virtual control points. The PnP problem then estimates the coordinates of these control points in the reference system of the camera (see [107] for details).

Once the central body part position and orientation are obtained, the body limbs are fitted by IK. In addition, the method takes advantage of

Algorithm 11: Fit human body from 2D annotations to 3D pose.

Input: 3D kinematic model ($\mathbf{K}$), landmarks ($\mathbf{p}$), camera parameters ($\mathbf{C}$) and floor plane ($\mathbf{f}$)

Output: Final pose parameters ($\mathbf{X}$)

Fit central body part (pelvis, trunk, neck, head):

    solvePnP(centralBody($\mathbf{p}, \mathbf{X}$)) $\rightarrow \mathbf{X}$ [107]

Fit body limbs:

    if *floor contact* **then**
        Compute intersection with $\mathbf{f} \rightarrow$ floor depth diff
        limb($\mathbf{X}$) + floor depth diff
    end
    Apply constrained viewpoint-dependent IK $\rightarrow \mathbf{X}$

the information provided by the possible interaction of the body with the predefined floor plane. If a foot is in contact with this plane, its depth is defined by the intersection of the floor plane and the perpendicular line to the 2D image that crosses the annotated 2D foot position. The possible error of the depth of the body joints is checked and corrected with this value. In cases where both feet are in contact with the floor, mean depth is computed for the corrective value applied to the different depths of the body joints. This procedure is also applicable to human body poses where body parts in contact with the floor are not necessarily feet.

In order to detect floor-contact constraints directly from image cues, the spatial relation of the reference floor and the entities in the scene with respect to the camera viewpoint could be considered, for example, by annotating as *contact* those body joint projections which lie on specific areas of the image for specific kinds of body pose projections (related to specific actions, such as walking, etc.) provided by state-of-the-art body part detectors (such as [25, 68, 69]). However, as this issue is scene and application dependent, we do not propose a general solution for it in this study.

5.3 Experimental results

Sequences from the well-known HumanEva dataset [139] were used to evaluate the performance of the CLFBA method. The dataset contains sequences of synchronised multiple views of different people performing several actions which are being captured simultaneously by optical marker-based motion capture (Vicon). We used 12 monocular video sequences from the

HumanEva-I subset, 119,097 valid frames in total (the valid ones are those with non-corrupted Vicon data), which show two different subjects (S1 and S3), performing two actions each (S1: Box1 and Walk1; S3: Box1 and Jog1), recorded from three different viewpoints (C1-C3). Figure 5.6 shows samples from these selected sequences. The resolution of the video images is 640x480. Thus, the accuracy obtained by CLFBA has been compared to Vicon and representative state-of-the-art learning-free [27] and learning-based approaches [140] and [141]. Both learning-based methods have been trained using selected sequences from the CMU motion capture dataset [142]. Hence, the considered relations between the 3D model and the 2D projections in the training stage can differ significantly from those in the testing data, as may happen during the deployment stage. The use of data from datasets generated under different conditions for training and testing is appropriate in order to evaluate generalisation problems that could arise in learning-based approaches, compared to learning-free.

In these experiments, the potential error resulting from the 2D image location of body parts was avoided, as this error corresponds to the accuracy of the detector (such as [25,68,69]) to be combined with the 3D body pose estimator. Therefore, for a fair comparison, the same 2D input data (extracted from the reconstructed Vicon skeleton and projected image coordinates), was used for all approaches.

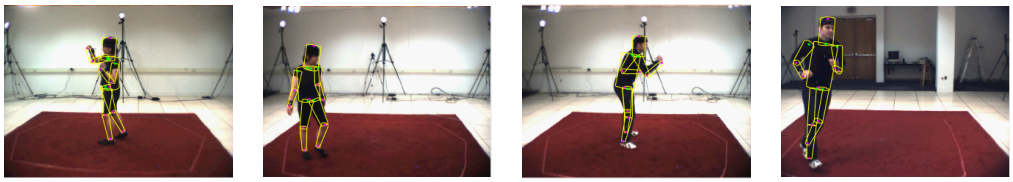


Figure 5.6: Samples from the HumanEva-I dataset sequences used in the experiments, with the pose estimations from Vicon overlapped. From left to right, S1-Box1-C3, S1-Walk1-C2, S3-Box1-C2 and S3-Jog1-C1

First, as a preliminary step for the analysis of the accuracy obtained by CLFBA, we evaluated the calibration parameters obtained by [133] (which is the method used for the camera configuration in CLFBA and [27]), compared to those accurate measures taken by Vicon. Table 5.1 shows the obtained results. Camera parameters from [140] and [141] cannot be included in this comparison as these approaches assume weak perspective projection, and therefore there is not a direct correspondence between their 3D space units

Table 5.1: Camera parameter estimation results of [133] (used for the camera configuration in CLFBA and [27]) and Vicon in the three camera viewpoints (C1-C3) from HumanEva-I dataset.

Cam.	Method	fx(px)	fy(px)	cx(px)	cy(px)	tx(cm)	ty(cm)	tz(cm)	Rx(º)	Ry(º)	Rz(º)
C1	[133]	872.56	870.64	327.18	265.68	-20.70	-25.08	510.32	74.81	75.86	-64.76
	Vicon	765.79	765.33	299.77	232.35	-20.28	-11.43	418.38	71.47	83.22	-69.51
	Error	106.77	105.31	27.41	33.33	0.42	13.64	91.93	3.34	7.36	4.75
C2	[133]	661.90	661.02	326.98	244.98	-10.12	-7.07	390.26	75.68	80.07	-65.87
	Vicon	688.42	686.79	273.00	221.97	12.73	8.40	357.61	71.47	83.22	-69.51
	Error	26.52	25.77	53.98	23.01	22.85	15.47	32.65	4.21	3.15	3.64
C3	[133]	733.90	733.01	326.98	244.98	26.57	-25.49	466.99	75.05	78.29	-66.21
	Vicon	724.27	723.57	302.63	204.91	35.18	3.14	430.55	71.47	83.22	-69.51
	Error	9.63	9.44	24.35	40.07	8.61	28.63	36.44	3.58	4.93	3.30
Avg errors		47.64	46.84	35.25	32.14	10.63	19.25	53.68	3.71	5.14	3.90
St-dev		51.90	51.29	16.29	8.59	11.35	8.18	33.19	0.45	2.11	0.76

and the world's metric measurements. In this experiment, we manually set the four corners of the reference floor, taking into account the dimensions of the scene described in [139]. Figure 5.2 shows an example of how these corners were placed.

As can be seen in Table 5.1, the in-plane translations (t_x, t_y) have less average error (about 15 cm) and variability (about 10 cm) than in the depth direction (t_z) (about 53 and 33 cm, respectively). The average depth distance with respect to the world coordinate system is about 370 cm, which results in relative average errors of around 4-14% in translation. Moreover, for each camera viewpoint, it can also be seen that the largest error variations happen again in t_z, while the measurements are more stable at t_x and t_y. These differences between in-plane (XY) and out-of-plane (Z) directions are expected because of the lack of direct measurements in the depth direction. All obtained rotations have similar error values, around 4º, but the R_y error is higher than the R_z error. This has been overseen as previously explained, due to the lack of direct measurements in the depth direction. However, the lower R_x error is due to the fact that the viewpoint benefits the estimation of the orientation in that direction. Besides, the orientation error stability from viewpoint to viewpoint is higher because of this constraint as well.

Regarding intrinsic parameters, the larger error comes from the focal length estimation (f_x, f_y), which has an average relative error of about 6% with a standard deviation of about 7%, while the average relative error for the principal point calculation (c_x, c_y) is about 6% with a standard deviation of about 2%. The main deviations can be found at camera C_1, because no appropriate image landmarks were used during the manual annotation of the reference floor corners, which resulted in a remarkable difference between the estimated plane and the real one.

Tables 5.2 and 5.3 show the 2D and 3D pose estimation errors, respectively, of CLFBA, [27], [140] and [141], compared to Vicon, for each body limb, the head, and also the full-body, including all the body joints. In the case of the 3D error, *Procrustes* alignment has been applied to the estimated 3D poses with the *ground-truth* (Vicon) to compute the error, as done in [140] and [141], because these two approaches rely on weak perspective projection and therefore the obtained 3D data cannot be measured against Vicon, directly.

It can be observed that the average lowest 2D re-projection errors correspond to CLFBA and [140], while [27] and especially [141] obtain higher error values. In the case of 3D errors, CLFBA and [27] obtain lower errors compared to [140] and especially [141]. According to these results, it seems that [140] can generalise better than [141], when there are significant differences between training and testing data, as happened in this experiment.

Learning-free approaches do not use any kind of training data, and therefore, they are not influenced by this kind of problem. However, if we analyse in detail the spatial relationship between the reference floor and the reconstructed 3D human body poses, we can observe that there might be remarkable differences between the reconstruction results obtained by CLFBA and [27], as shown in Figure 5.7 (e.g., separations between the floor and feet that should be in contact or different orientations of trunks with respect to the horizontal plane). As CLFBA handles hierarchically holistic scale variations with respect to the refined body part length variations during the optimisation, and considers floor-contact constraints explicitly, it can obtain more coherent reconstructions with respect to the shared 3D world when compared to [27].

Table 5.4 shows the errors in size measurements of the body parts in CLFBA. In this case, checking the result differences from action to action for each subject, it can be observed that the measurements can have remarkable differences. Those differences are due to the accumulated errors coming from

Table 5.2: 2D pose estimation errors in pixels per joint, for each body limb and for the full body, of CLFBA, [27], [140] and [141] with respect to Vicon in HumanEva-I dataset.

Action	**Method**	Left leg	Right leg	Left arm	Right arm	Head	Full body
$S1_{Box1}$	CLFBA	6.35	6.14	**5.99**	**5.79**	5.17	**5.85**
	[27]	9.02	8.86	7.54	7.35	**4.88**	7.73
	[140]	**5.98**	**6.01**	6.35	5.91	7.59	6.19
	[141]	22.79	22.63	18.39	17.24	12.15	19.52
$S1_{Walk1}$	CLFBA	**6.29**	**6.20**	**5.13**	**5.39**	**4.97**	**5.55**
	[27]	8.93	8.91	6.60	6.21	5.00	7.29
	[140]	7.44	6.85	7.21	7.36	6.24	7.04
	[141]	19.77	19.54	15.96	15.18	10.12	16.97
$S3_{Box1}$	CLFBA	6.88	6.84	**5.81**	**5.93**	5.90	**6.14**
	[27]	8.87	8.70	8.25	8.33	**5.23**	8.08
	[140]	**6.20**	**6.47**	6.32	6.62	7.79	6.49
	[141]	25.48	25.96	20.29	19.06	9.88	14.94
$S3_{Jog1}$	CLFBA	**6.78**	7.10	**6.04**	**6.44**	5.39	**6.32**
	[27]	8.04	8.51	7.31	7.38	**5.34**	7.39
	[140]	**6.78**	**6.75**	7.14	7.06	7.05	6.88
	[141]	23.37	23.43	18.03	17.83	12.04	19.97
Average (StdDev)	CLFBA	**6.57** (0.30)	6.57 (0.47)	**5.74** (0.42)	**5.89** (0.43)	5.36 (0.40)	**5.96** (0.33)
	[27]	8.71 (0.45)	8.74 (0.18)	7.42 (0.68)	7.32 (0.87)	**5.11** (0.21)	7.62 (0.36)
	[140]	6.60 (0.65)	**6.52** (0.49)	6.76 (0.63)	6.74 (0.63)	7.17 (0.7)	6.65 (0.38)
	[141]	22.85 (2.36)	22.89 (2.64)	18.17 (1.77)	17.33 (1.62)	11.05 (1.21)	17.85 (2.35)

the previous steps (camera calibration and body posing), depending on the perspective and the pose with respect to the viewpoint. This is expected, taking into account the lack of direct measurements in the depth direction and self-occlusions. The resulting average measurement error is approximately 9.8 cm with a standard deviation of about 3.8 cm.

Table 5.3: 3D pose estimation errors in cm per joint, for each body limb and for the full body, of CLFBA, [27], [140] and [141] with respect to Vicon in HumanEva-I dataset.

Action	**Method**	Left leg	Right leg	Left arm	Right arm	Head	Full body
$S1_{Box1}$	CLFBA	**11.81**	**11.94**	13.13	14.05	**5.41**	**11.72**
	[27]	15.17	15.72	16.43	17.49	8.14	14.93
	[140]	13.45	15.19	**12.90**	**11.17**	16.88	13.21
	[141]	17.34	16.19	23.44	24.62	12.11	19.16
$S1_{Walk1}$	CLFBA	**10.17**	**10.68**	**8.58**	**9.70**	**6.73**	**9.11**
	[27]	14.34	14.49	12.55	13.44	9.94	12.87
	[140]	15.77	15.70	15.59	15.01	16.33	14.96
	[141]	21.09	18.58	24.88	23.37	12.89	20.29
$S3_{Box1}$	CLFBA	**11.74**	**13.85**	**15.78**	**16.48**	**7.87**	**13.41**
	[27]	20.92	20.26	19.62	20.08	10.11	18.58
	[140]	17.49	20.20	20.97	20.13	21.17	19.17
	[141]	28.00	23.93	33.96	33.98	14.31	27.59
$S3_{Jog1}$	CLFBA	**11.29**	**11.21**	**10.58**	**9.63**	**4.90**	**9.83**
	[27]	15.25	15.26	13.53	12.77	7.56	13.08
	[140]	16.06	13.78	13.07	14.92	18.61	14.36
	[141]	23.31	25.12	31.73	23.75	16.32	24.07
Average	CLFBA	**11.25** (0.75)	**11.92** (1.39)	**12.02** (3.12)	**12.46** (3.38)	**6.23** (1.34)	**11.02** (1.94)
(StdDev)	[27]	16.42 (3.03)	16.43 (2.60)	15.53 (3.18)	15.94 (3.46)	8.94 (1.28)	14.86 (2.64)
	[140]	15.69 (1.67)	16.22 (2.77)	15.63 (3.76)	15.31 (3.68)	18.25 (2.18)	15.42 (2.60)
	[141]	22.44 (4.45)	20.95 (4.26)	28.50 (5.13)	26.43 (5.06)	13.91 (1.85)	22.78 (3.83)

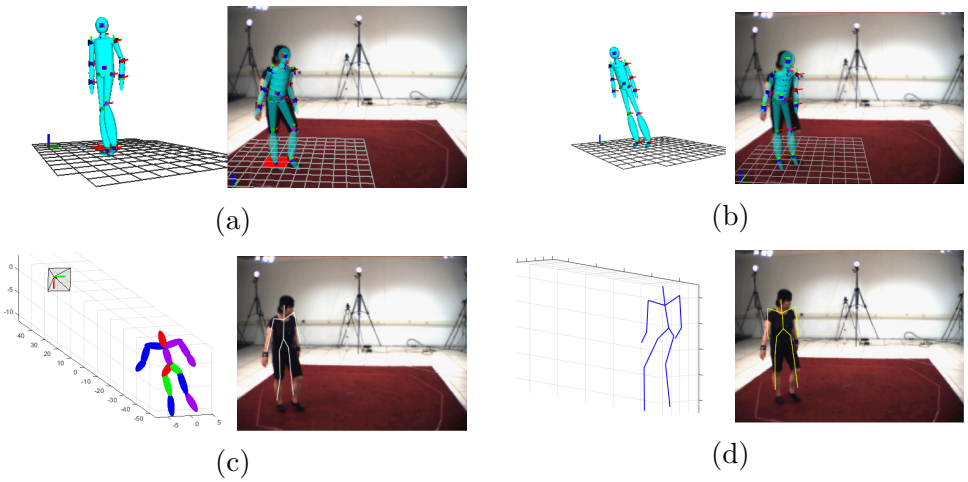


Figure 5.7: An example of the obtained pose reconstruction results (S1-Walk1-C2): a) CLFBA, b) [27], c) [140] and d) [141]

Finally, Table 5.5 shows the average time measurements required by the optimisation process, programmed in C++, and executed on a desktop PC Intel Core Quad @2.50GHz and 8 GB RAM. As can be seen, if the optimisation process is set to estimate the pose only or both the pose and the body dimensions but varying only the scale, the processing time is sufficient for real-time processing. In the case that both the pose and the body dimensions should be estimated, the method requires about one second to process, and therefore the frame rate would decrease to near real-time processing.

In the case of [140] and [141], the same testing samples required an average processing time of about five seconds. However, the results are not comparable as [140] and [141] are programmed in Matlab, and CLFBA in C++. However, taking into account the general time differences between C++ and Matlab programs in operations such as those included in these two algorithms, it can be assumed that the time consumption of the full process is comparable.

Table 5.4: Body part size estimations in cm of CLFBA and Vicon in HumanEva-I dataset

Action	**Method**	Left thigh	Right thigh	Left calf	Right calf	Left upper arm	Right upper arm	Left forearm	Right forearm	Central body length	Central body width
$S1_{Box1}$	CLFBA	30.91	30.91	38.94	38.94	21.46	21.46	19.55	19.55	55.05	28.11
	Vicon	39.65	43.03	38.58	36.66	27.61	26.43	24.92	25.82	58.83	29.63
$S1_{Walk1}$	CLFBA	33.63	33.63	42.01	42.01	24.07	24.07	22.00	22.00	56.38	28.71
	Vicon	38.44	40.35	39.57	36.55	25.07	25.73	24.98	24.41	60.12	32.99
$S3_{Box1}$	CLFBA	30.37	30.37	38.50	38.50	21.43	21.43	19.50	19.50	55.14	28.42
	Vicon	43.55	45.24	42.06	39.18	31.88	32.23	29.98	30.79	69.42	35.38
$S3_{Jog1}$	CLFBA	29.04	29.04	36.87	36.87	20.05	20.05	18.22	18.22	53.60	27.82
	Vicon	44.02	43.54	43.10	40.76	31.68	33.01	29.72	29.75	70.69	37.37
Avg. errors		10.43	12.05	3.15	3.08	7.31	7.60	7.58	7.87	9.72	5.58
StDev		4.57	3.76	2.45	2.06	4.82	5.21	4.07	4.37	6.98	3.46

Table 5.5: Average time measurements for the optimization process, depending on the parameters to be obtained

Task	**Time** (secs)
Estimate pose and body dimensions	1.097
Estimate pose and body dimensions only varying scale	0.016
Estimate pose only	0.001

5.4 Conclusions

In this section, we have presented a learning-free approach for estimating the 3D pose and the dimensions of a kinematic 3D body, contextualised in a predefined 3D world, given a set of 2D body features extracted from monocular images. This contextualisation has the advantage of providing further semantic information about the observed scene. In contrast, state-of-the-art methods typically assume the use of orthographic or weak perspective projections, and therefore the spatial 3D relation of the observed entities (e.g., people or objects) in a shared 3D space is not considered, as in the presented approach.

The presented approach handles hierarchically holistic scale variations concerning the refined body part length variations during the 3D-to-2D adjustment process, and considers floor-contact constraints explicitly, so it can obtain more coherent reconstructions for the shared 3D world, compared to other state-of-the-art approaches. Additionally, in contrast to learning-based methods, which are very frequent in recent literature, there is no need for a training stage with a database of human poses, just a set of constraints related to kinematics and their possible interactions with a reference floor. Future work will explore the possibility of including further and more sophisticated constraints in the reconstruction process, in order to reduce the solution search space, especially in the out-of-plane direction. These constraints in the out-of-plane direction can come from pre-trained data related to actions foreseen in the context of the final application. These pre-trained data could be used to initialise the automatic multi-body projection adjustment process explained in this study.

Chapter 6

Application fields

In previous chapters, we have seen various techniques to estimate the three-dimensional pose and gesture of the human face and the human body in monocular images. To show the applicability of the presented methods, in this chapter, we present two examples of final application scenarios in more detail.

The first application scenario focuses on the face 3D gesture and poses extraction to analyse the behaviour of a human driver. It is part of a driver monitoring system integrated into a driving simulator. In this context, the monitoring system is tasked with estimating the driver's attention using the information extracted from images captured with a monocular camera. Extracting the gesture and pose of the driver's face and his eye gaze using the method presented in Chapter 4, the monitoring system estimates the point of gaze of the user in the 3D space.

The second example application focuses on the body pose tracking method. It extracts estimations related to the playing style of different athletes to evaluate their performance when doing sports . The proposed method extracts the pose of various athletes from videos recorded during games played on real courts using the method presented in Chapter 5. Analysing the extracted body pose information, it evaluates the performance of each player comparing the game style with the players' considered reference.

6.1 Driver inattention monitoring

Typically, state-of-the-art eye gaze estimation techniques obtain the point of gaze (PoG) on one screen only [143]. However, in the case of driving simulators, there are usually more than one, e.g., one for the front view, one for each side view, another one for the dashboard (see Figure 6.1). Furthermore, there can be different objects of interest at different locations of each screen and obtaining the gaze fixations and saccades, derived from the PoG, accurately on each screen is important for driver behaviour analysis [144]. Additionally, it is also preferable to simplify the installation and calibration of sensors and to reduce power consumption as much as possible, avoiding alternative possibilities such as placing a dedicated PoG estimator on each screen. Thus, in our context, we only consider one monocular camera in front of the user and a humble CPU(e.g., those included in an embedded PC or a smartphone).

In automotive platforms, visual features of the face and eye regions of a driver provide cues about their degree of alertness, perception and vehicle control. Knowledge about drivers' cognitive state helps to predict, for example, if the driver intends to change lanes or is aware of obstacles and can thereby avoid fatal accidents. These systems use eye-tracking setups mounted on a car's dashboard along with computing hardware running machine vision algorithms, with computational capabilities far below from those of off-the-shelf desktop PCs. Major sources of error in automotive systems arise principally from platform and user head movements, variable illumination, and occlusion due to shadows or users wearing glasses, which need to be handled robustly but also efficiently due to the computational constraints. Current state-of-the-art of eye gaze estimation systems applied to automotive platforms include different kinds of approaches and uses. Some approaches consider eye movement features (e.g., fixations, saccades and smooth pursuits) to derive driver cognitive states, such as driver distraction [145]. Other approaches apply classification techniques to eye images related to different gaze zones, to detect where the driver is looking while driving [146]. Some approaches track facial features, 3D head poses and gaze directions relative to the car geometry to detect the eyes-off-the road condition of the driver [147]. Other approaches study the driver's gaze behaviour (e.g., glance frequency and glance time) to evaluate driving performance when they interact with other devices (e.g., a portable navigation system) while driving [148]. Finally, some approaches study the dynamics between the head pose and gaze be-

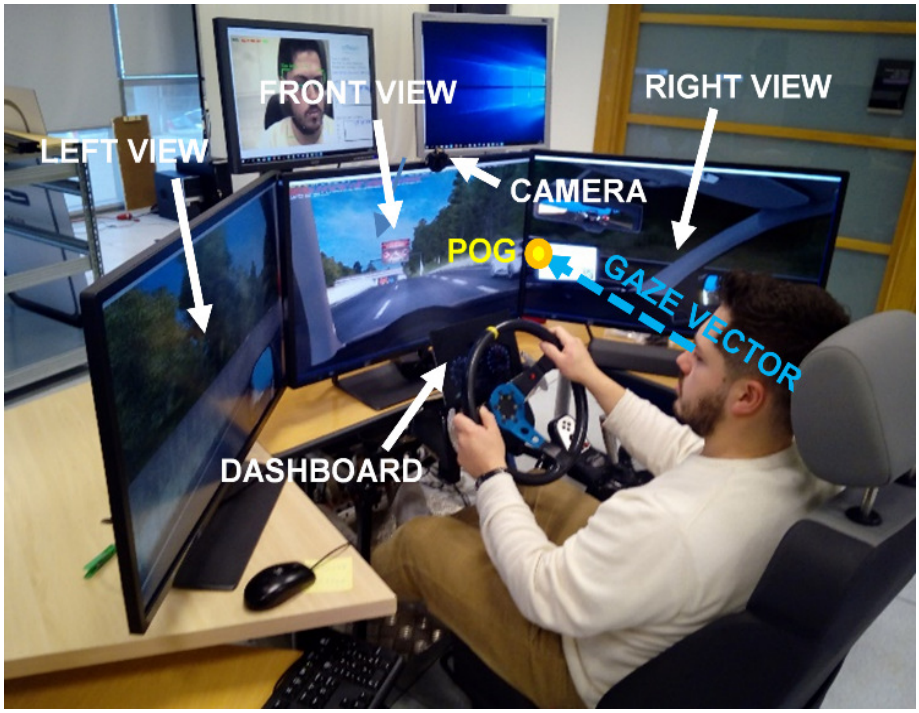


Figure 6.1: Multi-screen simulator setup for driver behaviour analysis, based on human-machine interaction, including PoG and 3D face tracking

haviour of drivers to predict gaze locations from the position and orientation of a driver's head [149] or to categorise different kinds of driver behaviour while driving [150].

The primary motivation of this application is to increase the level of sophistication of such cases by developing a more accurate, more robust, but still efficient method of estimating the head pose and eye gaze of drivers, compared to previous approaches. We paid particular attention to the case of multi-screen simulators, where we can directly establish the relation between the PoG and the rendered graphics. Therefore, the tracking system could extract richer data for behaviour analysis. In order to do so, it is necessary to relate the 2D image projections of the driver's facial and ocular cues, captured from the monocular camera, with the 3D space. Ideally, this would require not only obtaining the person's 3D eye gaze vectors from the images, but also the person's 3D eye positions and the surrounding potential targets'

geometries in the same 3D space, the camera characteristics from which that space is observed, and an additional calibration stage done by the user. However, in cases like automotive applications, it is not easy to obtain all such data. It is not comfortable for a driver to spend time calibrating the eye gaze system. Another important factor is that the estimated gaze vector should have a low level of noise while still being sensitive to quick eye movements. Finally, the estimated gaze vector should be robust enough to detect head movements, which in the case of driving, normally happen many times.

The approach to tackling all these factors consists of a hybrid procedure that combines appearance-based and model-based computer vision techniques to extract the 3D geometric representations of the user's face and gaze directions. It places all these elements in the same virtual 3D space as those of the monocular camera and the screens. This reconstructed virtual 3D world is where the driver's behaviour can then be analysed, based on the estimated PoG in the different targets of the scene and the 3D head pose, without necessarily requiring calibration data. It has been designed to have an acceptable balance between accuracy, robustness and efficiency so that it can be integrated into devices with low computational capabilities such as those that might be used in vehicles.

The following sections describe the proposed hybrid system, illustrate details about the performed experiments and present the conclusions achieved based on the results.

6.1.1 Methodology

The methods to estimate the eye gaze from monocular images and videos can be categorised into two types of approaches: model-based [147, 151] and appearance-based [146, 150, 152–156]. Next, we study more in detail the pros and cons of each, and then we explain our proposed hybrid approach.

Model-based vs appearance-based

The model-based approach relies explicitly on 3D graphical models that represent the geometry of the eye (typically as spheres) which fits the model to the person's detected eye features in the image (typically, the iris and the eye corners). Thus, the fitted 3D model allows inferring the 3D eye gaze vector, which determines where the person is looking. These methods have some drawbacks. For example, they require the precise location of the iris

of the eye in the image; this is often impossible, for example when the user's eyes are not wide open, which is the normal case. In order to estimate the eye gaze direction, they need the user's head coordinates system as reference. Therefore, the success of these methods is highly dependent on the precision with which the user's head coordinates system has been localised. Moreover, although simple, they require an initialisation scheme: the user needs to intentionally look at one or more points on a screen. Otherwise, the precision of the obtained eye vectors is not good enough. In summary, since they are pure geometric methods, their precision is strongly dependent on the precision of the estimated eyeball and pupil centres. However, it is generally not possible to obtain this information with high precision in common images.

In contrast, the appearance-based approach establishes a direct relationship between the person's eye appearance and the corresponding eye gaze data of interest (e.g., the 3D eye gaze vector) by applying machine learning techniques. Thus, it uses a dataset of annotated images to train a regression model and then deduces where the person is looking when it applies the trained model to the person's eye image extracted from the picture.

In the last few years, appearance-based methods have greatly benefited from the revolutionary results obtained by the emerging deep learning techniques in computer vision applications and have become the current state-of-the-art in the field. They allow for a much better generalisation of the learned relationship between the eye appearance and the corresponding eye gaze data than alternative machine learning approaches (based on *handcrafted* image features and *shallow* layered learning architectures). Deep learning techniques use a huge dataset of annotated images for training, with hundreds of thousands or even millions of samples. Those samples may include real data [152, 156], photo-realistic synthetic data [153, 155] or even a mixture of both [154]. This way, systems designed to estimate eye gaze direction obtain better accuracy with people whose appearance is not in the training dataset of the regression model.

However, an effective eye gaze direction estimation system does not only need to obtain accurate eye gaze data from eye images. It also has to properly apply the eye gaze data to the environment, so that it is possible to deduce where the person is looking.

Hybrid approach

Figure 6.2 shows the general overview of the workflow of our approach, where the inputs are a monocular image taken by one camera in front of the user, a parametric deformable 3D face model (Figure 6.3), the camera intrinsic parameters and the screen geometries. The outputs are his/her estimated PoG for the considered screens and his/her facial mesh in the 3D space, which includes information about his/her head position, orientation and expression. In this workflow, we distinguish three blocks: (1) the adjustment of the 3D face model to the user's face image, (2) the normalisation of the 3D gaze estimation and (3) the estimation of the eye gaze direction with respect to the targets.

The first block comprises computer vision procedures to detect and track facial regions on the image, localise facial landmarks and fit the 3D face model to those landmarks by optimising the following objective function:

$$e = \text{argmin} \frac{1}{n} \sum_{j=1}^{n} [d_j - p(f, w, h, \mathbf{t}, \mathbf{r}, \mathbf{s}, \mathbf{a})_j]^2 \tag{6.1}$$

where:

- $\mathbf{d} = d_1, d_2, d_3, ...$ are the detected 2D landmark positions.
- $\mathbf{p} = p_1, p_2, p_3, ...$ are the 2D projections of the corresponding 3D deformable model vertices. $\mathbf{p}$ is a function that depends on the camera parameters (f, w, h) and on the parameters of the graphical object ($\mathbf{t}$, $\mathbf{r}$, $\mathbf{s}$, $\mathbf{a}$). Function $\mathbf{p}$ represents the 2D projections on a surface of vertices, which are 3D. The goal is to minimise the distance between the detected 2D landmark positions in the image and the vertices of the projections.
- f is the focal length of the camera from which the image was obtained.
- w is the image pixel width.
- h is the image pixel height.
- $\mathbf{t} = t_x, t_y, t_z$ are the XYZ positions of the face model with respect to the camera.

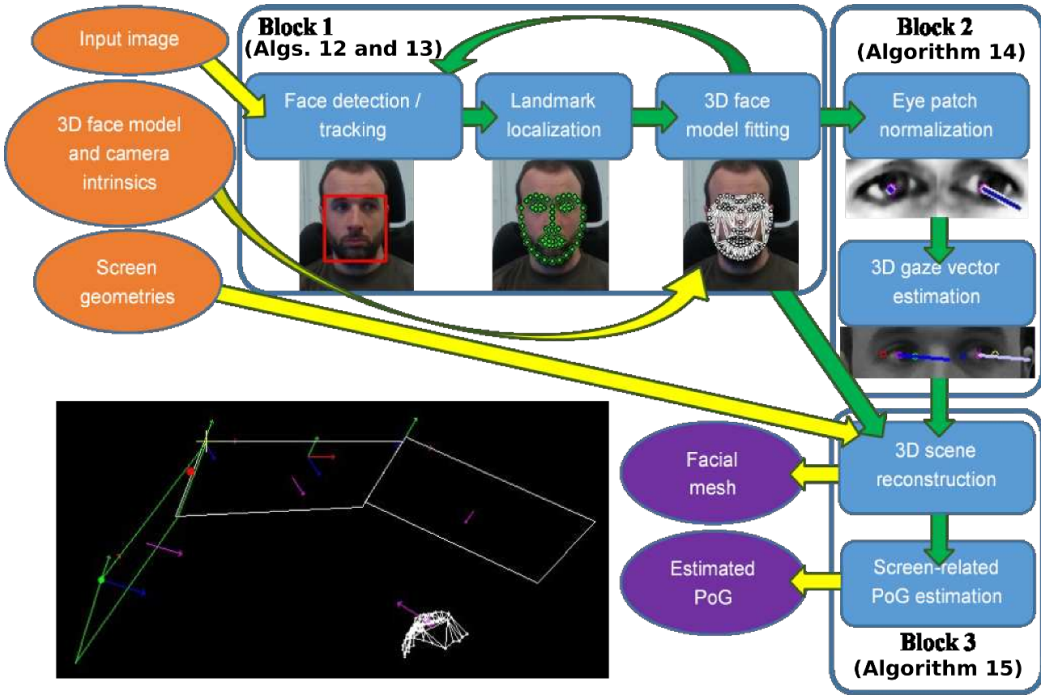


Figure 6.2: Workflow of the multi-planar PoG estimation and 3D face tracking approach.

- $\mathbf{r} = r_x, r_y, r_z$ are the roll-pitch-yaw rotation angles of the face model with respect to the camera.
- $\mathbf{s} = s_1, s_2, s_3, \ldots$ are the shape-related deformation parameters.
- $\mathbf{a} = a_1, a_2, a_3, \ldots$ are the action-related deformation parameters.
- n is the number of 2D landmark positions.
- e is the residual error.

For the localisation of the user's face region, and following the structure presented in previous chapters, two stages are distinguished: (1) the initial face detection and posterior re-detections when the tracking is lost, and (2) the in-between face tracking. This approach is relevant as tracking algorithms are typically more efficient and require less memory than those for face detection. Thus, the face detection algorithm is only activated when the user's

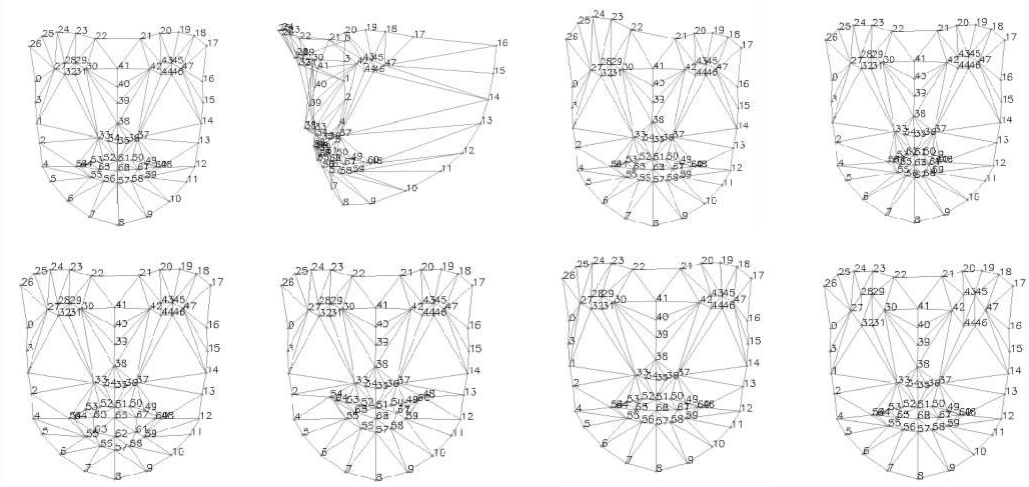


Figure 6.3: A generic deformable 3D face model and some of its deformation parameters compatible with our method.

face is not being tracked. For face detection, it uses the SSD deep neural network proposed in [157], which has shown to be robust under challenging conditions, trained specifically with multiple-pose faces. The tracking relies on CLNF [19], applied at landmark-level, which has a good balance between computational cost and localisation reliability and stability. The landmark distribution is constrained by a parametric 3D face model, to avoid impossible human facial shapes. It considers the tracking lost when the image under the face region does not correspond to a human face, according to the learned face pattern (see Algorithms 12 and 13 for further details).

Once the different facial parts are localised, it extracts the image regions around both eyes. Next, their shape and intensity distributions are normalised, so that a deep neural network, based on [152], can infer the corresponding 3D gaze vectors. Then, it calculates an overall gaze vector of the user as the weighted mean vector of both eyes with its origin at the midpoint of both eyes (see Algorithm 14).

The affine transformation matrix $\mathbf{M}$ is calculated as follows:

$$\begin{bmatrix} \alpha & \beta & (1-\alpha)\cdot c_x - \beta \cdot c_y \\ -\beta & \alpha & \beta \cdot c_x + (1-\alpha)\cdot c_y \end{bmatrix} \tag{6.2}$$

where:

Algorithm 12: Hybrid face model detection-tracking fitting algorithm.

Input: The image sequence (**I**)
Output: The face model parameters {**t**, **r**, **s**, **a**} that overlap the model to the user's face, throughout I

```
for I_j ∈ I do
    if Previous configuration b_{i-1} defined then
        Reset the face model parameters of the graphical model to the neutral configuration
        Run the face region detector in the image
        Store the detected user's face image patch and face region
    else
        Locate a stored face image patch in the image (via pattern matching)
        Verify that the located patch corresponds to a real face (via pattern classification)
        if Located face region contains a real face then
            Store the located face region
        end
    end
    if Face region available then
        Run the face landmark detector in the face region
        Adjust the 3D face model to the detected landmarks (Algorithm 13) → {t, r, s, a}
    end
end Optional
Filter {t, r, s, a} with an appropriate approach for face movements
```

- $\alpha = s * cos(\theta)$
- $\beta = s * sin(\theta)$
- $s = (w - 2 * m_1)$
- θ refers the horizontal rotation angle of the line that connects both eye corners.
- c_x, c_y are the image coordinates of the centre of rotation in the source image. Then, the source image I_{input} is transformed, which is to say, normalised in shape, using the matrix M, as follows.

$$I_{norm}^{shape}(x, y) = I_{input}(M_{11}x + M_{12}y + M_{13}, M_{21}x + M_{22}y + M_{23}) \tag{6.3}$$

Note that the applied eye shape normalisation procedure usually results in distorted images; normally, the further the user's face is from frontal view-

Algorithm 13: Three-stage face model adjustment algorithm.

Input: Set of 2D landmark positions d in the image, the relation list between the landmark and vertices and the camera parameters $\{f, w, h\}$

Output: The face model parameters $\{\mathbf{t}, \mathbf{r}, \mathbf{s}, \mathbf{a}\}$ that overlap the model to the user's face

Set the deformation parameters $\{\mathbf{s}, \mathbf{a}\}$ to zero

Convert the current parameter values to the normalised range workspace

Optimise, using for example the Levenberg-Marquardt algorithm [134, 135], Eq. (6.1) with $\{\mathbf{t}, \mathbf{r}\}$ as the only variables

Optimise, using for example the BFGS algorithm [158–161], Eq. (6.1) with $\{\mathbf{s}\}$ as the only variables

for $a_k \in \boldsymbol{a}$ **do**

| Optimise, using for example the BFGS algorithm, Eq. (6.1) with $\{a_k\}$ as the only variable

end

points, (i.e., the more distant an eye's appearance may look) , the more distorted the image becomes.

As a matter of example, Figure 6.4 shows three examples of the distortion that happens in the normalised appearance of distant eyes in non-frontal faces, when the head's yaw angle is changed. As can be observed, the green points do not match exactly the white ones because the deformability of the graphical object is not perfect. At most, e is minimised (Eq. (6.1)). Consequently, this distortion may affect the stability of the estimated gaze for different yaw rotation angles of the head. A similar instability may also happen for different pitch angles but to a lesser degree.

Thus, in order to reduce this effect, the vectors obtained in the previous step ($\{\mathbf{g}_l, \mathbf{g}_r\}^{reg}_{norm}$) are corrected by a factor that gives more importance to the dominant eye (the less distorted eye) and which are proportional to the head's pitch and yaw rotation angles, as follows:

$$\{\mathbf{g}_l, \mathbf{g}_r\}^{corrected}_{norm} = \{w_d \cdot \mathbf{g}_l, (1 - w_d) \cdot \mathbf{g}_r\}^{reg}_{norm} + \left\{ \begin{array}{c} K_y \cdot (r_y - r_{y0}) \\ K_x \cdot (r_x - r_{x0}) \\ 0 \end{array} \right\} \quad (6.4)$$

where:

- w_d is the weight of eye dominance.
- r_{x0} is the reference pitch angle.

Algorithm 14: Normalised left and right eye gaze vectors estimation algorithm.

Input: The image sequence $\mathbf{I}$, 2D left $\{\mathbf{e}_1, \mathbf{e}_2\}_l$ and right $\{\mathbf{e}_1, \mathbf{e}_2\}_r$ eye corner landmark positions, throughout $\mathbf{I}$, the adjusted face model geometry and parameters, throughout $\mathbf{I}$ and the pre-trained deep neural network for regressing 3D gazes from normalised eye images.

Output: The user's normalised left and right eye gaze vectors estimation $\{\mathbf{g}_l, \mathbf{g}_r\}_{norm}$, throughout $\mathbf{I}$

for $I_j \in \boldsymbol{I}$ **do**
- Calculate M for each eye (Eq. (6.2))
- Obtain I_{norm}^{shape} for each eye (Eq. (6.3))
- Obtain I_{norm} for each eye (via image equalisation)
- Mirror I_{norm} for the eye not corresponding to that considered by the regressor (left or right)
- Process both I_{norm} with the pre-trained deep neural network
- Un-mirror the response for the mirrored eye image ($\rightarrow \{\mathbf{g}_l, \mathbf{g}_r\}_{norm}^{reg})_j$
- Apply the dominant eye and head rotation's correction factor (Eq. (6.4)) ($\rightarrow \{\mathbf{g}_l, \mathbf{g}_r\}_{norm}^{corrected})_j$
- Divide both regression results by their corresponding Euclidean norms ($\rightarrow \{\mathbf{g}_l, \mathbf{g}_r\}_{norm})_j$

end

- r_{y0} is the reference yaw angle.
- K_x is the proportionality constant for the pitch angle.
- K_y is the proportionality constant for the yaw angle.

In the case of big out-of-plane head rotations where both eye images are too distorted to be reliable, the gaze estimation relies solely on the head direction. The values of these parameters and ranges are experimentally determined, depending on the final application. For instance, the reference pitch and yaw angles could be the average values from those observed during the image sequence, while the user's head poses are closer to frontal viewpoints. At the same time, the proportionality constants could be determined based on the observations of the gaze stability while the user is moving the head but maintaining the point of gaze. Finally, each vector is divided by the Euclidean norm to assure that the resulting vectors have a unit norm. We use the same method to obtain both normalised gaze vectors.

Remarkably, these 3D eye gaze vectors have been obtained without any previous calibration (e.g., without any initialisation procedures). This lack

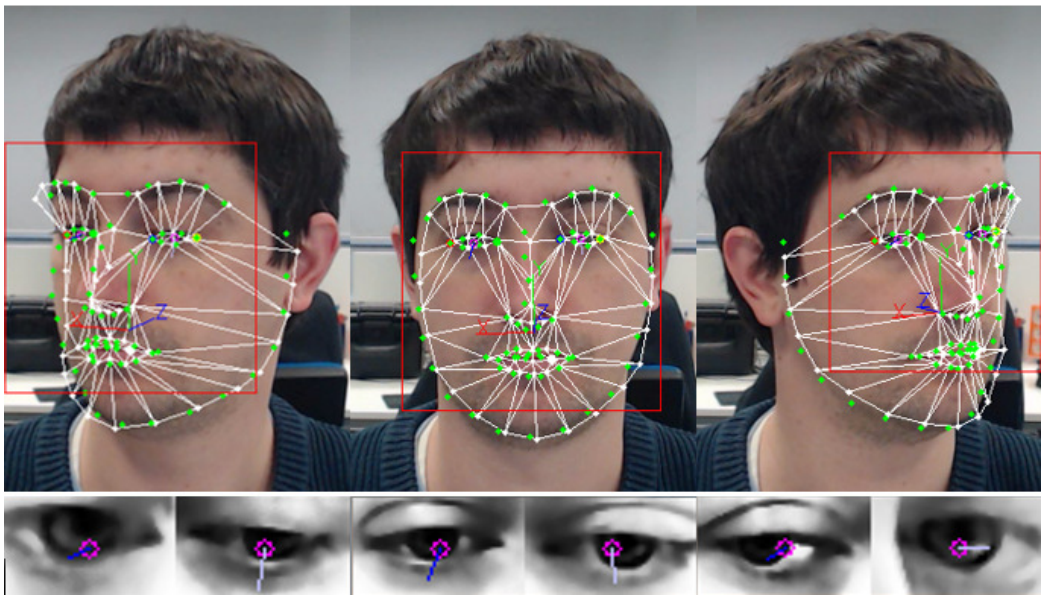


Figure 6.4: Examples of the distortion that happens in the normalised appearance of the most distant eyes in non-frontal faces, when the head's yaw angle is changed.

of calibration is especially important in applications requiring real-time monitoring of the eye gaze, such as automotive applications.

Algorithm 15 shows how the eye gaze direction is estimated with respect to the targets. First, it places the target geometries relative to the camera's coordinate system, which is the same reference used for the face and eye gaze vectors, already estimated in previous blocks. The camera's coordinate system has been previously established. In other words, it assumes that the camera's coordinate system is well-known. A target is modelled or referred to as a set of polygons formed by k points $\mathbf{b}$ and lines $\mathbf{l}$, and their corresponding planar surfaces $\{\mathbf{v}, q\}$ (where $\mathbf{v}$ is the normal vector and q the distance from the origin) that define the objects that need to be related to the user's point of gaze (e.g., a screen is represented by a rectangular plane). Then, it places the 3D face model in the scene with the resulting parameters. Next, it transforms the normalised left and right eye 3D gaze vectors, so that they refer to the coordinate system of the camera (i.e., not to the normalised camera viewpoint, as before). To this end, it removes the effect of the rotation angle θ included by the affine transformation used to normalise each eye

shape:

$$\{g_l, g_r\} = \left\{ \begin{array}{c} -cos(\theta) \cdot (\{g_l, g_r\}_{norm})_x + sin(\theta) \cdot (\{g_l, g_r\}_{norm})_y \\ -sin(\theta) \cdot (\{g_l, g_r\}_{norm})_x - cos(\theta) \cdot (\{g_l, g_r\}_{norm})_y \\ (\{g_l, g_r\}_{norm})_z \end{array} \right\} \quad (6.5)$$

Then, both gaze vectors are combined by calculating its geometric mean g, which is assumed to be the user's overall gaze vector. The gaze vector may optionally be filtered by considering its frame-to-frame motion and an appropriate filtering method for eye movements. The origin of this vector is preferably placed in the middle position (mean value) of both eye centres from the 3D face, ε. Thus, the point of gaze PoG for each target plane can be estimated:

$$\mathbf{PoG}_t = \varepsilon + \frac{(\mathbf{q} - v \cdot \varepsilon)}{v \cdot \mathbf{g}} \cdot \mathbf{g} \quad (6.6)$$

Figure 6.5: The considered zones of interest in the simulator to analyse the driver's PoG.

Finally, using a point-in-polygon strategy [162] it sees if any of the calculated PoGs lie within any of the screens. As can be observed, the point-in-polygon strategy may result in the PoG going through a polygon, or not going through any polygon. If it does not go through a polygon, the method provides the closest polygon. For example, in Algorithm 14 at line 11, if the PoG does not go through a polygon, the distance to the polygon is stored. And at line 12, the currently measured distance is compared to the minimum measured distance (which is stored), in order to guarantee that the closest polygon is finally selected.

Algorithm 15: Target-related point of gaze estimation algorithm.

Input: The set of polygons formed by k points $\mathbf{b}$ and lines $\mathbf{l}$, plane normal vectors $\mathbf{v}$ and plane distances q with respect to the camera that represent the target objects $\{\mathbf{b}_k, \mathbf{l}_k, \{\mathbf{v}, q\}\}_t$. The adjusted face model geometry and parameters $\{\mathbf{t}, \mathbf{r}, \mathbf{s}, \mathbf{a}\}$, throughout $\mathbf{I}$. The user's normalised left and right eye gaze vectors estimation $\{\mathbf{g}_l, \mathbf{g}_r\}_{norm}$, throughout $\mathbf{I}$.

Output: The user's PoG $\mathbf{PoG}_j$ with respect to the targets in the scene, throughout $\mathbf{I}$.

```
Place the target polygons with {b_k, l_k, {v, q}}_t
for I_j ∈ I do
    Place the 3D face model with {t, r, s, a}
    Transform {g_l, g_r}_norm with Eq. (6.5)
    Calculate the geometric mean vector
    (Optional) Filter with an appropriate approach for gaze movements
    BIG_NUMBER → d_t^min
    for target t do
        Calculate the point of gaze in the target's plane with Eq. (6.6) → PoG_t
        Apply a point-in-polygon strategy to the target's polygon
        if point is in polygon then
            PoG_t → PoG_j and break the loop
        else
            Store PoG_t and distance to polygon d_t
        end
        if d_t < d_t^min then
            d_t → d_t^min
            PoG_t → PoG_j
        end
    end
end
```

Table 6.1: Comparison among different state-of-the-art eye gaze estimation systems and ours.

Category	**Paper**	**Setup**	**Accuracy metrics (Mean % and/or °)**
Model-based	[147]	1 camera 1 IR illuminator 18 gaze zones considering day (no-glasses/glasses/sun-glasses) night (no-glasses/glasses) scenarios	95% on-the-road 90% off-the-road (for all scenarios)
	[151]	3 cameras (2 facing driver, 1 looking out) 6 gaze zones	94.9%
Appearance-based	[146]	1 camera 8 gaze zones	92.75%
	[150]	1 camera 6 gaze zones	94.6%
	[156]	1 camera 20 on-screen positions	10.8° (cross-dataset evaluation)
Hybrid	Ours	1 camera 7 gaze zones	97.0% / 4.6° (front screen) 87.7% / 11.5° (side screens)

6.1.2 Results

We have evaluated our approach with an experiment where eight people were recorded by a camera in front of them while using a driving simulator with three screens (Figure 6.1). The participants were requested to look at different control points located at zones of interest on the screens: (1) left window, (2) left side mirror, (3) horizon, (4) road, (5) navigation panel, (6) rear mirror and (7) right side mirror (Figure 6.5). They were free to rotate their head as needed (i.e., no instructions were given about this). We measured the accuracy of our approach in this setup without including a user calibration stage. Thus, if the PoG obtained directly, as explained above, lies within the targeted zone of interest, it is considered a correct response, otherwise it is considered wrong. In addition, we measured the angle between the vector that goes from the head to the targeted control point and from the head to the estimated PoG.

Table 6.1 shows the results, along with those obtained by other state-of-the-art model-based [147, 151] and appearance-based [146, 150, 156] alternatives with similar setups and conditions. Ideally, we would have re-implemented and adapted all these approaches to our setup so that we could measure the differences under the same working conditions. However, taking into account that many implementation details are not available in the publications, which can be important for the reproduction of the reported results, we preferred to include them here directly with their corresponding setups and accuracy metrics. In some cases, the results are in degrees between the estimated and ground-truth gaze vectors. In other cases, with a percentage of the number of times in which the eye gaze reaches the correct gaze zones. In our case, we provide both metrics so that it is easier to compare the approaches, despite the differences among the setups and conditions. Nevertheless, note that for this reason, the comparison is more qualitative than quantitative, except for the setups for their corresponding ground truth measurements.

[151] has the most different setup as it uses two cameras to capture the driver's data. Hence, it has the possibility of estimating 3D features directly and thus improves accuracy, compared to the monocular case. However, we prefer to avoid such as setup in order to simplify the installation and configuration (i.e., calibration) and reduce power consumption.

In the case of [147], it follows a similar scope to ours, using facial feature tracking, 3D head pose and gaze estimation, but with some relevant differ-

ences. The head-pose estimation algorithm relies on the 'weak-perspective' assumption, which with the kind of images obtained in this setup, produces an inherent error due to the orthographic projection that needs to be compensated. On the other hand, its proposed gaze vector estimation procedure is model-based, which has the drawbacks previously stated.

Both [146] and [150] rely on classifiers trained with the relations between gaze zones and feature descriptors composed of 2D facial part and ocular image cues. The drawback of these kinds of approaches is that, as they do not estimate 3D data, they need to be specifically trained for each setup, and provide more limited information for behaviour analysis.

[156] relies on a deep neural network to estimate the 3D gaze vector, similar to our approach, but includes both the normalised eye appearance and the head orientation as input data for the network. In this case, the authors evaluate the approach with people looking at a laptop screen, so no profile views are contemplated like those that occur in our case when users look at the side screens, and the eye appearances get distorted.

In our case, we obtain sufficient accuracy to relate rendered graphics with the user's observations, despite not having calibrated the system for each user. As expected, the accuracy is lower for the side screens, but still high enough (Figure 6.6). At any rate, these errors should be considered when designing the recognition areas for the interaction of the elements within the scene (i.e., for higher errors, the area of interaction around the element should be larger as well). Figure 6.7 shows that our approach can handle quick eye movements while maintaining a low level of noise for fixations.

To evaluate the efficiency of our approach and its suitability for running in devices which can then be used in real vehicles, we have integrated it with an app for smartphones with iOS and Android operating systems (Figure 6.8). This environment is close to a real scenario, where one cannot expect to install CPUs/GPUs like those of desktop/laptop PCs. It is worth noting that the operating system can also have an impact on the overall performance of the app. This difference is due to the multi-level structure and different programming languages in which the app needs to be programmed (i.e., the core of the approach is programmed in C++ for both operating systems, while the interface is in Objective-C for iOS and Java for Android). More specifically, we have tested the iOS app in an iPhone SE (with iOS 10.3.2) and the Android app in a Docomo smartphone (with Android 6). The measured average FPS (frames-per-second) of our app in each case was 30 and 20, respectively. These results reveal the efficiency and suitability of

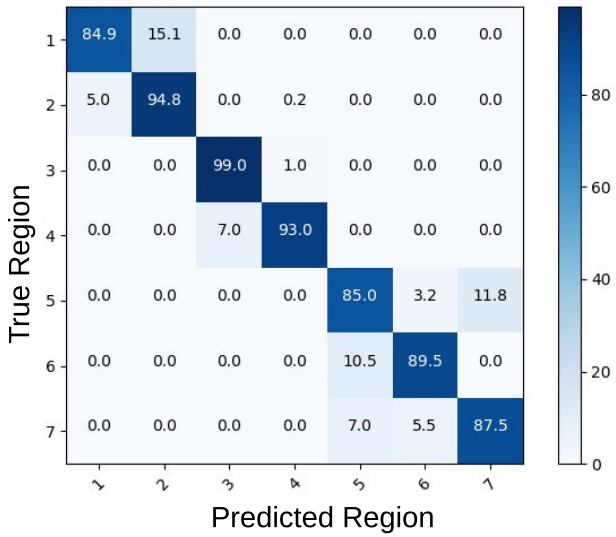


Figure 6.6: Confusion matrix of the predictions obtained by our approach for the considered gaze zones.

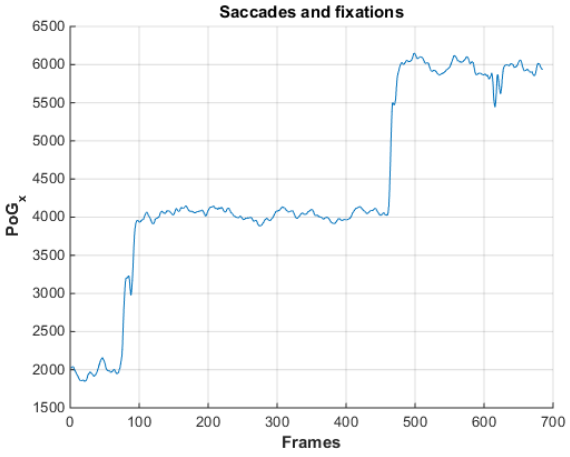


Figure 6.7: An example of PoGx signal where saccades and fixations can be appreciated, along with the level of noise.

our approach to be applied in a real-world scenario.

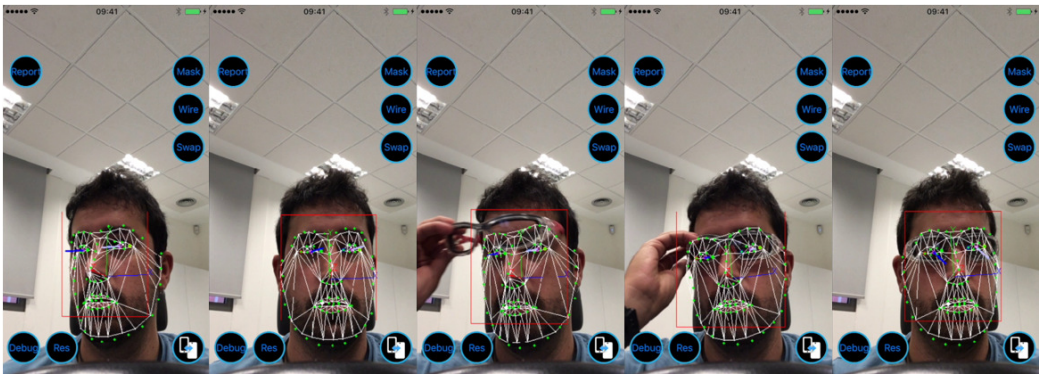


Figure 6.8: Examples of the approach running in an iPhone SE, while the user puts thick glasses on and the system keeps working.

6.1.3 Conclusions

One of the advantages of our approach is that with a simple setup, we can efficiently estimate the PoG of the user in multiple screens of a simulator, allowing us to directly relate the rendered graphics that represent the different elements of the scene with the user's observations. This way, it is easier to generate richer data for developing driving behaviour analysis approaches. Another advantage is that it can be integrated, processed and executed in devices with low computational capabilities, such as smartphones. Future work will principally focus on optimising the deep neural network designs for face detection, landmark localisation and eye gaze vector estimation stages to further improve their efficiency in ARM-based CPUs. We also plan to adapt the approach to real vehicle setups.

6.2 Sport skill analysis by 3D body pose extraction

In the last decades, cultural diversity has emerged as a primary concern, representing not only a resource to be preserved but also an asset to be promoted [163]. In the sports field, diversity is symbolised by a great number of Traditional Sports and Games (TSG), considered by UNESCO as a part of the Intangible Cultural Heritage [164].

Due to globalisation trends and the growing media influence of some sports, many TSG have already been lost or are at risk of disappearing. As stated by UNESCO, preserving information and knowledge about them is important for the generations to come. TSG should adapt to the cultural changes of the living generation, without affecting their originality.

Despite the key role of the specific sport techniques and playing skills in the game, the execution of these unique patterns of movements is often taken for granted, as part of the tradition and the experience. This information is considered essential to promote teaching and coaching at all levels, therefore guaranteeing their preservation.

To reach this goal, this section presents a new method that allows for the reconstruction of the 3D body poses of players while performing a sport-action and then evaluating the execution by comparing the techniques involved to a reference dataset, using monocular videos as input.

This approach takes advantage of the 3D human body pose tracking method presented in Chapter 5. In addition, it includes a semantic analysis procedure designed to evaluate the extracted body movements that form the technique of the performed skill.

This combined application allows us to:

- Study-specific sports techniques from legacy videos, by reconstructing the 3D motions of athletes from the past.
- Study the skills of the players from different periods by studying their techniques and comparing them in various videos.
- Monitoring the temporal evolution of techniques and playing styles of a sport by extracting data from legacy videos of different periods.
- Evaluate body motions comparing a sample motion with a golden-standard in 3D, extracting complete quantitative and qualitative performance

analysis of the athlete.

To evaluate the application of the method, we show (in section 6.2.4) some experimental results with complex sports actions involved in Pallapugno, a traditional Italian sport practised in the north-west part of the country, an area which includes some provinces of Liguria and Piedmont.

6.2.1 Data capture and analysis

In the market, we can find some dedicated software tools for motion analysis of sports videos. Popular examples are the Open Source tool Kinovea[1] and commercial tools such as ProAnalyst[2]. People with no specific technological skills can also use these software platforms because they are easy and intuitive. However, while they are able to provide accurate information, this is only possible under specific viewpoints (avoiding occlusions) and with high-quality video input, and currently, they do not provide 3D pose information from monocular images.

When the source is a legacy video, the quality of the image is often far from optimal (see legacy capture examples in Figure 6.9). Thus, it is reasonable to include a manual refinement step, were the inaccuracies introduced by the automatic estimation phase can be refined by the user to improve the accuracy of the movement analysis and the skill comparison steps.

To this end, Wei et al. [37], proposed a video-based motion modelling technique for generating physically realistic 3D human motion by monocular video sequences. Their work describes a semi-automatic process which allows for the identification of body joints starting from selected "key-frames", the contact with ground and objects. We estimate the transition between key-frames through the combination of interpolations and the correspondence between image textures.

The method exposed in section 5 revisits this kind of approach proposing a semi-automatic method divided into two main steps. First, it configures the camera by adjusting the reference floor of the pre-defined 3D world to four key-points on the image. Then it applies the body pose estimation procedure by fitting a parametrised multi-body 3D kinematic model to the 2D image body features. To reduce the users' manual interaction, the 2D

[1]Official website (10/07/2020): https://www.kinovea.org/

[2]Official website (10/07/2020): http://www.xcitex.com/proanalyst-motion-analysis-software.php

marker annotation task is almost entirely automated by a method similar to that shown by Cao et al. [25]. This method allows for the automatic detection of a set of human body joints (and other fiducial landmarks) on a monocular image, estimating their 2D location. Even if the detector configures the key-frames, the semi-automatic pipeline allows the user to correct the possible mistakes in the detection. The combination of both techniques ensures very accurate motion captures.

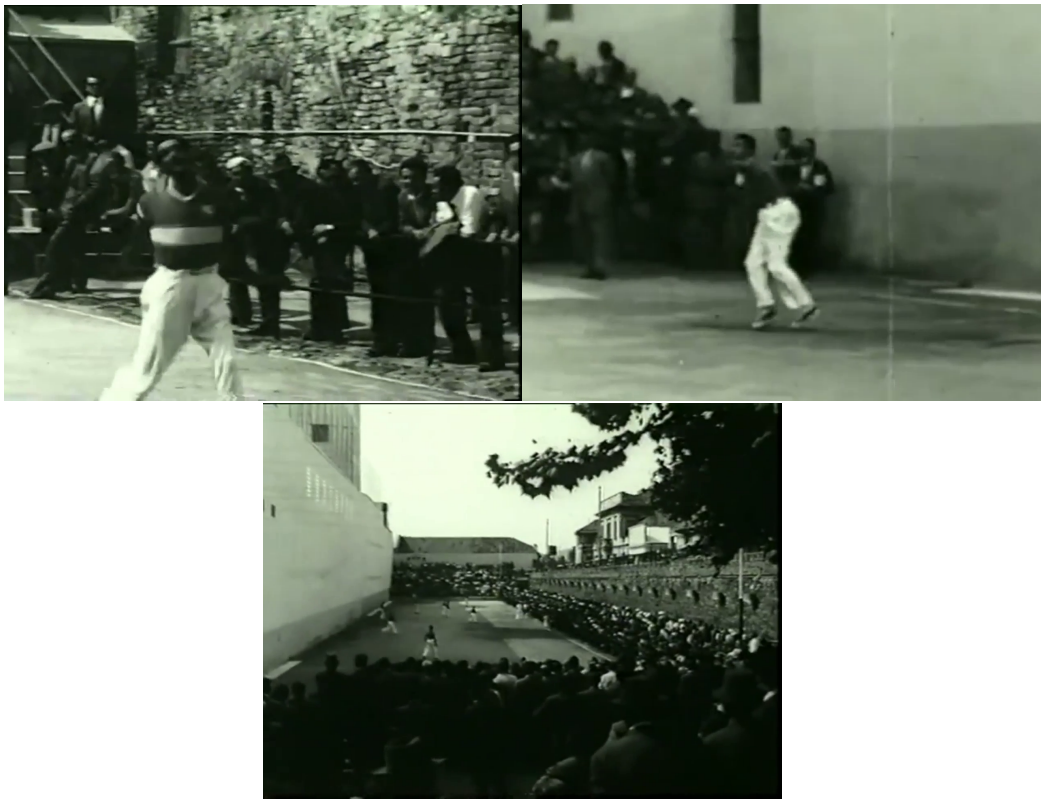


Figure 6.9: Three examples of an old broadcast video captured for TV.

Typical movement evaluation approaches rely on temporal alignment to take into account temporal inconsistencies in the compared motions, through either a dynamic time warping technique (DTW) [165] as in [166] or graph-based approaches [167]. Then, the quality of the motions is quantified against model movements typically using the distances between the measurements of the corresponding motion feature [167, 168] or the temporal alignment score [166, 169].

A common limitation related to the use of a pure scoring system is that it becomes harder to distinguish between performance levels due to the multidimensionality of the data. Therefore, in [169], a fuzzy logic approach was utilised to infer performance quality levels and offer more meaningful feedback. Furthermore, in [170], an LMA components based analysis is conducted to offer both qualitative and quantitative analysis results simultaneously.

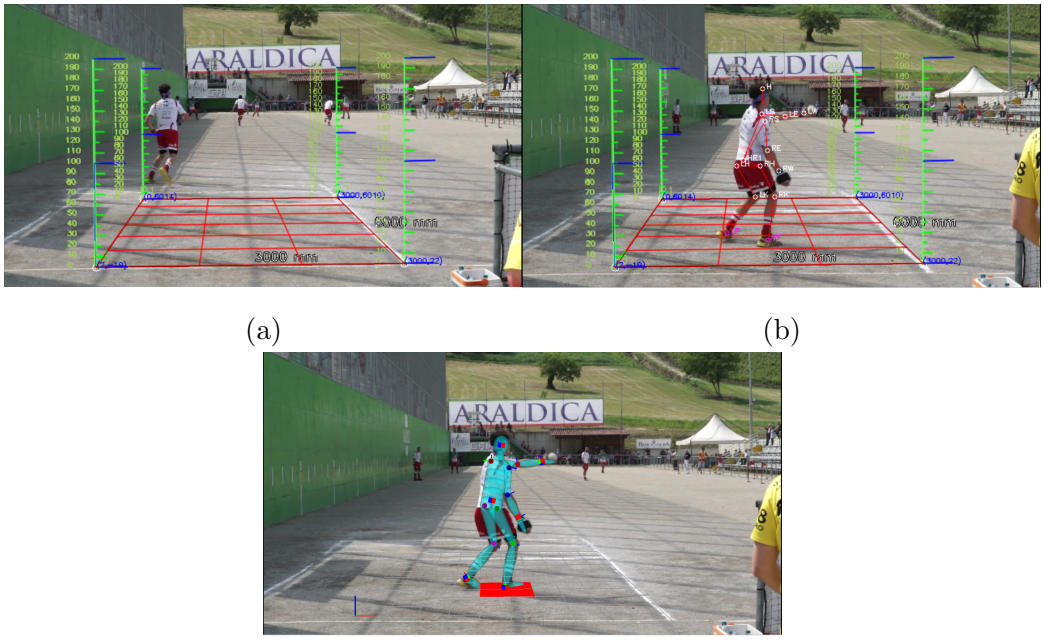
The proposed approach has adapted and extended the work by Alexiadis et al. [171],which describes a method for automatically evaluating dance performance from human motion capture data by comparing the movement of the user with the gold-standard performed by a teacher. The method aims at providing qualitative and quantitative information to assess performance through a semantic pyramid decomposition scoring scheme based on a time-series similarity measure instead of per feature distances. Ultimately, this analysis method derives the level of similarity between the two motions and offers meaningful and rich information by incorporating contextual knowledge through the form of experts' input.

6.2.2 Extraction of the athlete's 3D motion

For this method, we use a modified version of the *contextualised learning-free body adjustment* (CLFBA) approach described in section 5. This method consists of two main steps: a) a camera parameter estimation step to compute the intrinsic and extrinsic parameters of the camera, and b) a 3D body pose reconstruction using a set of reference frames (key-frames) to reconstruct the entire body motion of the video sequence. The second step includes some modifications to reduce user interaction, including an automatic 2D feature detector to avoid the manual annotation of those in the key-frames.

Estimation of the camera parameters

The camera configuration (intrinsic and extrinsic parameters) is estimated using a visible element in the scene (with a rectangular shape and in the plane of the floor, if possible) with known dimensions (commonly found in most sports, e.g., lines on the field). We named this element as the *reference element*. The user has to align a rectangular 3D graphical tool (that represents the floor plane) with the reference element (see left image in Figure 6.10).



(a) (b)

(c)

Figure 6.10: The figure shows: a) floor plane tool alignment with the reference element, b) 14 2D body joint landmark location, and c) the final pose reconstruction.

The method obtains the camera parameters by computing the homography between the image and the floor (the aligned rectangle) using a direct linear transformation [5]. We minimise the re-projection error over the set of the parameters of the camera using the Levenberg-Marquardt nonlinear optimisation method [121].

The captured reference floor must be readjusted each time the camera is moved or zoomed.

Body pose estimation

For the 3D body pose estimation, the user manually selects a set of key-frames from the video sequence. The number of key-frames depends on the length of the video sequence and the complexity of the analysed skill. Although an initial set of key-frames is selected first by the user, it can be modified during the reconstruction process if needed.

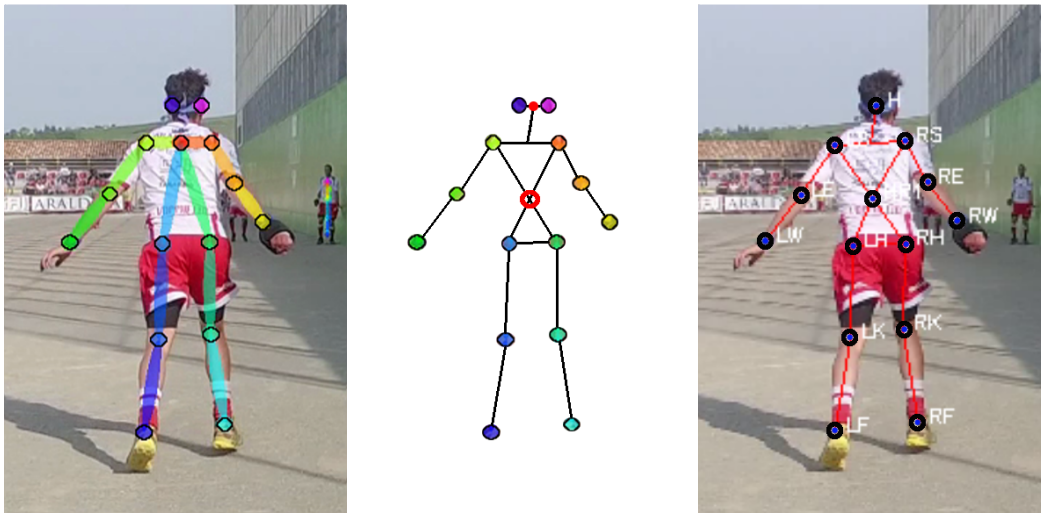
For each key-frame, a set of 14 body joints (Figure 6.11(c)) is superposed on the image using 2D markers that correspond to the main human body joints. It represents the spinal articulation as a point in the centre of the back.

The position of the 14 body joints is estimated automatically using an approach similar to [25]. This method is based on a CNN approach and can locate the entire body structure (see Figure 6.11 (a)) of multiple persons in a single image. As with regular CNN approaches, a previous training stage is needed. This stage uses a large dataset of manually annotated images to compute the proper weights of the neural network.

An iterative computation is applied to new images using the trained network, computing a list of possible joint locations and the relationship between them in each iteration. Each stage improves the result of the previous iteration, increasing accuracy, and reducing potential wrong estimations in earlier stages. As the image can represent more than one person, the detected joints must be related to those of the same body. To this end, a relation estimator encodes the location and orientation of the limbs. It determines which of the detected joints belongs to a particular person in the image.

This process gives, as a result, a list of related 2D joint positions per person in the image (see Figure 6.11 (a)). Comparing this output with the 14 joints needed for the key-frames (see Figure 6.11 (c)), it can be observed that the joint detector gives a different amount of points. The arm and leg joint points (including shoulder and hip points) match in both structures. The head base position is the middle point between the head side landmarks provided by the automatic estimation. Moreover, it defines the point representing the spinal articulation in the key-frame as the intersection of the lines that connect the left shoulder with the right hip and the right shoulder with the left hip. The diagram (Figure 6.11 (b)) shows how both point structures are related and the estimation of the missed points in a graphical way.

The automatic joint detector reduces the key-frame configuration time, but some frames could be hard to configure for the detector. In some cases, the detection fails, because of the pose, self occlusions or other non-controlled factors. Figure 6.12 shows a case where, even though similar frames are correctly estimated (the first and last frames in this sequence), there are some wrongly estimated frames. The probability of a wrongly estimated pose increases when the quality of the image decreases (expected in legacy videos). As the accuracy of the pose estimation is a key part of the movement extraction, we include a refinement step. Therefore, the user can correct the



(a) (b) (c)

Figure 6.11: Image (a) shows the estimation of the automatic joint detector for a given frame [25]. Image (c) shows the same frame with the needed 2D landmark correctly positioned for the 3D reconstruction. The image (b) shows the relation between both representations, and how the missed landmarks (in red) are computed.

wrongly estimated key-frames manually.

Once each key-frame has the 14 body joints properly located, a parametrised 3D kinematic model is automatically adjusted to the 2D joint positions in each key-frame using the CLFBA method.

For the manual refinement, the posing features located at key joints allow the pose modification of the model through IK.

An interpolation process automatically estimates the pose of the player between key-frames.

In traditional sports, a stage of action could contain a few players creating quite a large number of occlusions. Movement interpolation can reduce the impact of the occlusions in the capture process, filling the gaps between clear player images. Even if the occluded movement is an estimation, it could still be accurate if the estimated sequence is short (10-15 frames).

Figure 6.12: Five frames processed with the automatic pose detector. The pose in some of the frames is not correctly estimated.

6.2.3 3D motion analysis

The data extracted from a single video sequence can generate useful information about the skill of the player. However, this information must be interpreted by an expert to assess its performance quality. To facilitate data interpretation, the proposed method compares the motion sequence to be evaluated (called *trial* motion) with a reference motion sequence (called *reference* motion) captured from an elite and/or professional athlete performing the same sports action. This *reference* can be extracted from a video sequence specifically captured for the data comparison or could be generated from monocular TV broadcast videos (e.g., legacy videos). The system identifies their differences and generates an evaluation as feedback for the coach, who can use this information to design specific exercises for the player to improve their technique.

The methodology (see Figure 6.13) consists of three stages. Initially, we pre-process the trial motion to be make it compatible for comparison and to accommodate non-uniform and noisy representations. Then the two movements are temporally aligned to establish correspondences between them. Finally, the motion comparison follows, offering feedback to both the user and their coach.

Motion pre-processing

Initially, motions need to be re-targeted to enable their straightforward comparison. Motion re-targeting transforms the body structure of motions into a common canonical form to overcome errors due to variations in the body

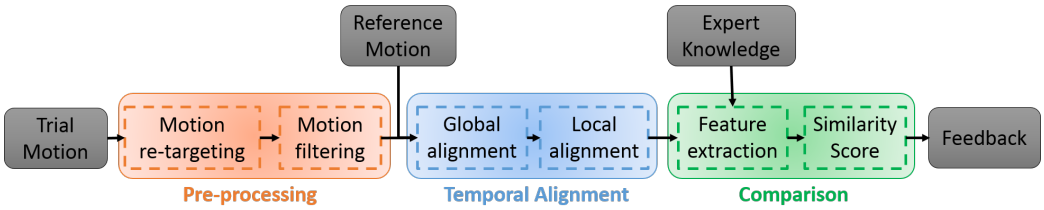


Figure 6.13: The complete end-to-end motion comparison pipeline

structure in the sub-sequent comparison step. We use an extended version of the method proposed by Ahmadi et al. [172] for the whole body. We extract motions as described in section 6.2.2, preserving the same body structure, and allowing a generic motion analysis pipeline. It enables comparisons with motions extracted through other digitisation means. In addition, body structure re-targeting further harmonises the two motions to an equally sized body, alleviating any limitations that might be related to variations of the body size and allowing Euclidean comparisons between the positions of the joints.

Then, motion filtering is applied to the canonical motions to filter out any noise introduced by the digitisation process and correct any erroneous pose estimations. As fast joint displacements typically characterise sport actions signals but at the same time may also suffer from erroneous estimations that result in sharp spikes, we use the Savitzy-Golay [173] amplitude preserving filter. As filter properties are in the time domain, it is straightforwardly differentiated. Thus, the smoothed derivative of the motion features signals can also be calculated, ultimately providing further additional features (e.g., velocity, and acceleration).

Temporal alignment

With the *trial* and *reference* motions now represented in the same canonical structure, the next step is the temporal alignment between the two motions to take into account the varying execution speed differences, spread between the different segments/phases of each action, coordination disparity, and starting/ending offsets. The approach described in [171] is used with some modifications.

First, the joint positions are pure quaternions, and then a global temporal alignment is achieved via quaternionic cross-covariance. We calculate the global shift between the two motions extracting the cross-covariance max-

imum. Contrary to [171], we use forward quaternionic cross-covariance instead of the circular one used in the dance scenario, as TSG actions do not typically include periodic motion patterns.

The next step involves local temporal alignment, driven by Dynamic Time Warping [165], operating on the pure quaternions. However, we use the highest kinetic energy limb instead of allowing all the joint positions to contribute, as typically, sports actions are performed by a primary, single limb. This approach, in turn, produces a set of dense correspondences between the two - now aligned - motion sequences.

Comparison

As the final feedback focuses on a coaching scenario, we seek to provide richer semantic information to drive the progression of the user through coaching, avoiding complicated low-level information (joint positions, joint velocities, 3D-flow based scores, etc.). Thus, the use of anthropometric features (e.g., joint flexion, extension, adduction, abduction, speed, and acceleration), has been selected as an easier way to communicate between the coach and user.

This semantic interpretation requires an expert's (e.g., sports analyst, coach) knowledge as input through a pre-defined set of motion features and their relative weighting. We use this input to drive the analysis of the sport technique, effectively infusing it with the expert's experience.

Moreover, semantic information is further complemented by segregating each action into phases and splitting the feature selection and weights among them. While phase extraction is manually annotated for the *reference* motion, the *trial* motion is automatically segmented through the correspondences established between both motion sequences during the alignment step (section 6.2.3). The frames between those phases denote important key-frames for the motion extraction step. Overall, the evaluation scheme is oriented towards mapping the expertise of the coach into the motion analysis methodology and offer meaningful evaluations in order to stimulate efficient training and coaching.

We extended the scoring process by utilising a similarity metric for comparing the motion features and obtaining a score in time. The Structured Similarity Index Metric (SSIM) [174] is adopted from the image quality analysis field and used for the calculation of the motion feature time-series similarity. In the one-dimensional formulation, the SSIM is the weighted combination of: an amplitude term, scoring the average value of a set of motion

features; a contrast term, scoring the variance of a motion feature; and a temporal term, scoring the correlation of a motion feature in time.

We calculate SSIM around temporal neighbourhoods of each motion feature, and their average is used to calculate the aggregated score. Finally, after calculating the score of each feature, the overall motion performance is depicted by the weighted average of all the selected motion features.

6.2.4 Experimental results with Pallapugno videos

We demonstrate the complete motion analysis pipeline by extracting and comparing two Pallapugno serves performed by an acknowledged practitioner. They were recorded at different times during two Pallapugno games, from similar challenging viewpoints behind the performer that introduce severe self-occlusions. They are both very fast actions for a common video capture device, with significant motion blur lasting 174 and 129 frames. We have empirically selected the longer movement sequence as the *reference* motion and the shorter as the *trial* motion, and annotate the phases for the *reference* motion.

First, the complete motion sequence is extracted (Figure 6.14). Then, we define and annotate the phases of the sports action, setting the key-frames as a frame in the transitions between them. We identify four time-points: start of the back-swing; the start of the front-swing (coinciding with the end of the back-swing); ball impact; and after-swing relaxation. Performance evaluation for each phase is accomplished via the weighted similarity contributions of M selected features $\mathcal{F} = \{\mathbf{F}_i \mid \mathbf{F}_1, \ldots, \mathbf{F}_M\}$ and is thus measured by the phase similarity score $S_i = \sum_{i=1}^{M} w_i \mathbf{F}_i$, where $\mathbf{F}_i$ denotes the measurement time series of the selected feature for the temporal segment of that phase. In this way, we achieve a semantic pyramidal decomposition scheme, which better communicates the shortcomings of each performance. First, we identify in which particular phase the user's technique fails. Then, we generate precise guidelines for improvement by pinpointing the lowest scoring features (i.e., *the right elbow needs to further extend during the back-swing*).

Table 6.2 presents a set of expert-defined (coach) weights that we use for the overall scoring. It identifies the respective skill phase they belong to, the motion feature of the joint and type, as well as the overall weighting of each phase.

Figure 6.15 illustrates the right shoulder and elbow extension features respectively for all phases of the reference and trial actions, as well as their

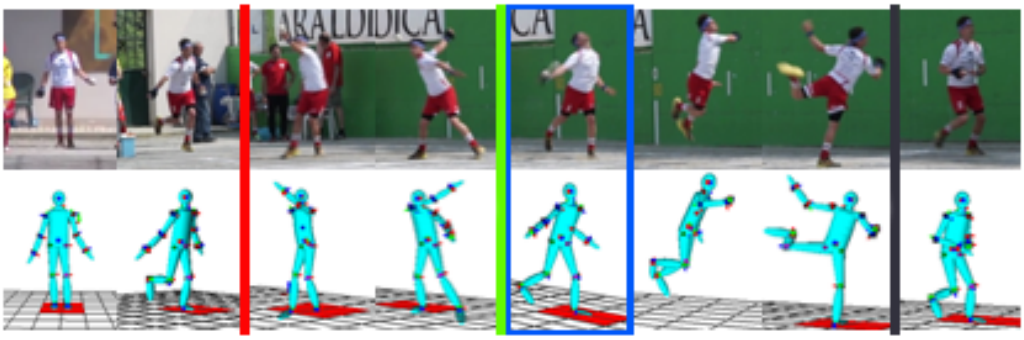


Figure 6.14: The image shows the key-frames of the 3D reconstruction of Pallapugno serve skill. The coloured lines represent the key-frame of: the start of back-swing (red), the back-swing to front-swing transition (green), the player-ball impact (blue), and the front-swing to follow-through transition (grey).

Phase	Joint	Feature	Weight	Phase Sum
Back-swing	Right Shoulder	Extension	16.00%	
Back-swing	Right Elbow	Extension	16.00%	**40%**
Back-swing	Right Knee	Flexion	8%	
Front-swing	Right Shoulder	Extension	12.00%	
Front-swing	Right Elbow	Extension	12.00%	
Front-swing	Right Wrist	Linear Velocity	8.00%	**40%**
Front-swing	Torso	Angular Velocity	6.00%	
Front-swing	Right Knee	Flexion	2.00%	
Follow through	Right Wrist	Linear Velocity	13.00%	
Follow through	Torso	Angular Velocity	2.00%	**20%**
Follow through	Right Elbow	Extension	2.00%	
Follow through	Right Shoulder	Extension	3.00%	

Table 6.2: Weights allocated to the motion features of each joint in the three different phases of the Pallapugno serve

similarity in time. While the captured motions are highly similar as they are both samples of the same action performed by the same expert, a discrepancy is apparent at the elbow's follow-through phase, during the right arm's relaxation after ball impact. The analysis scheme mentioned above will help in identifying this issue. For the analysed action, the follow-through phase

scores the lowest compared to its expected contribution. In numbers, it accumulates a score of 14.58% (the score being an extension of the weighted sum of the feature similarities for that phase to a percentage). This value has a ratio of 0.729 compared to the overall 20% nominal score as presented in Table 6.2 for the follow-through phase. At the same time, the ratios for the other two phases are 0.788 and 0.888 for the back-swing and front-swing respectively.

The feedback will guide the expert/coach through a decomposition approach to further investigate the follow-through phase and then identify the differences in technique. While the performance deviation can also be found in the follow-through phase for the shoulder extension (with the back-swing and front-swing phases also highly similar), there is a larger deviation for the right elbow extension feature. We defined the weights used for these terms experimentally. They were specifically 0.5 for the amplitude, 0.4 for the distribution and 0.1 for the structural.

6.2.5 Conclusions

We have presented a method to reconstruct and analyse the body movements of a sports player in monocular video sequences while performing a sports action, generating a semantic evaluation as feedback. TSG coaches and institutions can use this method to analyse the evolution of the sport and the skills of present and past players.

We validated the method in the assessment of the relative performance of two Pallapugno Serve skills performed by the same professional athlete in two different matches and courts.

Furthermore, we presented the improvement and acceleration of a process for the 3D pose estimation from monocular image. To this end, we established the automation of manual key-framing by employing an automatic detection method based on deep neural networks, significantly reducing the time and effort required in the process.

The entire methodology is intented to aid in the preservation and promotion of TSG, where lower-cost solutions are a key element in ensuring their eventual use. The proposed methodology can contribute to this end by also involving coaches in a multitude of ways which, during training sessions with players, will enable them to document their performance and progression through time.

A lot of technical challenges associated with performance evaluation, but

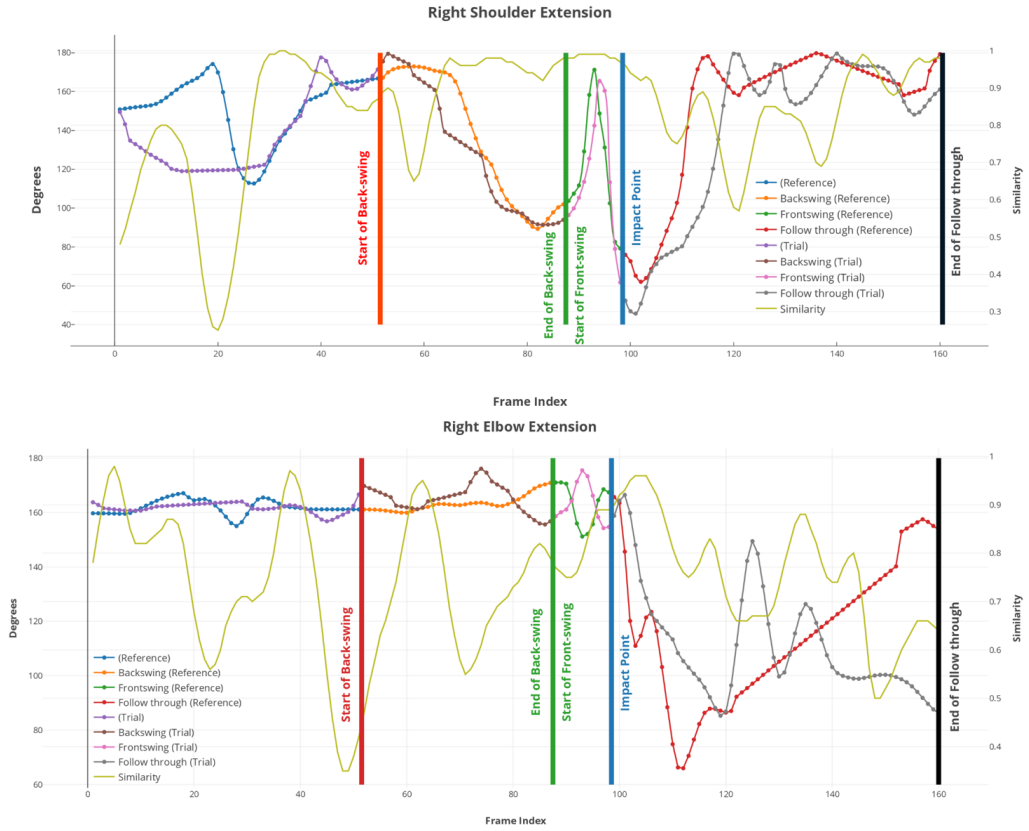


Figure 6.15: Comparison of the right shoulder (up) and right elbow (down) extension features for two Pallapugno serve actions. The feature's measurements over time as well as their similarity are presented. The vertical lines denote specific key-frames that mark the phase transitions as in Figure 6.14

.

also issues like objectivity versus personal intuition, had to be addressed jointly in a unified framework. To that end, a compromise between numerical analysis and coach input was found by allowing the coaches to influence the automatic scoring procedure and guide it through their expert-defined weighting schemes.

Chapter 7

Conclusions and future work

This section summarises the main conclusions from work carried out during the research process. In addition, it suggests some steps to follow in the future based on the results of the proposed methods.

7.1 Conclusions

This study presented various methods to track deformable three-dimensional objects on monocular RGB camera captures. For the final application context, it assumes hardware environments with limited computational power like onboard or embedded devices. Thus, the proposed methods focus on computational efficiency.

For the tracking process, this study considered two types of deformable 3D objects: objects with non-rigid deformations, and objects with rigid multi-body deformations. These two types of deformation represent most deformable objects, so the proposed methods are widely generalizable. To present an example of each type of deformation and guided by the main motivations of this research, it proposed a method for tracking human faces as a non-rigid deformable object and the human body pose as a rigid multi-body object.

Additionally, this study also proposed a tracking pipeline divided into two parts: the detection of feature points of the object in the image, and the adjustment of a pre-designed 3D deformable object using the detected feature points. Regarding the first part of this pipeline, we proposed two efficient methods to extract feature points from monocular images. For the second

part, we proposed two adjustment methods to fit generic 3D models (i.e., with non-rigid or rigid multi-body deformations) to the image representation of the object using the described feature points.

This research proposed two different techniques for the estimation of two-dimensional feature points in the image, using the facial feature point detection as a representative example. The first uses a gradient analysis in predefined sub-zones within the bounding box of the face in the image. This analysis turned out to be computationally very efficient while revealing the location of the feature points of the area under analysis. Performance tests using an iPad2, first released on March 11, 2011, demonstrated the method's efficiency in hardware with few resources.

The second technique is a regression process divided into various statistical models based on deep learning algorithms. It distributes the regression models into two processing layers. The first level extracts general face attributes (i.e., some landmarks and face orientation), and the second layer extracts zone-specific attributes like gestures, the eye gaze and specific feature points of each face region (i.e., eyes, eyebrows, and mouth). The proposed deep learning models combine multiple input data such as image and facial orientation value and generate multiple outputs using a multi-task processing strategy. Despite the large amount of information that it can extract from an image, experiments showed that the proposed method is computationally more efficient than other similar methods, improving even the precision of the estimation.

For the transition between the two-dimensional face landmark scope to the three-dimensional face tracking, this study proposed a method to fit a deformable 3D model to the user's face using the detected 2D feature points. As part of the process, this method adjusts the rigid parameters (i.e., position and orientation) of the face model in the 3D environment in front of the camera using a full projection scheme. This type of projection allows for the estimation of the 3D face position in the 3D space in front of the capturing camera. Thus, it is possible to identify the interactions of the tracked face with other elements of the scene. Moreover, it adjusts the shape of the three-dimensional model to the user's facial shape, as well as its gesture at each moment of the tracking, enabling the use of this information for posterior behaviour analysis. The experiments demonstrated the computational efficiency of the method. At the same time, the precision of the estimation is very similar to other methods proposed in the scientific literature, which present higher computational needs.

For the fitting of the body pose, this work proposed a method based on a generic deformable 3D model of the body, which includes a series of articular constraints. Similar to the face fitting method, it relies on a series of two-dimensional feature points (in this case, the most representative joint locations) to fit the three-dimensional model using a full-projection scheme. Experiments showed that the proposed method correctly extracts both the body pose and the estimation of the lengths of the different body elements without any previous training phase.

To test the validity of the proposed methods, and the presented tracking pipeline, 3D monitoring methods were tested in two final application environments.

We integrated the facial tracking method as part of an onboard driver monitoring system. The driver monitoring system uses the tracking information to get the users location in the 3D scene. Then, a second processing phase identified the driver's eye-gaze vector and presents it in a three-dimensional environment along with the position of various elements of the car or a driving simulator, defining the point-of-gaze of the driver. This information is crucial to determine what the user's focus of attention is, and their alertness. The experiments demonstrated the validity of the extracted data, being an accurate representation of the position and orientation of the user with respect to the camera.

The human body pose extraction method was also integrated into an application to estimate the athlete's performance during the practice of a sport. In this context, the system extracts the position and pose data of the athletes' body during sports practice and uses the data for subsequent analysis of the athlete's performance. This type of analysis helps instructors to improve the performance of athletes, making it possible to compare the technique of an athlete with previous captures or with captures from other players of reference. Experiments demonstrated the method's ability to extract valuable information when analyzing such activities.

7.2 Future work

This section outlines various avenues for further development and research.

For the task of detecting bidimensional feature points, it would be interesting to add more gestures to the detection. Gestures that include the opening of the mouth can be particularly confusing, since depending on whether

the jaw moves or not, the gesture could be interpreted as "open the mouth", or "separate the lips". This disambiguation is especially important in applications that analyze a user's speech or read lips through images. Furthermore, with the inclusion of asymmetric gestures, it could have direct medical applications such as rehabilitation analysis in cases of facial paralysis.

It is also necessary to include a full estimate of head orientation (yaw, pitch and roll) to improve the fitting. This information can be used for initialization of the 3D model in the subsequent adjustment phase, reducing possible initialization errors. The same strategy could be used with the detected gestures, which can be part of the 3D adjustment initialization, increasing the initial similarity between users' gesture and model configuration, reducing the possible local minima in the parameter optimization phase.

This study presented the detection of feature facial points as an example in the 2D extraction part. However, a similar system could improve the estimation of joints in the context of body adjustment. Thus, a first phase could estimate the general orientation of the body, and a second phase could use this orientation to locate the feature points of the limbs. In addition, an estimation of the gesture (global or local) could use possible body configurations depending on the context of the capture. This constraint should improve the estimation of body landmarks. For example, in the context of sports practice, gestures or actions related to sport (e.g., running, jumping, defending, taking off) could be taken as a basis to reduce the space of possible body positions.

In the same way, this additional information could be used for the initialization of the three-dimensional model, reducing the computation and thus the computational load.

For its part, the context of DNN-based regression methods provides particular possibilities for performance improvement. Depending on the hardware to be used, techniques such as model binarization, or weight optimization within the model can significantly reduce the computational load without sacrificing too much estimation precision. Combining these strategies with specific acceleration hardware such as the one presented in section 2.4 can significantly improve the final performance of the tuning process.

7.3 Relevant publications

During the research process, we presented a series of observable contributions to the scientific community in a series of publications in international conferences and journals, also including a patent application. Below is the list with all these contributions:

7.3.1 Journals

- Jon Goenetxea, Luis Unzueta, Fadi Dornaika, and Oihana Otaegui, "Efficient deformable 3D face model tracking with limited hardware resources," Multimedia Tools and Applications, vol. 79, pp. 12373-12400, 2020. JCR Impact Factor 2019: 2.313; Q2-34/108-"Computer Science, Software Engineering - SCIE"

- Luis Unzueta, Nerea Aranjuelo, Jon Goenetxea, Mikel Rodriguez and Maria Teresa Linaza, "Contextualised Learning-Free Three-Dimensional Body Pose Estimation from Two-Dimensional Body Features in Monocular Images," IET Computer Vision, vol. 10(4), pp. 299-307, 2016. JCR Impact Factor 2016: 0.878; Q4-0104/0133-"Computer Science, Artificial Intelligence - SCIE"

- Luis Unzueta, Waldir Pimenta, Jon Goenetxea, Luis Paulo Santos and Fadi Dornaika, "Efficient Generic Face Model Fitting to Images and Videos," Image and Vision Computing, vol. 32(5), pp. 321-334, 2014. JCR Impact Factor 2014: 1.587; Q1-0022/0104-"Computer Science, Software Engineering - SCIE"

7.3.2 Books and book chapters

- Luis Unzueta, Waldir Pimenta, Jon Goenetxea, Luis Paulo Santos and Fadi Dornaika, "Efficient Deformable 3D Face Model Fitting to Monocular Images," in Advances in Face Image Analysis: Theory and Applications, Fadi Dornaika, Ed. United Arab Emirates: Bentham Science Publishers, 2016, pp. 154-180. ISBN 978-1-68108-111-3. Indexed in EBSCO.

- Jon Goenetxea, Luis Unzueta, Maria Teresa Linaza, Mikel Rodriguez, Noel E. O'Connor and Kieran Moran, "Capturing the Sporting Heroes

of Our Past by Extracting 3D Movements from Legacy Video Content," in Digital Heritage: Progress in Cultural Heritage: Documentation, Preservation, and Protection. EuroMed 2014. Lecture Notes in Computer Science, vol. 8740, Marinos Ioannides, Nadia Magnenat-Thalmann, Eleanor Fink, Roko Zarnic, Alex-Yianing Yen and Ewald Quak, Eds. Switzerland: Springer International Publishing AG, 2014, pp. 48-58. ISBN 978-3-319-13694-3.

7.3.3 Conferences, congresses and workshops

- Jon Goenetxea, Luis Unzueta, Unai Elordi, Juan Diego Ortega and Oihana Otaegui, "Efficient Monocular Point-of-Gaze Estimation on Multiple Screens and 3D Face Tracking for Driver Behaviour Analysis," in Proceedings of the International Conference on Driver Distraction and Inattention, 2018 (online).

- Alejandro Clemotte, Harbil Arregui, Miguel A. Velasco, Luis Unzueta, Jon Goenetxea, Unai Elordi, Eduardo Rocon, Ramon Ceres, Javier Bengoechea, Iosu Arizkuren and Eduardo Jauregui, "Trajectory Clustering for the Classification of Eye-Tracking Users with Motor Disorders," in Proceedings of Jornadas de Automática, Madrid, 2016. ISBN: 978-84-617-4298-1. Awarded the "Best Bioengineering Work."

- Francois Destelle, Amin Ahmadi, Kieran Moran, Noel E. O'Connor, Nikolaos Zioulis, Anargyros Chatzitofis, Dimitrios Zarpalas, Petros Daras, Luis Unzueta, Jon Goenetxea, Mikel Rodriguez, Maria Teresa Linaza, Yvain Tisserand and Nadia Magnenat-Thalmann, "A Multi-Modal 3D Capturing Platform for Learning and Preservation of Traditional Sports and Games," in Proceedings of the ACM International Conference on Multimedia (ACMMM), Brisbane, Australia, 2015, pp. 747-748. Class 1 Conference in GII-GRIN-SCIE (GGS) 2017.

- Luis Unzueta, Jon Goenetxea, Mikel Rodriguez and Maria Teresa Linaza, "Viewpoint-dependent 3D Human Body Posing for Sports Legacy Recovery from Images and Video," in Proceedings of the European Signal Processing Conference (EUSIPCO), Lisbon, Portugal, 2014, pp. 361-365. SJR Indicator 2014: 0.253; 0055/0258-"Computer Science; Signal Processing"

- Noel E. O'Connor, Yvain Tisserand, Anargyros Chatzitofis, François Destelle, Jon Goenetxea, Luis Unzueta, Dimitrios Zarpalas, Petros Daras, Maria Teresa Linaza, Kieran Moran and Nadia Magnenat-Thalmann, "Interactive Games for Preservation and Promotion of Sporting Movements," in Proceedings of the European Signal Processing Conference (EUSIPCO), Lisbon, Portugal, 2014, pp. 351-355. SJR Indicator 2014: 0.253; 0055/0258-"Computer Science; Signal Processing"

7.3.4 Patent applications

- Luis Unzueta, Jon Goenetxea, Unai Elordi and Oihana Otaegui, "Method, system and computer program product for eye gaze direction estimation". EP3506149-A1, 2019. [Status: Pending].

Bibliography

[1] N. Kourkoumelis and M. Tzaphlidou, "Eye safety related to near infrared radiation exposure to biometric devices," *TheScientificWorldJournal*, vol. 11, pp. 520–8, 03 2011.

[2] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand Keypoint Detection in Single Images using Multiview Bootstrapping," *Cvpr*, 2017.

[3] F. Wu, L. Bao, Y. Chen, Y. Ling, Y. Song, S. Li, K. N. Ngan, and W. Liu, "MVF-Net: Multi-view 3D face morphable model regression," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 959–968, 2019.

[4] J. Liang and M. C. Lin, "Shape-Aware Human Pose and Shape Reconstruction Using Multi-View Images," in *International Conference on Computer Vision (ICCV)*, pp. 4352–4362, 2019.

[5] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[6] T. Baltrusaitis, P. Robinson, and L. P. Morency, "OpenFace: An open source facial behavior analysis toolkit," *2016 IEEE Winter Conference on Applications of Computer Vision, WACV 2016*, 2016.

[7] P. Huber, G. Hu, R. Tena, P. Mortazavian, W. P. Koppen, W. J. Christmas, M. Rätsch, and J. Kittler, "A multiresolution 3d morphable face model and fitting framework," in *Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, pp. 79–86, 2016.

[8] A. Bulat and G. Tzimiropoulos, "How far are we from solving the 2D & 3D Face Alignment problem? (and a dataset of 230,000 3D facial landmarks)," in *IEEE International Conference on Computer Vision (ICCV)*, (Venice, Italy), IEEE, 2017.

[9] J. Deng, J. Guo, Y. Zhou, J. Yu, I. Kotsia, and S. Zafeiriou, "RetinaFace: Single-stage Dense Face Localisation in the Wild," *ArXiv*, 2019.

[10] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M.-h. Shafiei, H.-p. Seidel, D. Casas, C. Theobalt, M. Shafiei, H.-P. Seidel, and W. Xu, "VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–13, 2017.

[11] B. Ko, "A Brief Review of Facial Emotion Recognition Based on Visual Information," *Sensors*, vol. 18, no. 2, p. 401, 2018.

[12] J. Thies, M. Zollhöfer, M. Nießner, L. Valgaerts, M. Stamminger, and C. Theobalt, "Real-time expression transfer for facial reenactment," *ACM Transactions on Graphics*, vol. 34, no. 6, pp. 1–14, 2015.

[13] T. Mittal, P. Guhan, U. Bhattacharya, R. Chandra, A. Bera, and D. Manocha, "Emoticon: Context-aware multimodal emotion recognition using frege's principle," *ArXiv*, vol. abs/2003.06692, 2020.

[14] D. Aneja, B. Chaudhuri, A. Colburn, G. Faigin, L. Shapiro, and B. Mones, "Learning to generate 3d stylized character expressions from humans," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 160–169, March 2018.

[15] N. Almohaimeed and M. Prince, "A Comparative Study of different Oject Tracking Methods in a Video," *International Journal of Computer Applications*, vol. 181, no. 41, pp. 1–8, 2019.

[16] M. Zollhöfer, J. Thies, P. Garrido, D. Bradley, T. Beeler, P. Pérez, M. Stamminger, M. Nießner, and C. Theobalt, "State of the art on monocular 3D face reconstruction, tracking, and applications," *Computer Graphics Forum*, vol. 37, no. 2, pp. 523–550, 2018.

[17] Y. Chen, Y. Tian, and M. He, "Monocular human pose estimation: A survey of deep learning-based methods," *Computer Vision and Image Understanding*, vol. 192, no. January, p. 102897, 2020.

[18] V. Kazemi and S. Josephine, "One Millisecond Face Alignment with an Ensemble of Regression Trees," *Computer Vision and Pattern Recognition (CVPR)*, 2014.

[19] T. Baltrušaitis, P. Robinson, and L. P. Morency, "Constrained local neural fields for robust facial landmark detection in the wild," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013.

[20] D. Tome, C. Russell, and L. Agapito, "Lifting from the deep: Convolutional 3D pose estimation from a single image," in *Conference on Computer Vision and Pattern Recognition, CVPR*, vol. 2017-Janua, pp. 5689–5698, IEEE, 2017.

[21] E. Richardson, M. Sela, and R. Kimmel, "3d face reconstruction by learning from synthetic data," in *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 460–469, 2016.

[22] A. Tewari, M. Zollöfer, H. Kim, P. Garrido, F. Bernard, P. Perez, and T. Christian, "MoFA: Model-based Deep Convolutional Face Autoencoder for Unsupervised Monocular Reconstruction," in *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[23] G. Pavlakos, L. Zhu, X. Zhou, and K. Daniilidis, "Learning to Estimate 3D Human Pose and Shape from a Single Color Image," in *Conference on Computer Vision and Pattern Recognition*, IEEE, 2018.

[24] C. Cao, H. Wu, Y. Weng, T. Shao, and K. Zhou, "Real-time facial animation with image-based dynamic avatars," *ACM Transactions on Graphics*, vol. 35, no. 4, pp. 1–12, 2016.

[25] Z. Cao, T. Simon, S. E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *Conference on Computer Vision and Pattern Recognition, CVPR*, vol. 2017-Janua, pp. 1302–1310, IEEE, 2017.

[26] L. Unzueta, W. Pimenta, J. Goenetxea, L. P. Santos, and F. Dornaika, "Efficient generic face model fitting to images and videos," *Image and Vision Computing*, vol. 32, no. 5, pp. 321–334, 2014.

[27] L. Unzueta, J. Goenetxea, M. Rodriguez, and M. Linaza, "Viewpoint-dependent 3D human body posing for sports legacy recovery from images and video," in *European Signal Processing Conference*, 2014.

[28] J. Goenetxea, L. Unzueta, U. Elordi, J. D. Ortega, and O. Otaegui, "Efficient monocular point-of-gaze estimation on multiple screens and 3D face tracking for driver behaviour analysis," in *Proceedings of the 6th Driver Distraction and Inattention conference*, (Gothenburg, Sweden), pp. 118–125, Sweden MEETX AB, 2018.

[29] J. Goenetxea, L. Unzueta, F. Dornaika, and O. Otaegui, "Efficient deformable 3D face model tracking with limited hardware resources," *Multimedia Tools and Applications*, 2020.

[30] J. Goenetxea, M. Linaza, M. Rodriguez, and L. Unzueta, "Viewpoint-dependent 3D Human Body Posing for Sports Legacy Recovery From Images and Video," in *Digital Heritage. Progress in Cultural Heritage: Documentation, Preservation, and Protection*, Springer International Publishing, 2014.

[31] J. Goenetxea, L. Unzueta, M. T. Linaza, M. Rodriguez, N. E. O\textquoterightConnor, and K. Moran, "Capturing the sporting heroes of our past by extracting 3D movements from legacy video content," in *Digital Heritage. Progress in Cultural Heritage: Documentation, Preservation, and Protection*, (Cham), pp. 48–58, Springer International Publishing, 2014.

[32] F. Destelle, A. Ahmadi, K. Moran, N. O'Connor, N. Zioulis, A. Chatzitofis, D. Zarpalas, P. Daras, L. Unzueta, J. Goenetxea, M. Rodriguez, M. Linaza, Y. Tisserand, and N. Thalmann, "A Multi-Modal 3D capturing platform for learning and preservation of traditional sports and games," in *MM 2015 - Proceedings of the 2015 ACM Multimedia Conference*, 2015.

[33] L. Unzueta, N. Aranjuelo, J. Goenetxea, M. T. Linaza, and M. Rodriguez, "Contextualised learning-free three-dimensional body pose es-

timation from two-dimensional body features in monocular images," *IET Computer Vision*, vol. 10, no. 4, pp. 299–307, 2016.

[34] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, pp. 681–685, June 2001.

[35] G. Fyffe, A. Jones, O. Alexander, R. Ichikari, and P. Debevec, "Driving high-resolution facial scans with video performance capture," *ACM Transactions on Graphics*, vol. 34, no. 1, pp. 1–13, 2014.

[36] X. Zhou, X. Sun, W. Zhang, S. Liang, and Y. Wei, "Deep kinematic pose regression," in *Computer Vision–ECCV Workshops*, vol. 9915 LNCS, pp. 186–201, Springer, 2016.

[37] X. Wei and J. Chai, "Videomocap: Modeling physically realistic human motion from monocular video sequences," in *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, (New York, NY, USA), Association for Computing Machinery, 2010.

[38] A. Bulat and G. Tzimiropoulos, "Binarized Convolutional Landmark Localizers for Human Pose Estimation and Face Alignment with Limited Resources," in *IEEE International Conference on Computer Vision (ICCV)*, (Venice, Italy), IEEE, 2017.

[39] A. Agarwal and B. Triggs, "Recovering 3d human pose from monocular images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 44–58, January 2006.

[40] S. X. Ju, M. J. Black, and Y. Yacoob, "Cardboard people: a parameterized model of articulated image motion," in *International Conference on Automatic Face and Gesture Recognition*, pp. 38–44, 1996.

[41] M. Andriluka, S. Roth, and B. Schiele, "Pictorial structures revisited: People detection and articulated pose estimation," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1014–1021, 2009.

[42] L. Sigal, M. Isard, H. Haussecker, and M. J. Black, "Loose-limbed people: Estimating 3D human pose and motion using non-parametric belief propagation," in *International Journal of Computer Vision*, vol. 98, pp. 15–48, 2012.

[43] 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, no. 6, pp. 248:1—-248:16, 2015.

[44] S. Zuffi and M. J. Black, "The stitched puppet: A graphical model of 3d human shape and pose," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3537–3546, IEEE, 2015.

[45] X. Jin and X. Tan, "Face alignment in-the-wild: A Survey," *Computer Vision and Image Understanding*, vol. 162, pp. 1–22, 2017.

[46] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker, "Multi-pie," *Image Vision Comput.*, vol. 28, p. 807–813, May 2010.

[47] C. Sagonas, E. Antonakos, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 faces In-the-wild challenge: Database and results," *Image and Vision Computing*, vol. 47, pp. 3–18, 2016.

[48] K. Messer, J. Matas, J. Kittler, J. Luettin, and G. Maitre, "Xm2vtsdb: The extended m2vts database," in *Second international conference on audio and video-based biometric person authentication*, vol. 964, pp. 965–966, 1999.

[49] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, Jin Chang, K. Hoffman, J. Marques, Jaesik Min, and W. Worek, "Overview of the face recognition grand challenge," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 947–954 vol. 1, 2005.

[50] A. M. Martinez, "The ar face database," *CVC Technical Report24*, 1998.

[51] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar, "Localizing parts of faces using a consensus of exemplars," in *CVPR 2011*, pp. 545–552, 2011.

[52] V. Le, J. Brandt, Z. Lin, L. Bourdev, and T. S. Huang, "Interactive facial feature localization," in *Computer Vision – ECCV 2012* (A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds.), (Berlin, Heidelberg), pp. 679–692, Springer Berlin Heidelberg, 2012.

[53] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2879–2886, 2012.

[54] M. Köstinger, P. Wohlhart, P. M. Roth, and H. Bischof, "Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 2144–2151, 2011.

[55] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[56] H. Qin, J. Yan, X. Li, and X. Hu, "Joint Training of Cascaded CNN for Face Detection," *Cvpr*, pp. 3456–3465, 2016.

[57] G. Tzimiropoulos and M. Pantic, "Fast Algorithms for Fitting Active Appearance Models to Unconstrained Images," *International Journal of Computer Vision*, vol. 122, no. 1, pp. 17–33, 2017.

[58] J. Xiao, T. Moriyama, T. Kanade, and J. Cohn, "Robust full-motion recovery of head by dynamic templates and re-registration techniques," *International Journal of Imaging Systems and Technology*, vol. 13, pp. 85 – 94, September 2003.

[59] C. Cao, Y. Weng, S. Lin, and K. Zhou, "3D Shape Regression for Real-time Facial Animation," *ACM Transactions on Graphics*, vol. 32, no. 4, pp. 1–10, 2013.

[60] J. P. Lewis, K. Anjyo, T. Rhee, M. Zhang, F. Pighin, and Z. Deng, "Practice and Theory of Blendshape Facial Models," in *Eurographics 2014 - State of the Art Reports* (S. Lefebvre and M. Spagnuolo, eds.), The Eurographics Association, 2014.

[61] P. Garrido, L. Valgaert, C. Wu, and C. Theobalt, "Reconstructing detailed dynamic face geometry from monocular video," *ACM Transactions on Graphics*, vol. 32, no. 6, pp. 1–10, 2013.

[62] P. Garrido, M. Zollhöfer, D. Casas, L. Valgaerts, K. Varanasi, P. Pérez, and C. Theobalt, "Reconstruction of Personalized 3D Face Rigs from

Monocular Video," *ACM Transactions on Graphics*, vol. 35, no. 3, pp. 1–15, 2016.

[63] L. A. Jeni, J. F. Cohn, and T. Kanade, "Dense 3d face alignment from 2d videos in real-time," in *11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, vol. 1, pp. 1–8, 2015.

[64] R. A. Güler, G. Trigeorgis, E. Antonakos, P. Snape, S. Zafeiriou, and I. Kokkinos, "DenseReg: Fully Convolutional Dense Shape Regression In-the-Wild," in *Computer Vision and Pattern Recognition (CVPR)*, pp. 2614–2623, IEEE, 2017.

[65] Z. Q. Cheng, Y. Chen, R. R. Martin, T. Wu, and Z. Song, "Parametric modeling of 3D human body shape—A survey," *Computers and Graphics (Pergamon)*, vol. 71, pp. 88–100, 2018.

[66] X. Zhou, M. Zhu, S. Leonardos, K. G. Derpanis, and K. Daniilidis, "Sparseness meets deepness: 3D human pose estimation from monocular video," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 4966–4975, IEEE, 2016.

[67] H. Rhodin, M. Salzmann, and P. Fua, "Unsupervised geometry-aware representation for 3D human pose estimation," in *The European Conference on Computer Vision (ECCV)*, vol. 11214 LNCS, (Munich), pp. 765–782, Springer, 2018.

[68] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Computer Vision and Pattern ...*, IEEE, 2014.

[69] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "Deepercut: A deeper, stronger, and faster multi-person pose estimation model," in *Computer Vision – ECCV*, vol. 9910 LNCS, pp. 34–50, Springer International Publishing, 2016.

[70] D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, and C. Theobalt, "Monocular 3D human pose estimation in the wild using improved CNN supervision," in *International Conference on 3D Vision, 3DV*, pp. 506–516, 2017.

[71] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black, "Keep it SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image," *ArXiv*, pp. 34–36, 2016.

[72] M. Omran, C. Lassner, G. Pons-Moll, P. V. Gehler, and B. Schiele, "Neural Body Fitting: Unifying Deep Learning and Model Based Human Pose and Shape Estimation," in *International Conference on 3D Vision (3DV)*, pp. 484–494, IEEE, 2018.

[73] G. Varol, D. Ceylan, B. Russell, J. Yang, E. Yumer, and C. Schmid, "BodyNet: Volumetric Inference of 3D Human Body Shapes," in *Computer Vision – ECCV*, (Cham), pp. 20—-38, Springer International Publishing, 2018.

[74] H. Joo, T. Simon, and Y. Sheikh, "Total Capture: A 3D Deformation Model for Tracking Faces, Hands, and Bodies *," in *Cvpr 2018*, pp. 8320–8329, 2018.

[75] M. Habermann, W. Xu, M. Zollhoefer, G. Pons-Moll, and C. Theobalt, "DeepCap: Monocular Human Performance Capture Using Weak Supervision," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2020.

[76] D. Lundqvist, A. Flykt, and A. Öhman, "The karolinska directed emotional faces – kdef," *CD ROM from Department of Clinical Neuroscience, Psychology section*, 1998.

[77] A. Jepson, D. Fleet, and T. El-Maraghi, "Robust online appearance models for visual tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1296–1311, 2003.

[78] F. Dornaika and F. Davoine, "On appearance based face and facial action tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 9, pp. 1107–1124, 2006.

[79] R. Yu, C. Russell, N. D. F. Campbell, and L. Agapito, "Direct, dense, and deformable: Template-based non-rigid 3D reconstruction from RGB video," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 11-18-Dece, pp. 918–926, 2016.

[80] G. Hu, F. Yan, J. Kittler, W. Christmas, C. H. Chan, Z. Feng, and P. Huber, "Efficient 3D morphable face model fitting," *Pattern Recognition*, vol. 67, no. February, pp. 366–379, 2017.

[81] C. Cao, M. Chai, O. Woodford, and L. Luo, "Stabilized Real-time Face Tracking via a Learned Dynamic Rigidity Prior," *ACM Transactions on Graphics*, vol. 37, no. 6, p. 233, 2018.

[82] Y. Weng, C. Cao, Q. Hou, and K. Zhou, "Real-time facial animation on mobile devices," *Graphical Models*, vol. 76, no. 3, pp. 172 – 179, 2014.

[83] C. Cao, Q. Hou, and K. Zhou, "Displaced dynamic expression regression for real-time facial tracking and animation," *ACM Transactions on Graphics*, vol. 33, no. 4, pp. 1–10, 2014.

[84] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. NieBner, "Face2Face: Real-Time Face Capture and Reenactment of RGB Videos," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2387–2395, 2016.

[85] Y. Feng, F. Wu, X. Shao, Y. Wang, and X. Zhou, "Joint 3D Face Reconstruction and Dense Alignment with Position Map Regression Network," in *European Conference on Computer Vision (ECCV)* (V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds.), (Munich), Springer, Cham, 2018.

[86] B. Egger, W. A. P. Smith, A. Tewari, S. Wuhrer, M. Zollhoefer, T. Beeler, F. Bernard, T. Bolkart, A. Kortylewski, S. Romdhani, C. Theobalt, V. Blanz, and T. Vetter, "3D Morphable Face Models—Past, Present, and Future," *ACM Transactions on Graphics*, vol. 39, no. 5, pp. 1–38, 2020.

[87] V. J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, C.-J. Wu, B. Anderson, M. Breughe, M. Charlebois, W. Chou, R. Chukka, C. Coleman, S. Davis, P. Deng, G. Diamos, J. Duke, D. Fick, J. S. Gardner, I. Hubara, S. Idgunji, T. B. Jablin, J. Jiao, T. S. John, P. Kanwar, D. Lee, J. Liao, A. Lokhmotov, F. Massa, P. Meng, P. Micikevicius, C. Osborne, G. Pekhimenko, A. T. R. Rajan, D. Sequeira, A. Sirasao, F. Sun, H. Tang, M. Thomson, F. Wei, E. Wu, L. Xu,

K. Yamada, B. Yu, G. Yuan, A. Zhong, P. Zhang, and Y. Zhou, "Mlperf inference benchmark," 2019.

[88] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 483–499, Springer International Publishing, 2016.

[89] M. Rajchl, M. C. H. Lee, O. Oktay, K. Kamnitsas, J. Passerat-Palmbach, W. Bai, M. Damodaram, M. A. Rutherford, J. V. Hajnal, B. Kainz, and D. Rueckert, "Deepcut: Object segmentation from bounding box annotations using convolutional neural networks," *IEEE Transactions on Medical Imaging*, vol. 36, no. 2, pp. 674–683, 2017.

[90] Z. Zhang, P. Luo, C. C. Loy, and X. Tang, "Facial landmark detection by deep multi-task learning," in *European Conference on Computer Vision (ECCV)*, pp. 94–108, Springer International Publishing, 2014.

[91] W. Li, F. Abtahi, and Z. Zhu, "Action unit detection with region adaptation, multi-labeling learning and optimal temporal fusing," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 6766–6775, 2017.

[92] Z. Shao, Z. Liu, J. Cai, and L. Ma, "Deep Adaptive Attention for Joint Facial Action Unit Detection and Face Alignment," *ArXiv*, 2018.

[93] N. Rathee and D. Ganotra, "An efficient approach for facial action unit intensity detection using distance metric learning based on cosine similarity," *Signal, Image and Video Processing*, vol. 12, no. 6, pp. 1141–1148, 2018.

[94] E. Sanchez-Lozano, G. Tzimiropoulos, and M. Valstar, "Joint Action Unit localisation and intensity estimation through heatmap regression," in *BMVC*, pp. 1–12, 2018.

[95] P. Ekman, W. V. Friesen, and H. Hager, *Facial action coding system. Investigator's Guide*. 2002.

[96] D. Aneja, A. Colburn, G. Faigin, L. Shapiro, and B. Mones, "Modeling Stylized Character Expressions via Deep Learning," in *Computer*

Vision ACCV 2016 (S.-H. Lai, V. Lepetit, K. Nishino, and Y. Sato, eds.), vol. 1, pp. 136—-153, Springer International Publishing, 2017.

[97] G. Pons and D. Masip, "Multi-task, multi-label and multi-domain learning with residual convolutional networks for emotion recognition," *CoRR*, 2018.

[98] N. Jain, S. Kumar, A. Kumar, P. Shamsolmoali, and M. Zareapoor, "Hybrid deep neural networks for face emotion recognition," *Pattern Recognition Letters*, vol. 115, pp. 101 – 106, 2018. Multimodal Fusion for Pattern Recognition.

[99] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black, "Generating 3d faces using convolutional mesh autoencoders," in *European Conference on Computer Vision (ECCV)* (V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds.), (Cham), pp. 725–741, Springer International Publishing, 2018.

[100] T. Baltrušaitis, M. Mahmoud, and P. Robinson, "Cross-dataset learning and person-specific normalisation for automatic Action Unit detection," in *International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, pp. 1–6, IEEE, 2015.

[101] E. Ohn-Bar and M. M. Trivedi, "Looking at humans in the age of self-driving and highly automated vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 90–104, 2016.

[102] L. Fridman, "Human-centered autonomous vehicle systems: Principles of effective shared autonomy," *CoRR*, vol. abs/1810.01835, 2018.

[103] J. Olivares-Mercado, K. Toscano-Medina, G. Sanchez-Perez, H. Perez-Meana, and M. Nakano-Miyatake, "Face recognition system for smartphone based on lbp," in *2017 5th International Workshop on Biometrics and Forensics (IWBF)*, pp. 1–6, 2017.

[104] H. Baqeel and S. Saeed, "Face detection authentication on smartphones: End users usability assessment experiences," in *2019 International Conference on Computer and Information Sciences (ICCIS)*, pp. 1–6, 2019.

[105] Y. Lin, J. Shen, S. Cheng, and M. Pantic, "Mobile face tracking: A survey and benchmark," *Proceedings of the Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA*, pp. 18–22, 2018.

[106] J. M. Saragih, S. Lucey, and J. F. Cohn, "Real-time Avatar Animation from a Single Image," in *Automatic Face & Gesture Recognition and Workshops*, pp. 117–124, IEEE, 2011.

[107] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, 02 2009.

[108] Y. Wu and Q. Ji, "Facial Landmark Detection: a Literature Survey," *International Journal of Computer Vision*, 2018.

[109] J. M. Saragih, S. Lucey, and J. F. Cohn, "Face alignment through subspace constrained mean-shifts," in *IEEE 12th International Conference on Computer Vision*, pp. 1034–1041, Sept 2009.

[110] S. Ren, X. Cao, Y. Wei, and J. Sun, "Face alignment at 3000 FPS via regressing local binary features," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, no. 1, pp. 1685–1692, 2014.

[111] T. Baltrušaitis, A. Zadeh, Y. C. Lim, and L. P. Morency, "OpenFace 2.0: Facial behavior analysis toolkit," in *Proceedings - 13th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2018file:///home/goe/Downloads/baltrusaitis2018.pdf*, IEEE, 2018.

[112] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[113] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," in *Proceedings of the IEEE International Conference on Image Processing*, vol. 1, pp. 900–903, 2002.

[114] M. Zhou, Y. Wang, X. Feng, and X. Wang, "A robust texture preprocessing for AAM," in *Proceedings of the International Conference on Computer Science and Software Engineering*, vol. 2, pp. 919–922, 2008.

[115] S. Suzuki and K. Be, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985.

[116] T. Sim, S. Baker, and M. Bsat, "The CMU pose, illumination, and expression database," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 12, pp. 1615–1618, 2003.

[117] A. Zadeh, T. Baltrušaitis, and L. P. Morency, "Convolutional Experts Constrained Local Model for Facial Landmark Detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2017-July, pp. 2051–2059, IEEE, 2017.

[118] E. Wood, T. Baltrušaitis, L.-P. Morency, P. Robinson, and A. Bulling, "Learning an appearance-based gaze estimator from one million synthesised images," in *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, pp. 131–138, 2016.

[119] J. Ahlberg, "Candide-3 - an updated parameterized face," 2001.

[120] D. F. Dementhon and L. S. Davis, "Model-based object pose in 25 lines of code," *Int. J. Comput. Vision*, vol. 15, p. 123–141, June 1995.

[121] J. More, *Levenberg–Marquardt algorithm: implementation and theory.* University of Dundee, Jan 1977.

[122] D. E. King, "Max-margin object detection," *CoRR*, vol. abs/1502.00046, 2015.

[123] N. Markus, M. Frljak, I. S. Pandzic, J. Ahlberg, and R. Forchheimer, "A method for object detection based on pixel intensity comparisons," *CoRR*, vol. abs/1305.4537, 2013.

[124] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, 2016.

[125] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *ArXiv*, 2017.

[126] J. P. Lewis, "Fast template matching," *Pattern Recognition*, vol. 10, no. 11, pp. 120–123, 1995.

[127] J. Shen, S. Zafeiriou, G. G. Chrysos, J. Kossaifi, G. Tzimiropoulos, and M. Pantic, "The first facial landmark tracking in-the-wild challenge: Benchmark and results," in *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pp. 1003–1011, Dec 2015.

[128] J. yves Bouguet, "Pyramidal implementation of the lucas kanade feature tracker," *Intel Corporation, Microprocessor Research Labs*, 2000.

[129] X. Zhu, Z. Lei, X. Liu, H. Shi, and S. Z. Li, "Face alignment across large poses: A 3d solution," *CoRR*, vol. abs/1511.07212, 2015.

[130] P. M. R. Martin Koestinger, Paul Wohlhart and H. Bischof, "Annotated Facial Landmarks in the Wild: A Large-scale, Real-world Database for Facial Landmark Localization," in *Proc. First IEEE International Workshop on Benchmarking Facial Image Analysis Technologies*, 2011.

[131] J. Ostermann, *Face Animation in MPEG-4*, pp. 17–55. John Wiley & Sons, Ltd, 2003.

[132] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Visual simultaneous localization and mapping: a survey," *Artificial Intelligence Review*, vol. 43, pp. 55–81, Jan 2015.

[133] M. Nieto, J. D. Ortega, A. Cortes, and S. Gaines, "Perspective multiscale detection and tracking of persons," in *MultiMedia Modeling* (C. Gurrin, F. Hopfgartner, W. Hurst, H. Johansen, H. Lee, and N. O'Connor, eds.), (Cham), pp. 92–103, Springer International Publishing, 2014.

[134] K. Levenberg, "A Method for the Solution of Certain Non – Linear Problems in Least Squares," *Quarterly of Applied Mathematics*, vol. 2, pp. 164–168, 1944.

[135] D. W. Marquardt, "An algorithm for least-squares estimation of non-linear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[136] "H-Anim standard." Available at: "https://www.web3d.org/working-groups/humanoid-animation-hanim. Accessed: 2020-07-07.

[137] J. Gablonsky and C. Kelley, "A locally-biased form of the direct algorithm," *Journal of Global Optimization*, vol. 21, pp. 27–37, Sep 2001.

[138] M. J. D. Powell, *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*, pp. 51–67. Dordrecht: Springer Netherlands, 1994.

[139] L. Sigal, A. Balan, and M. J. Black, "HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion," *International Journal of Computer Vision*, vol. 87, pp. 4–27, Mar. 2010.

[140] V. Ramakrishna, T. Kanade, and Y. Sheikh, "Reconstructing 3d human pose from 2d image landmarks," in *Computer Vision – ECCV 2012* (A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds.), (Berlin, Heidelberg), pp. 573–586, Springer Berlin Heidelberg, 2012.

[141] I. Akhter and M. J. Black, "Pose-conditioned joint angle limits for 3d human pose reconstruction," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1446–1455, June 2015.

[142] "CMU Graphics Lab motion capture database." Available at: "http://www.mocap.cs.cmu.edu/. Accessed: 2020-07-07.

[143] A. Kar and P. Corcoran, "A review and analysis of eye-gaze estimation systems, algorithms and performance evaluation methods in consumer platforms," *IEEE Access*, vol. 5, pp. 16495–16519, 2017.

[144] E. Kasneci, G. Kasneci, T. C. Kübler, and W. Rosenstiel, "Online recognition of fixations, saccades, and smooth pursuits for automated analysis of traffic hazard perception," in *Artificial Neural Networks* (P. Koprinkova-Hristova, V. Mladenov, and N. K. Kasabov, eds.), (Cham), pp. 411–434, Springer International Publishing, 2015.

[145] Y. Liang, M. L. Reyes, and J. D. Lee, "Real-time detection of driver cognitive distraction using support vector machines," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 2, pp. 340–350, 2007.

[146] M.-C. Chuang, R. Bala, E. Bernal, P. Paul, and A. Burry, "Estimating gaze direction of vehicle drivers using a smartphone camera," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn. Work. (CVPRW)*, p. 165–170, 06 2014.

[147] F. Vicente, Z. Huang, X. Xiong, F. De La Torre, W. Zhang, and D. Levi, "Driver Gaze Tracking and Eyes off the Road Detection System," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 2014–2027, 2015.

[148] R. Zheng, K. Nakano, H. Ishiko, K. Hagita, M. Kihira, and T. Yokozeki, "Eye-gaze tracking analysis of driver behavior while interacting with navigation systems in an urban area," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 4, pp. 546–556, 2016.

[149] S. Jha and C. Busso, "Analyzing the relationship between head pose and gaze to model driver visual attention," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2157–2162, 2016.

[150] L. Fridman, J. Lee, B. Reimer, and T. Victor, "'owl' and 'lizard': patterns of head pose and eye pose in driver gaze classification," *IET Computer Vision*, vol. 10, no. 4, pp. 308–313, 2016.

[151] A. Tawari, K. H. Chen, and M. M. Trivedi, "Where is the driver looking: Analysis of head, eye and iris for robust gaze zone estimation," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 988–994, 2014.

[152] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "Appearance-based gaze estimation in the wild," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4511–4520, 2015.

[153] E. Wood, T. Baltruaitis, X. Zhang, Y. Sugano, P. Robinson, and A. Bulling, "Rendering of eyes for eye-shape registration and gaze

estimation," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 3756–3764, 2015.

[154] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2242–2251, 2017.

[155] R. Ranjan, S. De Mello, and J. Kautz, "Light-weight head pose invariant gaze tracking," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 2237–22378, 2018.

[156] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "Mpiigaze: Real-world dataset and deep appearance-based gaze estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 1, pp. 162–175, 2019.

[157] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 21–37, Springer International Publishing, 2016.

[158] C. G. Broyden, "The Convergence of a Class of Double-rank Minimization Algorithms: 2. The New Algorithm," *IMA Journal of Applied Mathematics*, vol. 6, no. 3, pp. 222–231, 1970.

[159] R. Fletcher, "A new approach to variable metric algorithms," *The Computer Journal*, vol. 13, no. 3, pp. 317–322, 1970.

[160] D. Goldfarb, "A Family of Variable Metric Methods Derived by Variational Means," *Mathematics of Computation*, vol. 24, pp. 23–26, 1970.

[161] D. F. Shanno, "Conditioning of Quasi-Newton Methods for Function Minimization," *Mathematics of Computation*, vol. 24, pp. 647–656, 1970.

[162] E. Haines, *Point in Polygon Strategies*, p. 24–46. USA: Academic Press Professional, Inc., 1994.

[163] UNESCO, "Investing in cultural diversity and intercultural dialogue," 2009.

[164] R. Kurin, "Safeguarding intangible cultural heritage," *Museum international*, no. 1-2, pp. 66–77, 2004.

[165] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series.," in *International Conference on Knowledge Discovery and Data Mining*, no. 16 in AAAIWS'94, pp. 359–370, Seattle, WA, AAAI Press, 1994.

[166] N. M. Khan, S. Lin, L. Guan, and B. Guo, "A visual evaluation framework for in-home physical rehabilitation," in *International Symposium on Multimedia*, pp. 237–240, IEEE, 2014, 2014.

[167] O. Çeliktutan, C. B. Akgul, C. Wolf, and B. Sankur, "Graph-based analysis of physical exercise actions," in *International workshop on multimedia indexing and information retrieval for healthcare*, pp. 23–32, ACM, 2013, 2013.

[168] G. Sun, P. Muneesawang, M. Kyan, H. Li, L. Zhong, N. Dong, B. Elder, and L. Guan, "An advanced computational intelligence system for training of ballet dance in a cave virtual reality environment," in *International Symposium on Multimedia*, pp. 159–166, IEEE, 2014, 2014.

[169] C. J. Su, "Personal rehabilitation exercise assistant with kinect and dynamic time warping," *International Journal of Information and Education Technology*, no. 4, p. 448, 2013.

[170] A. Aristidou, E. Stavrakis, and Y. Chrysanthou, "Motion analysis for folk dance evaluation," in *Eurographics Workshop on Graphics and Cultural Heritage*, pp. 55–64, Eurographics Association, 2014.

[171] D. S. Alexiadis and P. Daras, "Quaternionic signal processing techniques for automatic evaluation of dance performances from mocap data," *Transactions on Multimedia*, no. 5, pp. 1391–1406, 2014.

[172] A. Ahmadi, F. Destelle, C. Richter, D. Monaghan, N. Connor, and K. Moran, "Framework for comprehensive analysis of a swing in sports using low-cost inertial sensors," in *International Sensors Conference*, pp. 2211–2214, IEEE, 2014.

[173] A. Savitzky and M. J. Golay, "Smoothing and differentiation of data by simplified least squares procedures.," *Analytical chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

[174] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *Transactions on Image Processing*, no. 4, pp. 600–612, 2004.

Appendix A

Appendix Title

A.1 Candide model modified deformations

This section shows the modified AU and SU deformations applied to the original Candide [119] model to define our Candide-3m model.

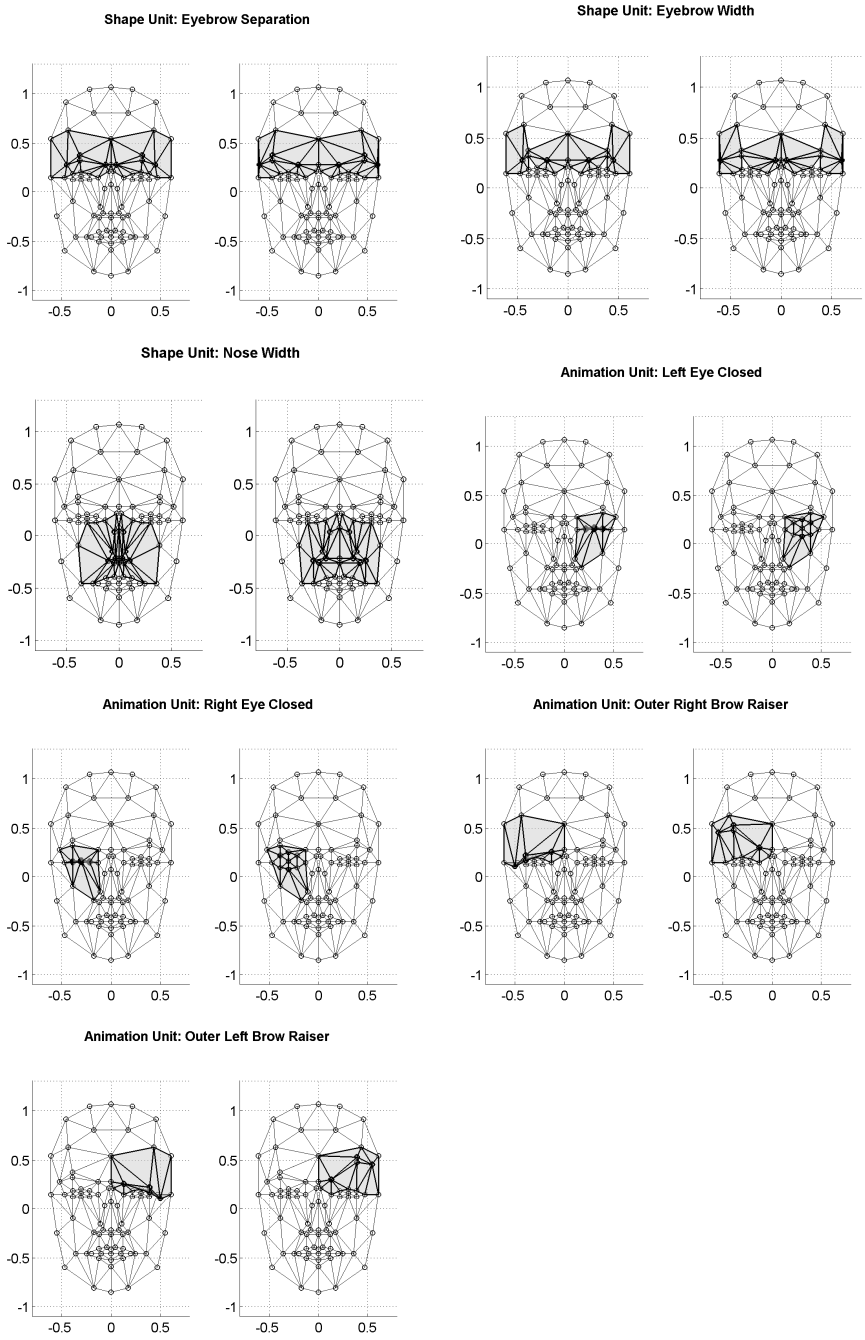


Figure A.1: The added and modified SUs and AUs in Candide-3m with respect to Candide-3, showing their variation from -1 to 1 values, where 0 corresponds to the neutral configuration.