Master in Artificial Intelligence (UPC, URV, UB)

Master of Science thesis

Head pose recovery and shape estimation in still images

Marc Oliu Simón

Advisors: Sergio Escalera and Xavier Pérez-Sala

June 29, 2014

Abstract

Head pose recovery and shape estimation are two tasks of Computer Vision with wide fields of application which usually are applied together in order to extract information from face images. The range of applications for these tasks include face recognition [37, 6], 3D modelling [5] and vigilance monitoring [22, 4], which are useful, among others, for procedures such as driver attention estimation, security systems and computer graphics.

The focus of current research in this area is the development of more robust and accurate approaches that can perform head pose recovery and shape estimation in a faster way, being able to perform both tasks in real-time without the use of expensive hardware. While approaches have been developed for real-time shape estimation [14, 9, 38] and head pose recovery [29, 20], these algorithms perform only one of these tasks, with the methods capable of performing both of them barely being efficient enough to run in real-time using commodity hardware [1]. Some exceptions are the compound techniques, where the 2D geometry is first obtained, afterwards fitting a 3D model to that geometry.

In this work, a new algorithm based on the *Supervised Descent Method* (SDM) [38] is proposed, which is capable of simultaneously recovering the 3D geometry of the face and estimating the head pose, given a grayscale image. The main differences between SDM and the new proposed approach are the use of a parametric ASM model instead of directly regressing the model shape, the use of SIFT descriptors at the landmarks adapted to the shape scale, and a robust method to estimate the final fitting from multiple initializations.

Furthermore, two approaches are studied and compared in order to extend the original 2D method to the fitting of 3D geometries in grayscale images: The frist approach consists on using an iterative alignment method in order to adjust a 3D ASM model to the obtained 2D shape, while the second consists on directly fitting the 3D model to the image.

Contents

1 Introduction

Two recurring problems in computer vision are the recovery of an object geometry and estimation orientation in space. The object geometry, or shape, is defined as a set $L = \{L_i | L_i = \langle x_i, y_i[, z_i] \rangle\}$ of landmarks, where each landmark corresponds to a point location, consistently labeled among different shapes. The problem of locating each landmark in the image plane by giving its 2D or 3D coordinates is referred to as geometry fitting or shape alignment. The problem of pose estimation for rigid or semi-rigid objects, as in the case of heads, consists on determining the orientation of the object in the given image. This is done by determining the rotation angle of the object when considering the space where the object is found as a plane, or by giving the three rotation angles consisting on roll, pitch and yaw in the case of an object positioned in a 3D space.

In this work, a method is developed which performs facial shape alignment and head pose recovery on intensity images, locating a set of 2D/3D specific face landmarks, and giving the three angles of rotation specifying the orientation of a face image relative to the camera point of view. The proposed algorithm is based on the *Supervised Descent Method* (SDM), developed by Xiong and De la Torre [38], an algorithm that implements a cascade of linear regressors estimating the descent direction of each landmark. In order to do so, SDM is initialized with the mean face shape, and for each cascade step, a fast *Scale-Invariant Feature Transform* (SIFT) [26] descriptor is extracted at each estimated landmark location. The SIFT descriptors are then concatenated and a *Principal Component Analysis* (PCA) [30] transform is applied to reduce their dimensionality. The resulting feature vector is used as input of the linear regressor to estimate the descent direction and step length for each landmark, obtaining a better estimate of the shape for the next cascade step. This method is explained with more detail in Section 3.1.

The proposed method modifies the SDM algorithm by optimizing the deformation weights corresponding to an Active Shape Model (ASM) [13] describing the allowed variability of a face model, instead of directly adjusting the position of the face landmarks. Furthermore, the position of the model in the image, its scale and rotation angle, are also optimized. Another change introduced in the new method is the use of adaptable SIFT windows, which are resized according to the scale parameter at each cascade step. Finally a new approach to selecting the best fit among different initializations is used, keeping the centroid fit (the one minimizing the the sum of euclidean distances to the other fits).

From the proposed parameterized approach for the 2D shape regression, two different approaches are developed to obtain the 3D face geometry and head pose. The first one consists on aligning a 3D ASM to the 2D geometry by means of a restricted camera model [18] in order to obtain the corresponding 3D geometry and head pose. The second one extends the proposed 2D shape regression method to directly regress the 3D geometry. This is done by regressing the weights of the 3D ASM and the parameters of the restricted camera model.

In the rest of the document, first, an overview on the state of the art on both face shape estimation and head pose recovery is given in Section 2, explaining the main types of methods and some examples of each. Afterwards, the proposed method is explained in Section 3, showing the differences between the current approach and the SDM method it is based on, also detailing how other techniques are integrated at different steps of the process.

Finally, the experimental methodology is proposed in Section 4 with a series of experiments to assess both the accuracy of fitting the landmarks of the proposed 2D/3D methods, and the accuracy of the head pose estimation for the 3D methods, as well as the required execution time for each proposed technique. The results for these experiments are then described in Section 5, showing a slight improvement in the accuracy for the parametric approach over the SDM method it is based on, while still maintaining the same average computational time. From the two proposed 3D geometry fitting approaches, the direct *3D regression* approach is found to obtain an equivalent accuracy when compared to first performing a 2D regression and then aligning a 3D ASM model to it, but with the direct *3D regression* being much faster.

2 State of the art

In the case of faces, the geometry can be estimated as a set of 2D/3D landmarks at anatomical locations of the face, while the pose is always specified in terms of roll pitch and yaw. In this section the main techniques currently used for estimating both the shape and pose of a face are explained, commenting on their performance and weaknesses. First the shape estimation [13, 15, 27, 38] and pose estimation [19, 18, 8, 20] techniques are explained separately, afterwards explaining some approaches which are capable of performing both processes simultaneously [39, 32, 1].

2.1 Shape estimation

There are three main types of shape estimation methods for face images, as shown in Figure 1. The most basic approaches are those based on *feature detection* [7]. These consist on locating each landmark individually, either by matching local image descriptors or by using a set of previously known locations as a reference.

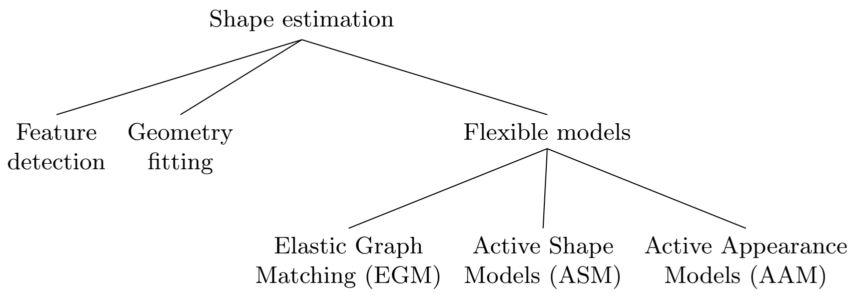

Shape estimation
Feature detection
Geometry fitting
Flexible models
Elastic Graph Matching (EGM)
Active Shape Models (ASM)
Active Appearance Models (AAM)

Figure 1: Classification of facial shape estimation methods according to the type of information these make use of.

More complex methods include *shape regression* [15, 11, 9, 38] and *deformable models* [25, 28, 16, 27]. They both have in common the joint detection of all desired landmarks, using all of the available descriptors of the face appearance given the current face geometry to help locate each landmark. While shape regression methods directly optimize the location of the landmarks, deformable models use a previously trained model describing the allowed modes of variation of the face in order to fit the model to the image. This last type restricts the overall shape of the face as a whole, usually performing a dimensionality reduction on a set of training geometries in order to explain the shape variations with a smaller set of parameters.

2.1.1 Feature detection

A naive approach to detecting the face shape consists on independently locating the face landmarks [7, 14]. This can be done by applying an array of detectors to the face, each one of which directly detects an individual feature or helps locating each feature separately.

An example of this type of approach is the method developed by Bonitto and Cumani [7], which first performs a segmentation of the face region. With the segmented face, silhouette landmarks are located by checking the curvature of the different pixels and selecting those matching the desired one. Face landmarks are then located by using gradient intensity criteria at the location and the relative position of the silhouette landmarks. While this method uses information on other landmarks, only secondary landmarks (those found at the silhouette) are taken into account, and these are estimated individually.

Another example is the method developed by Dantone *et al.* [14], which uses conditional regression forests in order to locate a set of facial landmarks. Each tree is trained with a set of randomly selected facial patches from a subset of the training images, and uses both Haar-like features and Gabor wavelets as features. At the terminal nodes, each tree specifies the relative location at which each landmark is to be found. In order to locate the landmarks in a face region, a dense grid of facial patches is fed to all trees, each casting a vote to where each facial feature is located. While each regression tree helps locating all facial landmarks, the location of each landmark is found independently of the rest, without updating the position of a landmark relative to the estimated position of the rest.

2.1.2 Shape regression

The methods based on directly fitting the geometry adjust all the face landmarks simultaneously, iteratively minimizing the fitting error of the geometry over the image [15, 11, 38]. These methods are based on detecting the descent direction of each one of the landmarks, but using the same input information at each landmark. This means there is a certain degree of correlation between the different landmarks descent directions.

A simple and extremely fast model fitting the face geometry is *Cascaded Pose Regression* [15, 9], which consists on a cascade of fern regressors. The algorithm is initialized with a set of initial shapes, and at each step of the cascade the shape is adjusted by means of a fern regressor. The regressor uses geometry-indexed pixel intensity differences as inputs and determines a variation for each of the shape landmarks. The output of the different regressor determines an update to the face geometry to be used in the next cascade step. Once each initial shape has been adjusted to the image, the one minimizing the distance to the rest is kept as the best solution. While this approach is very fast due to the simplicity of its features, it is for the same reason that it tends to get stuck on local minima.

A similar more robust approach is the one developed by Cao *et al.* [11]. This approach uses multiple regressors at each cascade step, indexing the feature pixels by using a linear interpolation between two landmarks instead of a coordinate relative to a single landmark position. By using multiple regressors at each step, the algorithm is more robust to local minima, also requiring less cascade steps to converge.

A more accurate approach is the *Supervised Descent Method* (SDM) [38], developed by Xiong and De la Torre. This method also starts with multiple initial face geometries which are fitted to the face image through a cascade of linear regressors, afterwards using the mean fitted face as the final solution. In order to do so, a simplified SIFT descriptor is generated for each of the n landmarks of the face geometry at each cascade step i. These descriptors are then concatenated into a single feature vector, and the dimensionality of the vector is reduced using PCA. This vector of features Φ_i is then used as parameters of the linear regressors, with each regressor determining an update to each of the $2 \cdot n$ geometry parameters. During training, the linear regressors learnt the descent direction and step length from the training data by indirectly minimizing the error $\Phi_i - \Phi_*$ between the target and current descriptor values at each cascade step as explained in Section 3.1. This method is less sensible to local minima thanks to the use of more robust features, while still being fast enough to perform real-time fitting of video sequences.

2.1.3 Flexible models

The methods based on flexible models fit a non-rigid model to the facial structure for which the pose is to be recovered [37, 28]. In this approach, the model is adjusted through a series of parameters that define the geometry and restrict it instead of directly adjusting the landmarks.

Elastic Graph Matching

The most basic flexible model is *Elastic Graph Matching* (EGM) [25], developed by Lades *et al.*. This method consists on generating a deformable geometric model called *bunch graph*, where different face features are represented as nodes of a graph $g_i = \langle V, E \rangle$, with each edge E capturing the relative position between two vertices. The vertices V describe the image position to match through a jet of wavelet responses. In order to perform head pose estimation, a model bunch graph is generated for each discrete pose, and given a new face image, the minimum distance to each elastic graph is calculated and the pose of the one with the minimum distance assigned. In order to fit a model bunch graph to the image, the formula describing the matching error seen in Equation 1 is minimized through a two-step process.

$$C_{total}(\{x_i^I\}) = \lambda \sum_{i,j \in E} S_e(\overrightarrow{\Delta}_{ij}^I, \overrightarrow{\Delta}_{ij}^M) - \sum_{i \in V} S_v(D^I(x_i^I), D_i^M) \tag{1}$$

This equation penalizes by one side the differences S_e for each edge $\langle i, j \rangle \in E$ between the distances $\overrightarrow{\Delta}_{ij}^M$ in the model graph and the distances $\overrightarrow{\Delta}_{ij}^I$ in the image graph. By the other, it penalizes the differences S_v for each vertex $i \in V$ between the jet descriptor D_i^M of the model and the one obtained at the corresponding image location, $D^I(x_i^I)$. The λ parameter is the weight of the deformation penalty. In a first step only the position of the image graph is adjusted through annealing. Once the position is determined, the different landmark positions are updated individually, also following an annealing stochastic descent approach.

Wiskott *et al.* [37] improved on EGM with the development of their *Elastic Bunch Graph Matching* (EBGM) algorithm, which uses the same graph representation of EGM. The main contribution of the new approach is its image fitting algorithm, which is faster while obtaining more accurate results thanks to a coarse to fine grain location of the graph vertices, allowing for a better initial location of the graph in the image before individually optimizing the vertices independently. The algorithm is a four-step heuristic approach minimizing the error seen in Equation 1. In the first step only the face position is adjusted from an initial position estimate, afterwards adjusting both position and size. As a third step, the size is further refined, but allowing for an aspect ratio distortion, that is, allowing different scales for width and height. Finally, the graph nodes are allowed to vary their position independently.

Active Shape Models

Another type of model is the *Active Shape Model* (ASM) [13], which is based on fitting a deformable model to a given face image. In order to do so, first a group of landmark points are labelled over the training set, afterwards iteratively aligning them by minimizing the distance between each training instance and the mean of the aligned instances with Procrustes Analysis (PA), with the mean shape being re-calculated at each iteration. By performing a *Principal Component Analysis (PCA)* over the variations between the aligned data and the mean shape, the variation of the face geometry is captured in a space of lower dimensionality. This new lower dimensionality space captures most of the data variance through a set of orthonormal dimensions or bases known as eigenvectors. Each one describes a mode of variation of the geometry, with its eigenvalue specifying the proportion of the geometry variance it explains. By assigning a weight to each one of these new bases, a face geometry is defined.

In the original ASM work by Cootes *et al.* [13] an algorithm is provided for fitting the model to an image, in which weights are updated using an iterative adjustment method. This method uses either the direction and intensity of the edges near each landmark (for edges lying at the contour of the fitted figure), or a simple vector representation of the visual appearance at the current landmark location. After each adjustment, the weights are rescaled as necessary in order to lie inside the PCA ellipsoid defined by the maximum variation allowed for each weight, which is proportional to its corresponding eigenvalue.

Milborrow and Nicolls [28] developed an extension of the typical ASM fitting approach which uses the gradients of fixed-size squared regions around each face landmark, as well as an adaptive maximum threshold which is decreased over time for each weight based on the corresponding eigenvalue, in order to improve the model fitting accuracy. This allows for more fitting flexibility in the first iteration, restricting the model to fall inside the accepted shape variation as the model converges.

While ASM image fitting methods tend to provide accurate results given a good model of the shape variability, a good initialization of the shape is required, since the algorithm is sensible to local minima. This problem also extends to other *Active Appearance Model* (ASM) approaches to a lesser degree, which are explained below.

Active Appearance Models

Active Appearance Models (AAM) [12, 16, 27] are an extension of ASM, where not only the geometric deformation of the data is captured, but also the appearance variation. In order to do so, first the geometric variation model is generated, afterwards wrapping the image texture of each individual image with a transform bringing the landmark locations to the mean geometric shape. With the image textures aligned, it is now possible to capture their variation by using a PCA. The final model captures the shape and texture variations in an uncorrelated manner.

When fitting an AAM to an image, the goal is to select the shape and appearance weights in order to minimize the difference between the target image and the parametrized model. This is represented by the minimization function seen in Equation 2, where each model pixel $x \in s_0$, defined by its mean appearance $A_0(x)$ plus a linear combination of q appearance bases $A_i(x)$ given by the weights a should match the appearance at the corresponding image pixels, which are aligned with the model through a wrapping function $W(x; b)$. The wrapping function determines the location of a given model pixel on the image by performing a translation according to the model shape, given by the shape weights b. The parameters to optimize are the shape weights b and the appearance weights a.

$$\sum_{x \in s_0} \left[A_0(x) + \sum_{i=1}^{q} a_i A_i(x) - I(W(x; b)) \right]^2 \tag{2}$$

Edwards *et al.* [16] developed a method using AAM which learns linear regressors that determine the variation on the shape and appearance weights, as well as the position, rotation and scale of the model, through linear regression. The model uses a cascade of regressors that use the differences between the image pixel intensities and those of the current model appearance to update the parameters. The problem with this type of method is that by trying to optimize appearance and shape simultaneously, non-linear relations with the pixel differences appear, requiring many iterations to converge and not giving highly accurate results.

In order to solve these problems, Mathews and Baker showed in [27] an efficient way to use Inverse Compositional Image Alignment (ICIA) algorithm to align a 2D AAM to a given image. Their approach keeps the advantages of ICIA, which allows for the pre-computation of the Jacobian J and inverse Hessian H^{-1} matrices only once before beginning the optimization. Furthermore, they show a way to decouple the optimization of the shape weights b from the appearance weights a, iteratively optimizing the first and giving a closed form solution for the later once computed the shape weights.

2.2 Pose estimation

Head pose estimation consists on determining the orientation of the head, usually in terms of the roll, pitch and yaw rotations relative to the frontal face view, and can be divided into two main groups, as shown in Figure 2. The first group includes methods that directly estimate the head pose based on the appearance of the head image [8, 23, 29, 20]. The second one uses previous information on the face geometry in order to estimate the pose from the landmark locations [36, 19, 18].

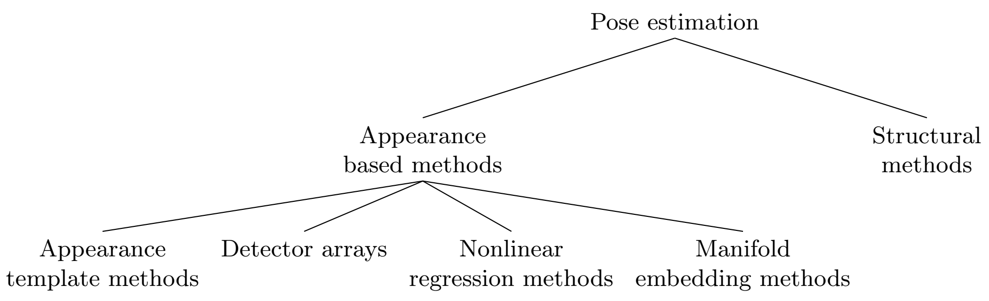

Pose estimation
Appearance based methods
Structural methods
Appearance template methods
Detector arrays
Nonlinear regression methods
Manifold embedding methods

Figure 2: Classification of head pose estimation methods according to the type of information these make use of.

2.2.1 Appearance based methods

Appearance based methods define a series of features directly describing the visual appearance of the image, afterwards using them to perform a classification or regression determining the head pose. Appearance based methods can be classified into different subtypes depending on the method used to determine the pose.

Appearance template methods

Consists on comparing a new head image to a set of examples labelled with their pose and assigning the pose of the most similar example. This approach can consist on a direct *image comparison* between the target and any template, which can be performed by calculating the Mean Squared Error (MSE) or the normalized cross-correlation between images. This direct approach suffers from many drawbacks, the main one being that the similarity between different poses for the same person could be greater than the similarity between two face images of different persons in the same pose, thus generating a poor head pose estimator.

Another approach is to perform a *filtered image comparison*, where two feature vectors generated from the evaluated and template images are compared. The feature vectors are obtained either by applying a dimensionality reduction to the pixel intensities by means of PCA, or by using other direct filtering techniques like Gabor wavelets. This approach is more resilient to the problems specific to direct image comparison. An example of this approach is the method developed by Brown and Tian [8], which uses Gabor templates generated from averaging the output of Gabor wavelets at each image pixel at four different orientations, for all images of a given pose. In order to match a new face, its Gabor templates are generated and compared to those of the different considered poses.

Two problems common to both *image comparison* and *filtered image comparison* are the great number of comparisons to be performed, which means the approach is computationally complex, and to a lesser degree the fact that only a discrete set of poses are considered. The later problem is in some cases overcome by performing interpolation between similarly-scored head poses and comparing the target to the interpolated images. Another handicap for this type of approach is that the face is assumed to have been previously located in the image.

Detector arrays

Consists on training a detector for each discrete pose and assigning the pose of the detector with the highest support. This approach is based on the general object detection methods, where a detector is trained for each of a set of discrete poses, and solves many of the problems of the *appearance template models.*

There are two main approaches for *detector arrays.* The most common one is the application of individual detectors, each one casting a vote for a given position. This type of approach includes machine learning techniques like *Support Vector Machines* (SVM) and AdaBoost cascade classifiers. A good example of this approach is the method developed by Jones and Viola [23] as an extension of their face detection algorithm. In this new algorithm, a decision tree made of AdaBoost classifiers is built in order to determine an estimated head pose before applying the face detector itself.

On the other hand, a *router network* can be used to first identify the most likely pose, and afterwards a conventional classifier used in order to determine if that pose is indeed present in the analysed image. This approach is followed by Rowley *et al.* [33], where a *router network* determines the roll angle of the interest region by directly using the intensity values of the histogram-equalized region, prior to performing face detection.

Since each detector is also capable of determining the presence of the face for that given pose, it is not necessary to perform face detection as a previous step. Also, the trained detectors are capable of ignoring face variations not corresponding to pose changes.

The problem of having to apply multiple detectors found for *appearance template models* still persists when using individual detectors, although it is solved in the case of using a routing network as a previous step. In addition, this type of approach poses the problem of having to train each classifier with negative examples of the other classes. This makes the training difficult when a high number of discrete poses are taken into account, having negative examples very similar to the positive ones.

Nonlinear regression methods

Consists on generating a non-linear regression model which estimates the head pose from a feature vector representing the image data, with the data being either the raw image data or some form of dimensionally-reduced set of features, normally obtained through PCA. The most common regression methods used are *Support Vector Regression* (SVR) methods and Neural Networks methods.

There are two main approaches to using these methods. One possibility is to generate multiple outputs, each one of them indicating the likelihood for a specific discrete pose, which is usually done with *Multilayer Perceptron* (MLP) networks. Brown and Tian [8] developed an approach which uses Gabor wavelets at four different orientation for each pixel of the histogram-equalized face region as features for a 3-layer MLP network. The network has as output 9 continuous values giving the probability for each one of the 9 considered yaw values.

Another type of approach consists on generating a continuous output value itself representing the head pose. In that case normally a regression model is trained for each individual degree of freedom. This is the case of the algorithm developed by [29], which estimates the pitch an yaw of the head by using two SVR regressors. These use an *Histogram of Oriented Gradients* (HOG) descriptor of the head region as the feature vector. On the other hand, Stiefelhagen *et al.* [35] use a neural network simultaneously estimate the head roll and yaw, using a single MLP with two continuous outputs.

This type of approach is very fast, and yields good results specially in the continuous approach. The main problem for these methods is the introduction of errors due to a bad face localization, which is not directly performed by the method itself.

Manifold embedding methods

These methods consist on obtaining a low-dimensional representation of the image capturing the variation of the head pose. The image representation is then used to perform a classification or regression to determine the head pose parameters.

Srinivasan and Boyer [34] follow the eigenfaces approach in order to determine the best pose estimation for a face image. In order to do so, a PCA is performed for each view of a given range of orientations, creating a face model for each orientation. When estimating the pose of a new face, the weights for each view model best describing the given face image are found, and the view that more accurately fits the face is kept as the correct one.

Jain and Crowley [20] use a set of Gaussian derivative filters at the face region pixels at two different scales in order to describe the face, afterwards applying PCA to reduce the feature vector dimensionality. The dimensionally reduced data is then used by two SVM classifiers using a Radial Basis kernel in order to determine the roll and yaw parameters of the face orientation.

2.2.2 Structural methods

This second group of methods detect the face pose based on the face geometry in addition to possible information on the image appearance. These approaches normally use of a set of landmark locations to be found in the face in order to geometrically analyse the image when determining the pose. To do so, the relative positions of the features are evaluated following a pre-defined geometric model. This type of methods are based on psychological studies on human perception of the head and gaze orientation [36]. These studies show that basic criteria such as bilateral symmetry of the head and deviation of the nose tip from the symmetry plane are some of the cues humans use to determine the orientation.

In [19], the inner and outer eye corners and nose tip locations are used in order to estimate the face projection. To do so, an orthographic camera model is used to determine the best rotation of a 3D model that matches these points in the monocular image. Another approach developed by Qiang is to approximate the face region to an ellipse [21], adjusting the rotation of the camera in order to account for its distortion. This second approach, though, supposes most faces have the same approximate outer shape, not taking into account the population variability in the height to width ratio of faces.

On the other hand, a scaled orthographic camera model and an iterative process [18] can be used to find the best match of both, the 3D face geometry and pose, given the 2D geometry of the same landmarks.

2.3 Simultaneous approaches

There are some methods capable of estimating both the face geometry and pose simultaneously, without the need of performing both processes independently or in a consecutive manner. The most popular of these approaches are described in this section [39, 32, 1].

Zhu and Ramanan developed a method based on *mixtures of trees* [39] which is not only capable of estimating the face geometry and pose, but also of performing face detection. The method uses a common pool of parts V, which are then organized into mixtures of threes. Each mixture describes a set of restrictions to the position a subset $V_i \in V$ of parts, effectively defining a viewpoint for the general 3D structure. Also, each part $v_i \in V$ has a set of templates v_i, which not only determine the part being detected but are also capable of discriminating the viewpoint, with a given viewpoint only accepting a subset $v_{i_j} \in V_i$ of the part templates. By detecting the different templates on the images, the likelihood of each mixture of trees changes in function of the templates and parts detected and their relative position, performing detection, shape estimation and head pose recovery simultaneously.

It is also possible to make use of AAM (or the closely related *3D Morphable Models* [5]) to directly fit 3D models of the face. By doing this, the geometry of the face is recovered in three dimensions, also recovering the head pose by defining a director vector on the AAM itself. Some of these approaches have been recently studied [32, 1], leading to the development of efficient gradient descent methods to fit them into 2D images. Since these methods describe the variability of faces in three dimensions, the range of movements are more restricted than in the 2D case, where 2D face poses of different perspectives are combined to create a 2D model that can also explain other impossible face configurations.

In [32] Romdhani and Vetter use a modified version of the Inverse Compositional Image Alignment (ICIA) algorithm in order to fit a 3DMM to the image. By using the ICIA it is possible to pre-compute the Jacobian J and inverse of the Hessian H^{-1}, which makes the algorithm much more efficient than directly performing a gradient descent. The problem with this algorithm is that while the shape optimization can be decoupled from the appearance one in the case of 2D models, as seen in Section 2.2.1, in the case of 3D this is not possible [1]. This results in a much more expensive algorithm.

Even though it is not possible to directly perform an efficient ICIA gradient descent implementation on 3D models, a workaround has been proposed by Baker and Matthews [1]. The workaround consists on simultaneously fitting a 3D and 2D model, fitting the 3D model through the 2D one instead of using the image.

3 Method

In this section, the proposed method for recovering the 3D face shape and head pose is explained. The algorihm is based on the *Supervised Descent Method* (SDM), which obtains the 2D geometry of the face and is explained in detail in Section 3.1. The differences with SDM are then detailed in Section 3.2, afterwards proposing two different approaches to extending the fitting method in order to recover the pose and geometry of 3D faces. The first one consists on finding the optimal projection of a 3D ASM to the already obtained 2D landmarks (Section 3.3). The second one modifies the proposed method in order to work with a 3D ASM, directly recovering the 3D geometry and head pose while fitting the image (Section 3.4).

3.1 Supervised Descent Method

The *Supervised Descent Method* (SDM) [38] is a method developed by Xiong and De la Torre for detecting a set of facial landmarks on intensity images which runs in real time, being capable of fitting up to 30 face images per second in current commodity hardware. In order to do so, the algorithm learns a cascade of linear regressors, each one of which further optimizes the location of the facial landmarks given an initial estimation. In order to do so, the algorithm uses a series of simplified *Shape-Invariant Feature Transform* (SIFT) [26] descriptors at each of the current landmarks position estimates.

Landmark descriptors

The simplified SIFT descriptor takes into account an area of 32x32 pixels around the landmark to be described. Because of the fixed window size, the descriptor does not provide scale-invariance, differing from the standard version of SIFT. In the first step, the pixels inside the region are evaluated, subtracting their gradient orientations and intensities according to Equations 3 and 4 respectively. In these equations, x and y are the spatial coordinates of the pixels, and the function G defines the pixel value at these coordinates after applying a Gaussian smoothing with $\sigma = 1.6$.

$$\Theta(x, y) = tan^{-1}\left(\frac{G(x, y+1) - G(x, y-1)}{G(x+1, y) - G(x-1, y)}\right) \tag{3}$$

$$M(x, y) = \sqrt{(G(x+1, y) - G(x-1, y))^2 + (G(x, y+1) - G(x, y-1))^2} \tag{4}$$

The gradient magnitudes are then accumulated into a 36-bin histogram of gradient orientatons according to the measured orientation at each location. When accumulating them, the magnitude is weighted according to a Gaussian function located at the landmark and with a standard deviation equal to half the window size, that is, $\sigma = 16$. Once calculated, the principal gradient orientation θ is obtained from the histogram of gradient orientations, and a new 32x32 window is defined such that it is rotated towards that orientation.

In the second step, the new oriented window gradient orientations and magnitudes are recalculated by following Equations 3 and 4 previously used to calculate the general window histogram of gradient orientations, but using the rotated pixel coordinates instead of the neighbouring pixels. By doing this, the new magnitude and orientation values are measured taking into account the rotation of the window. That is, the new coordinates x' and y', which are obtained according to Equation 5, are used instead.

$$\begin{aligned} x' &= x \cdot cos(\theta) - y \cdot sin(\theta) \\ y' &= x \cdot sin(\theta) + y \cdot cos(\theta) \end{aligned} \tag{5}$$

Once calculated the rotated window gradient orientations and magnitudes, the window is divided into a 4x4 grid, and an histogram of gradient orientations with 8 bins is generated for each grid cell. As in the first step, the magnitude of the gradient is weighed by using a Gaussian weighting with a standard deviation of $\sigma = 16$ centred at the landmark coordinates. Finally, the values of each histogram H are normalized such that the sum of all bin values equals to 1, making the descriptor invariant to illumination intensity changes. The 16 histograms are then concatenated into a single vector of $8 \cdot 16 = 128$ real values. Figure 3 is a visual representation of how a SIFT descriptor is built given the oriented window.

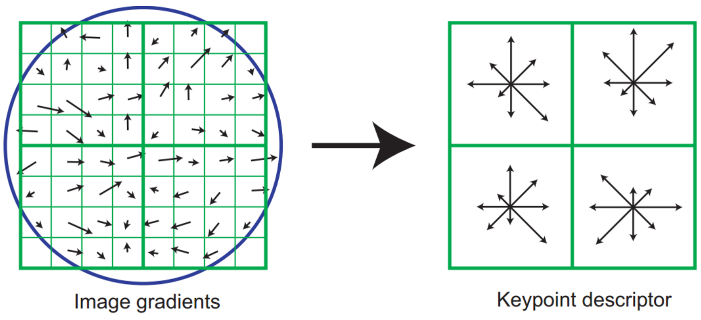


Figure 3: Construction of a SIFT descriptor. The oriented window gradients are divided into bins, in this example into a $2 \cdot 2$ grid, and a histogram of oriented gradients with 8 bins is created for each one. The magnitude of the gradient is weighted by a gaussian function, here represented by a blue circle, with σ equal to half the window size.

Linear regressors

At each cascade step, the algorithm's target is to minimize the error between the landmarks real location and the current estimation. In order to do so, the error shown in Equation 6 is minimized. This function defines the fitting error of the current landmark location L_0 plus the landmark location update ΔL as the squared euclidean distance between the SIFT descriptors $h(x)$ at the updated landmark location $d(L_0 + \Delta L)$ and the SIFT descriptors at the target location $\Phi_* = h(d(L_*))$.

$$f(L_0 + \Delta L_1) = ||h(d(L_0 + \Delta L_1)) - \Phi_*||_2^2 \tag{6}$$

The goal of SDM is to learn a series of descent directions and step sizes in order to iteratively approximate the landmark estimates to the target position, such that $L_1 = L_0 + \Delta L_1$. In order to do so, a second order Taylor expansion is performed over Equation 6, as shown in Equation 7. In this formulation, $J_f(L_0)$ and $H(L_0)$ are correspondingly the Jacobian and Hessian matrices of function f evaluated over L_0.

$$f(L_0 + \Delta L_1) \approx f(L_0) + J_f(L_0)^T \Delta L_i + \frac{1}{2}\Delta L^T H(L_0)\Delta L \tag{7}$$

In [26] it is shown how a first update to the data is derived from Equation 7, which is formulated as $\Delta L_1 = R_0 \cdot (\Phi_0 - \Phi_*)$, where $R_0 = -2H^{-1}J_h^T$. This equation is then reformulated in order to match the form of a linear regressor, as shown in Equation 8.

$$\begin{aligned} \Delta L_1 &= R_0 \cdot \Phi_0 - R_0 \cdot \Phi_* \\ &= R_0 \cdot \Phi_0 - b_0 \end{aligned} \tag{8}$$

With this formulation it is possible to quickly learn the descent matrix R_0 and the offset $b_0 = R_0 \cdot \Phi*$ by solving the least-squares problem. This same approach is used at each step of the cascade to learn a linear regressor of the same form, which looks for the descent direction approximating Φ_i to the target results Φ_*. The original work in [26] experimentally demonstrates that this method converges to the solution with just 4 to 5 cascade steps in most cases.

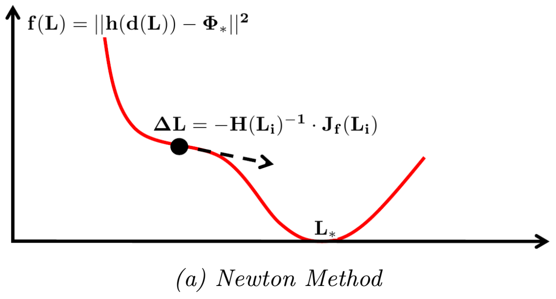

$\mathbf{f(L) = ||h(d(L)) - \Phi_*||^2}$
$\mathbf{\Delta L = -H(L_i)^{-1} \cdot J_f(L_i)}$
$\mathbf{L_*}$
(a) Newton Method
$\mathbf{f(L) = ||h(d(L)) - \Phi_*||^2}$
$\mathbf{\Delta L^1 = R_i \cdot \Phi_i^1}$
$\mathbf{\Delta L^2 = R_i \cdot \Phi_i^2}$
$\mathbf{\Delta L^3 = R_i \cdot \Phi_i^3}$
$\mathbf{L_*^1}$
$\mathbf{L_*^2}$
$\mathbf{L_*^3}$
(b) Supervised Descent Method

Figure 4: Comparison between (a) Newton's descent method and (b) Supervised Descent Method. SDM learns from a set of examples, being able to avoid local minima and converge faster to the solution. Also, it doesn't need to compute the Jacobian nor the Hessian matrices.

Compared to Newton's descent method, as seen in Figure 4, SDM does not need to compute the Hessian and Jacobian matrices at each step in order to obtain the descent direction and magnitude. Instead, it learns the expected descent direction and magnitude given the feature vector Φ_i from a set of examples. This has the advantage of being able to obtain a correct update to L even when the current region is a local minima. Also, since the SDM does not depend on the descent inclination to determine the magnitude, a better estimate is obtained, converging faster to the solution.

Algorithm training

In order to train the linear regressor at each cascade step, first the available training data is augmented. In order to do so, the same images are re-used 10 times, using a different shape initialization at each re-use. Since the target of the regressor is to predict the difference $\Phi_0 - \Phi*$, it is equivalent for the training process to modify the initial geometry or to perform this changes on the image itself. The original SDM algorithm first calculates the mean and variance for the geometry translation and scaling by taking into account the whole dataset ground truth. Afterwards, it uses these values to draw 10 Monte Carlo samples from the distribution.

With the dataset augmented, the cascade of regressors is learned by using the current descriptor values Φ_i, drawn from the estimated face geometry at the given cascade step, and the target shape error $L_{dif} = L_* - L_i$. Each regressor learns a shape variation which is then applied at the current shape estimate L_i in order to get the shape estimate at the next cascade step, L_{i+1}. The whole training process is illustrated with the pseudo-code in Figure 5.

Fitting a test image

With the algorithm already trained, in order to fit an image to the algorithm first the mean shape is placed at the image as the initial estimate for the fitting process. Afterwards, the features Φ_i at the current estimate landmarks are drawn and fit to the cascade step regressor with weights M_R^i and biases M_b^i, obtaining the updated landmark estimates L_{i+1}. The final landmark location estimates are obtained after applying all the cascade regressors.

3.2 Proposed method for fitting the 2D geometry

The proposed method is based on the SDM approach explained in Section 3.1, but with a series of changes to make it more robust and adaptable to 3D shapes. The proposed changes, by one side, prepare the method to be easily extensible to 3D fitting by working on a parametric way instead

```
$L_0 \leftarrow SampleShapes(L_*, ||I|| \cdot 10)$
$M \leftarrow \{\}$
for all $i \in [0:4]$ do
    //Get SIFT feature vectors
    $feat \leftarrow \{\}$
    for all $j \in [1:||I||]$ do
        for all $k \in [j \cdot 10 - 9 : j \cdot 10]$ do
            $feat_k \leftarrow GetSiftDescriptors(L_i^k, I_j)$
        end for
    end for

    //Reduce SIFT vectors dimensionality
    $M_{mft}^i \leftarrow mean(feat)$
    $M_{pca}^i \leftarrow PCA(feat - M_{mft}^i)$
    $\Phi_i \leftarrow M_{pca}^i \cdot (feat - M_{mft}^i)$

    //Get linear regressor
    $L_{dif} \leftarrow L_* - L_i$
    $M_R^i \leftarrow \Phi_i^\dagger \cdot L_{dif}$
    $M_b^i \leftarrow M_R^i \cdot \Phi_i - mean(\Phi_i)$

    //Update shape estimates
    $L_{i+1} \leftarrow L_i + (M_R^i \cdot \Phi_i + M_b^i)$
end for
```

Figure 5: Pseudo-code for the Supervised Descent Method training process.

of direclty regressing the shape. Also, the use of adaptive SIFT window sizes and the addition of a centroid fit selection approach further increases the robustness of the method.

3.2.1 Parametric approach

Instead of directly fitting the geometry to the image, an *Active Shape Model* (ASM) [13] is created, optimizing the parameters of the model along with the displacement, rotation and scaling parameters. Because of how ASM works, this reduces the total amount of parameters to optimize, while still explaining most of the shape variability.

Active Shape Model

An *Active Shape Model* (ASM) is a method based on *Principal Component Analysis* (PCA) in order to find a new coordinate system where the faces space can be explained with a lower dimensionality. Each dimension of this new space is uncorrelated with the rest, and follows the directions of maximum variance of the data.

In order to compute the PCA of a dataset with n variables and m instances, first the empirical mean for each variable, the mean vector B_0, is extracted at each instance, resulting on a matrix $A^{\langle n \times m \rangle}$ containing the variations relative to the empirical mean of each variable. Afterwards, the covariance matrix $V^{\langle n \times n \rangle}$ for the data is calculated, where each position represents the covariance between any two variables.

It is now possible to perform a matrix diagonalization in order to decompose the covariance matrix V into three components, as seen in Equation 9. This decomposition yields as a result a matrix $B^{\langle n \times n \rangle}$ of *eigenvectors*, being each column an eigenvector specifying the weights for the original dimensions linear combination defining a new dimension in the PCA space. Matrix $D^{\langle n \times n \rangle}$ is a diagonal matrix where each value corresponds to the *eigenvalues* of the eigenvectors in B.

$$V = B \cdot D \cdot B^T \tag{9}$$

By reordering the eigenvectors and eigenvalues in descending order according to the eivenvalues, the new dimensions are sorted from those explaining more variance of the data to those explaining less of it. The proportion of variance explained by each eigenvector is obtained by dividing each eigenvalue by their sum. After applying PCA to the geometry, it is now possible to reduce the data dimensionality by keeping only those eigenvectors whose eigenvalues sum ammount to the desired proportion of variance to be kept.

In order to apply PCA to explaining the variance of a shape, first the different shapes of the dataset must be aligned, making the landmarks coincide as much as possible without distorting the data. This implies applying a rotation, scaling and translation to each shape in the dataset, which is accomplished through a technique known as *Generalized Procrustes Analysis* (GPA) [2]. This is an iterative technique which at each iteration first computes the mean shape by averaging the current shape values. Afterwards, each shape is individually aligned to the canonical form of the mean by finding the best transform to match the shape to the mean. This process is repeated until a convergence criterion is met, moment in which the shapes are aligned. The final alignment mean shape obtained at the last alignment step is then subtracted from the aligned data in order to perform the PCA.

By keeping only a subset of the dimensions on the new shape space (those explaining more variance), it is possible to explain most of the data variation with a much smaller set of variables. For instance, in the case of the LFPW dataset, which consists on face images obtained from the internet and labeled with 21 2D facial landmarks, the shape variations are completely explained with $2 \cdot 21 = 42$ parameters. The corresponding ASM that explains 95% of the variance, though, consists of only 21 parameters. A shape L following the model restrictions can now be generated by assigning values to the m values of the weight vector b and adding a linear combination of the weighted eigenvectors B_i to the mean B_0, as shown in Equation 10.

$$L = B_0 + \sum_{i=1}^{m} b_i \cdot B_i \tag{10}$$

The most relevant eigenvectors or bases of an ASM, which explain the main modes of variation of the shape, can explain either a change in the perspective from which the shape is observed, or by an intrinsic variation of the shape. This is illustrated in Figure 6, where the deformation of the mean 2D face shape is shown when the weight assigned to a specific eigenvector is modified.

When fitting an ASM, it is also possible to restrict the fitted shape to a valid model shape. This is done by defining an hyper-dimensional oval, where the value for each dimension is restricted to lie within a given number of standard deviations from the zero mean. The standard deviation for each eigenvector is defined by the squared root of its eigenvalue. When a vector of weights b lies outside of that oval, the vector is rescaled to that maximum length. In the proposed approach, this technique is used to rescale the vector to a maximum length of $3 \cdot \sigma$ from the model mean after each cascade step.

Fitting parameters

Along with the ASM weights b required to define the shape, there are five more parameters required to position it into the image, giving a final parameter vector $p = \langle b, s_x, s_y, d_x, d_y, \theta \rangle$. These are the vertical and horizontal scaling parameters s_x and s_y, the shape translation parameters d_x and d_y, and a rotation angle θ. Taking these into account, the number of fitting parameters for a 2D shape amounts to $||p|| = m + 5$.

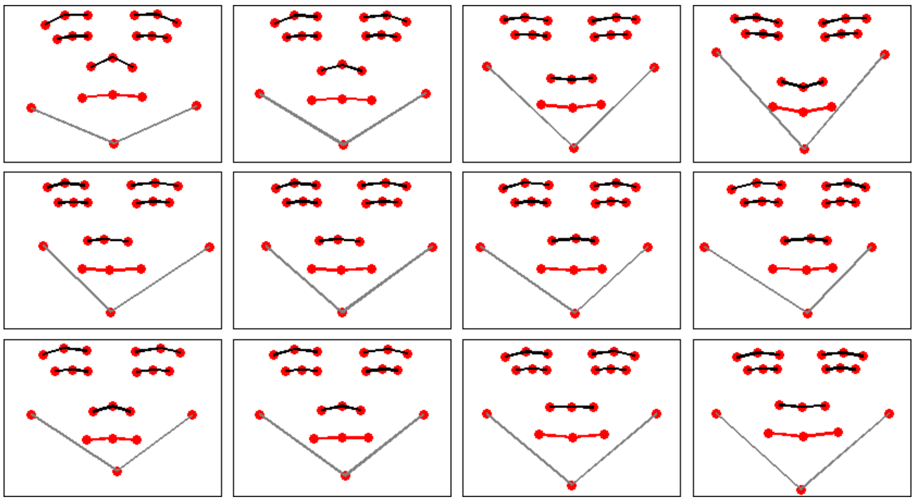


Figure 6: Three main modes of variation of the 2D ASM generated from a subset of 1572 images from the AFLW dataset. The rows correspond to the first, second and third modes of variation, while each column corresponds to a variation of -2, -1, 1 and 2 σ respectively. The first mode corresponds to a change in pitch, the second to a change in yaw, and the third to the degree to which the face is smiling.

Because of how the ASM works, adding more landmarks to the model does not imply a linear increase on the number of weights b required to define that shape in the ASM model. This means that while the cost of calculating the SIFT descriptors remains the same for both SDM and the proposed approach, the cost of performing the linear regression is much lower. Even though, in the new approach it is necessary to calculate the precise landmarks position from the fitting parameters before extracting the SIFT descriptors. The computational cost of the proposed approach is later compared with that of SDM.

Recovering the geometry

In order to obtain the SIFT descriptors at the corresponding landmark locations, it is necessary to transform the parameters vector into the corresponding geometry at each cascade step. This is done by first obtaining the unaligned shape $L'$from the ASM weights b, as shown in Equation 10, where m is the number of basis. Once the reconstructed shape is obtained, it is rotated, scaled and translated according to the rest of parameters of p. This is done through the Equation 11, where R is a rotation and scaling transform matrix, and T is a translation matrix.

$$L = R \cdot L' + T = \begin{pmatrix} s_x \cdot cos(\theta) & -s_x \cdot sin(\theta) \\ s_y \cdot sin(\theta) & s_y \cdot cos(\theta) \end{pmatrix} \cdot L' + \begin{pmatrix} d_x \\ d_y \end{pmatrix}^{\langle 1 \times n \rangle} \tag{11}$$

Computational complexity

In the case of SDM, the cost of performing the linear regression for n landmarks amounts to a matrix product of the linear regression matrix R_i of size $2n \times 128n$ and a descriptors matrix Φ_i of size $128n \times 1$, representing the descriptors subtracted from the image at the current landmark estimates. The complexity of this product operation equals $\Theta(2n \cdot 128n) = \Theta(256 \cdot n^2)$.

When using the new proposed technique, the problem is divided in two steps, first having to calculate the landmark locations from the fitting parameters, and afterwards using the SIFT descriptors to update the fitting parameters. The first step is performed by multiplying the matrix with the relevant eigenvectors, of size $2n \times m$, by the vector of bases, of size m, and afterwards applying a spatial transform to the landmarks in order to include the translation, rotation and scaling to the landmarks. The transform is done by multiplying a 3×3 matrix by the $3 \times n$ matrix representing the geometry. The second step is similar to that used by SDM, but updating the parameters vector p instead of the bases. This means the computational cost of the second step is of $\Theta(||p|| \cdot (n \cdot 128)) = \Theta(128 \cdot n \cdot ||p||)$, with the total cost of the proposed algorithm being $\Theta(3 \cdot 3 \cdot n + 2 \cdot n \cdot m + 128 \cdot n \cdot ||p||) \approx \Theta(9 \cdot n + 130 \cdot n \cdot m)$.

The computational complexity of SDM, thus, is of $\Theta(n^2)$, while that of the proposed algorithm is of $\Theta(n \cdot m)$. It is important to note that the complexity is only for the algorithm itself, without taking into account the computational complexity of calculating the SIFT descriptors, which is the same in both cases. Because of $m << n$, specially as n increases, it is shown that the proposed approach does not proportionally increase the computational cost of the algorithm, but rather makes it more efficient.

Advantages and inconvenients

The proposed change has some advantages, as well as some inconvenients compared to SDM. By one side, using a parametric approach helps other steps of the algorithm, since as it is shown in Section 3.2.2, by having the scaling parameters directly available it is possible to make the SIFT descriptors adaptive to the model scaling without increasing the cost of the algorithm. It also helps determining the head pose when the algorithm is adapted to work in a 3D space, as it is shown in Section 3.4.

A second advantage of the method is its ability to restrict the face geometry to the modes of variation observed in the training data. Since the weights b are scaled to fit within a radius of 3σ from the ASM mean shape, the ability of SDM to simultaneously fit all the landmarks by sharing information of their descriptors is further increased by using their current locations to force all points to take a valid shape.

The downside to the ability of the model to restrict the possible shapes is the inability to represent shape variations that are not observed in the training data and cannot be extrapolated as a linear combination of the available observations, but this usually is not a problem for big datasets with a wide range of shape variations, as is common in the on-the-wild datasets used in this work. Another problem with this approach is the appearance of rotation singularities. In the 2D case, for instance, it is the same to define a rotation angle of 0 or $2 \cdot \pi$. This can introduce problems for faces which are seen upside-down, but these are easily solved by the centroid fit selection approach explained in Section 3.2.3.

3.2.2 Adaptive SIFT window sizes

The SIFT descriptor used in SDM, as explained in Section 3.1, uses a fixed window size in order to extract the descriptor at each landmark. While this greatly increases the speed of subtracting the descriptors, not having to find the invariant scale of the location, it also restricts the algorithm to work at a given image size. Since the algorithm works by approximating the current descriptors vector Φ_i to the target Φ_*, the extracted descriptors must be of a similar size in all cases.

This implies that, by one side, an interpolation must be made before processing an image, scaling it to the desired size. Also, the face size must be previously known. These problems mean an increase in the required pre-processing, and could be easily avoided given a fast way to adapt the window size.

The proposed change is to use the mean $s_m = (s_x + s_y)/2$ of the scaling parameters introduced in Section 3.2.1 in order to adjust the SIFT window size. A given default window size of 32×32 is assigned to the model when the scaling parameter equals one, and it is updated such that $w_{size} = 32 \cdot s_m$ at each cascade step as the parameters adjust to ft the window. This approach also has the advantage of having equivalent descriptors for the same scale normalized error values, making it easier for the linear regressor to estimate the correct scale.

3.2.3 Centroid fit selection

While in the case of having a previous approximate estimate of the face rotation, a single initial shape estimate can in most cases converge to the correct face geometry, in cases where the face orientation is unknown it is useful to start with multiple shape initializations with different values of the rotation angle θ. This is useful, for instance, when performing face tracking on video sequences, where at the first frame multiple initializations are used, selecting the best after fitting them.

$$\begin{aligned} & d(L^i, L^j) = \sum_{p=1}^{n} \sqrt{(x_p^j - x_p^i)^2 + (y_p^j - y_p^i)^2} \\ & where\ L_p^i = \langle x_p^i, y_p^i \rangle \end{aligned} \tag{12}$$

In order to select the best fit between k initializations, a matrix $S^{\langle k \times k \rangle}$ of matching distances s calculated. A cell $S_{i,j}$ contains the distance between the fitted initializations L^i and L^j, defined as the sum of the euclidean distances $d(L^i, L^j)$ between matching landmarks, as shown in Equation 12, where n is the number of landmarks of the shape. Given the distances matrix, it is now possible to select the fit L^{min} by selecting the row/column with the minimum sum of distances.

3.3 Recovery of the 3D geometry from a 2D fit

The first approach to extending the proposed method to fitting the 3D geometry of a face is the use of an additional step after fitting the 2D geometry, which optimizes the parameters of a 3D ASM along with those of a restricted camera model in order to find the best 3D shape parameters and pose simultaneously.

3D Active Shape Model

A 3D ASM of the face geometry is generated by using the Facewarehouse [10] dataset. This dataset consists on a total of 3000 high resolution 3D facial range scans of 150 individuals aged 7-80 from different ethnicities displaying 20 different facial expressions.

In order to generate the 3D ASM, first the interest points for the fitting process are selected in order to match those available at the corresponding 2D dataset. Afterwards, the geometries are aligned by using an orthogonal *Generalized Procrustes Analysis*, only allowing for rotation, translation and an equal scaling in all three dimensions. Since the data is directly captured in 3D, there are no distortions caused by the head pose, and the shapes can be directly aligned to the canonical form.

Iterative alignment and 3D ASM weights selection

The process followed to adjust the 3D ASM to the 2D face landmarks is an iterative two-step process, where first the 3D rotation, translation and scaling are obtained, afterwards optimizing the 3D ASM weights to adjust the 3D shape orthographic projection to the 2D landmarks. These steps are repeated until the alignment error stops decreasing, as shown in the pseudo-code of Figure 7. Here, $\mathcal{B}_i^{3D}$ represents the matrix form of base B_i^{3D}, and is of size $\langle 3 \times n \rangle$. $\mathcal{B}_i^{2D}$ is the equivalent representation for the 2D bases B_i^{3D}.

Initially, the algorithm sets the 3D ASM deformation weights b to zeros, generating the mean face shape. This geometry L^{3D} is then used to find the best rotation and scaling matrix R and translation T matching the 2D rigid geometry L^{2D}. Then the iterative process starts, which consists on first updating the 3D shape weights b and afterwards re-aligning the 3D shape to the 2D geometry.

```
$m \leftarrow ||B^{3D}||$
$b \leftarrow 0^{m \times 1}$
$P \leftarrow [1, 0, 0; 0, 1, 0]$

$L^{3D} \leftarrow \mathcal{B}_0^{3D} + \sum_{i=1}^{m} b_i \cdot \mathcal{B}_i^{3D}$
$\langle R, T \rangle \leftarrow AlignShapes(L^{3D}, L^{2D})$

$L^{3D} \leftarrow R \cdot L^{3D} + T$
$err \leftarrow \sum_{i=1}^{n} \sqrt{(P \cdot L_{1,i}^{3D} - L_{1,i}^{2D})^2 + (P \cdot L_{2,i}^{3D} - L_{2,i}^{2D})^2}$

$\Delta_{err} \leftarrow -\infty$
while $\Delta_{err} < 0$ do
  // Project and align 3D ASM mean and bases
  $B_0^{2D} \leftarrow P \cdot (R \cdot \mathcal{B}^{3D} + T)$
  for all $\mathcal{B}_i^{3D} \in \mathcal{B}^{3D}$ do
    $\mathcal{B}_i^{2D} \leftarrow P \cdot R \cdot \mathcal{B}_i^{3D}$
  end for

  // Update 3D ASM weights
  $b \leftarrow (B_{1:m}^{2D})^{\dagger} \cdot (L^{2D} - B_0^{2D})$
  $L^{3D} \leftarrow \mathcal{B}_0^{3D} + \mathcal{B}_{1:m}^{3D} \cdot b_{1:m}$

  // Re-align 3D shape and get error
  $\langle R, T \rangle \leftarrow AlignShapes(L^{3D}, L^{2D})$
  $L^{3D} \leftarrow R \cdot L^{3D} + T$

  $\Delta_{err} \leftarrow \sum_{i=1}^{n} \sqrt{(P \cdot L_{1,i}^{3D} - L_{1,i}^{2D})^2 + (P \cdot L_{2,i}^{3D} - L_{2,i}^{2D})^2} - err$
  $err = err + \Delta_{err}$
end while
```

Figure 7: Pseudo-code for the 3D ASM alignment to a 2D shape.

In order to update the 3D shapes, first the 3D ASM mean $\mathcal{B}_0^{3D}$ is aligned to the shape by applying the transform R, translation T and finally performing an ortographical projection, which generates a 2D mean shape $\mathcal{B}_0^{2D}$. Each base $\mathcal{B}_i^{3D}$ is also transformed by using the rotation and scaling transform R and ortographically projected, but the translation is not applied. This is due to the ASM bases describing the modes of variation of the data after subtracting the mean, which implies the bases always remain centred at zero. This is shown in Equation 13.

$$\begin{aligned} L^{3D} &= R \cdot (\mathcal{B}_0^{3D} + \sum_{i=1}^{m} (b_i \cdot \mathcal{B}_i)) + T = R \cdot \mathcal{B}_0^{3D} + R \cdot \sum_{i=1}^{m} (b_i \cdot \mathcal{B}_i) + T \\ &= (R \cdot \mathcal{B}_0^{3D} + T) + \sum_{i=1}^{m} (R \cdot b_i \cdot \mathcal{B}_i) \end{aligned} \tag{13}$$

The weights for the 3D ASM model can now be found by finding the optimal weights of the projected 2D ASM model. In order to do so, the problem is formulated as a minimization problem and solved through least-squares. This is done by solving the equation $b = (B_{1:m}^{2D})^{\dagger} \cdot (L^{2D} - B_0^{2D})$, where m is the number of bases of the ASM. After computing the weights, the resulting unaligned 3D shape is re-calculated and re-aligned to the 2D data. Now the error of matching the aligned 3D shape to the 2D geometry is calculated as the sum of euclidean distances between the landmarks and compared to the previous error. The weight re-calculation and shape alignment steps are repeated until the error stops decreasing.

Rigid 3D and 2D shapes alignment

In the previous alignment method between a 3D ASM model and a 2D shape, it is required to find the best rotation, scaling and translation aligning a rigid 3D shape to the target 2D one, which is done through the method $AlignShapes(L^{3D}, L^{2D})$. What this method does is to find the best parameters for a scaled orthographic camera model known as the *restricted camera model* [18]. The particularity of this method is its use of a single scaling factor instead of scaling each dimension independently. This camera model is selected in order to prevent distortions of the 3D model, since when distortions are allowed, as in the case of the general orthogonal camera model, the scaling tends to absorb shape variations that should be explained by perspective changes and variations of the ASM weights.

In order to calculate the best translation, rotation and scaling to match the 3D shape to the 2D one, first the mean of each dimension is subtracted from both the 2D and 3D shapes, moving them to the coordinate origin. This results in the displacements d^{2D} and d^{3D}. Since in the case of aligning a 3D ASM it is already centred at the coordinate origin, the optimal translations for alignment corresponds to d^{2D}. The value assigned to the third dimension of the translation is irrelevant, since in an orthogonal camera model the projection is unaffected by the distance to the projection plane and the camera. Afterwards, least-squares is used on the centred data in order to find the best transform $X^{\langle 2\times 3\rangle}$ such that $L^{2D} = X^{\langle 2\times 3\rangle} \cdot L^{3D}$. This gives a transform that is likely to distort the landmarks, and projects them into the 2D plane. In order to solve these issues, the next step is to add an additional row defining the location of the points in the third dimension. This is done by calculating the cross product of the first two vectors, such that $X_{3,:} = X_{1,:} \times X_{2,:}$. This defines a third dimension is orthogonal to the first two ones.

Afterwards, RQ decomposition is applied to the X transform matrix. This transform decomposes the matrix such that $X = U \cdot V$, where V is a matrix where each row is orthogonal to the rest, and U is an upper triangular matrix. If the original X matrix had been a rotation matrix of orthogonal bases, U would now contain the positive scaling factor of each dimension in the diagonal, and zeros at the rest of positions. Now it is possible to enforce these constraints to the transform by forming a diagonal matrix W from the sings of the U diagonal and multiplying it with both the U and V matrices. This results in a non-scaled rotation matrix $R' = W \cdot V$ and a diagonal matrix $S = U \cdot W$ containing the scaling at each dimension. The final scaling for the *restricted camera model* is obtained by averaging the scaling factors of the two first dimensions, such that $s = (s_x \cdot s_y)/2$. Finally, the rotation with the factor scaling is created such that $R = s \cdot R'$.

3.4 Extension to direct 3D fitting

Another approach to obtain the 3D geometry of faces from grayscale images is the extension of the proposed 2D fitting approach to directly fitting 3D geometries. This approach is more direct than the previously proposed one, since it only requires a reformulation of the adjusted parameters and the extension of the technique used to transform the parameters vector into an aligned face geometry before extracting the SIFT descriptors. The same 3D ASM model seen for recovering the 3D geometry in Section 3.3 is now used during the image fitting stage.

Reformulating the parameters vector

During the explanation of the parametric approach for fitting 2D shapes in Section 3.2.1, a parameters vector $p = \langle b, s_x, s_y, d_x, d_y, \theta \rangle$ has been defined, which takes into account the ASM weights b, as well as two scaling parameters s_x and s_y, and two translation parameters d_x, d_y. In order to adapt this representation to the 3D case, some changes are made to the formulation. By one side, the two scaling parameters are merged into a single s parameter, in order to prevent different varying scaling factors to absorb pose and shape variations. Also, two additional angles γ and η are added in order to represent the three rotation angles corresponding to roll, pitch and yaw.

$$p = \langle b, s, d_x, d_y, \theta, \gamma, \eta \rangle \tag{14}$$

The resulting feature vector is the one seen in Equation 14. Note that since an orthographic projection is used, there is no need to add a parameter d_z for the displacement in the third dimension. Also, the weights vector b now corresponds to the weights of the 3D ASM.

Recovering the geometry

It is also necessary to reformulate Equation 11 which aligns the recovered shape $L^{3D'}$ to the image by using the rotation, scaling and translation parameters. Since the points to be recovered must be in a 2D space in order to extract the SIFT descriptors from the image, an orthogonal projection is performed along with the 3D transform. The used method is shown in Equation 15.

$$\begin{aligned} L^{2D} &= [P \cdot (s \cdot R_\eta \cdot R_\gamma \cdot R_\theta)] \cdot L^{3D'} + T = \left[\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \cdot R'\right] \cdot L^{3D'} + T = R \cdot L^{3D'} + T \\ where\ R &= s \cdot \begin{pmatrix} c(\eta)c(\theta) + s(\eta)s(\gamma)s(\theta) & c(\eta)s(\theta) - s(\eta)s(\gamma)c(\theta) & s(\eta)c(\gamma) \\ -c(\gamma)s(\theta) & c(\gamma)c(\theta) & s(\gamma) \end{pmatrix} \end{aligned} \tag{15}$$

Here R' is the composite rotation matrix along roll (θ), pitch (γ) and yaw (η) scaled by a factor of s. This rotation matrix is projected in order to obtain a compact transform matrix R before applying it to the $L^{3D'}$ landmarks. By directly computing R, the calculation of the third dimension of the rotation is avoided. After projecting the geometry, the 2D translation is added to the transformed landmarks by summing $T = \left[(d_x, d_y)^{\langle n \times 1 \rangle}\right]^T$.

4 Experimental methodology

In this section the experiments performed to assess the accuracy of the different algorithms, their execution time and the used datasets are explained. The results of these experiments are shown in Section 5. First the three datasets used for the experiments are explained in Section 4.1, two of which are used to check the algorithm geometry fitting accuracy and the other for the head pose recovery. Afterwards, the specific experiments for measuring the geometry fitting and pose recovery accuracies are explained in Sections 4.2-4.3. A qualitative evaluation of the methods results is also pefromed.

4.1 Used data

Three datasets are used to perform the experiments. The *Annotated Facial Landmarks in the Wild* (AFLW) [24] and *Labeled Face Parts in the Wild* (LFPW) [3] are two *in the wild* datasets containing face images obtained from the internet. These are labelled with a set of landmarks at specific locations of the face. The third is the *Pointing '04* dataset [17], which is labelled only with the pose of the head, and is used to evaluate the head pose recovery performance for the two head pose recovery proposals. These datasets are now explained in detail.

Annotated Facial Landmarks in the Wild

Annotated Facial Landmarks in the Wild (AFLW) [24] is a dataset containing 25993 images obtained from the internet. The images display a wide variety of face poses for people of different ethnicities under different illumination and image quality conditions. While 21 landmarks are defined, these are only defined when visible, which makes many of the images not usable to train the algorithm. Only 2359 images are kept after discarding those with missing values for the landmark locations.

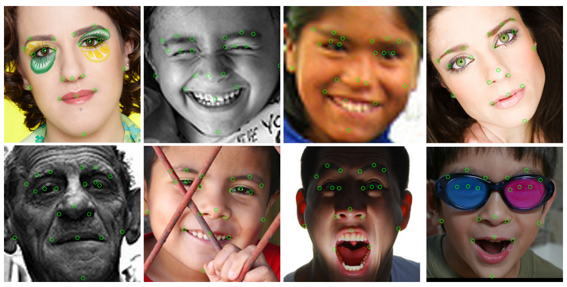


Figure 8: Image examples from the AFLW dataset. The dataset contains both colour and grayscale images of people from different ethnicities and ages, labelled with the 21 facial landmarks shown im the image.

In Figure 8 some examples of face images drawn from the AFLW dataset are shown. The dataset includes images at various resolutions, qualities and sizes. Furthermore, the pictures are from people of different ethnicities and ages, with various poses diverging from the frontal view in roll, pitch and yaw.

Labeled Face Parts in the Wild

Labeled Face Parts in the Wild (LFPW) [3] is a dataset of face images labelled with 35 landmarks around the face. Because of the dataset being downloaded as a list of URL addresses to images instead of it directly containing the images, these are slowly disappearing from the Internet, with the size of the dataset decreasing over time. While the original dataset consisted on a total of 1430 images, only 836 were still available when

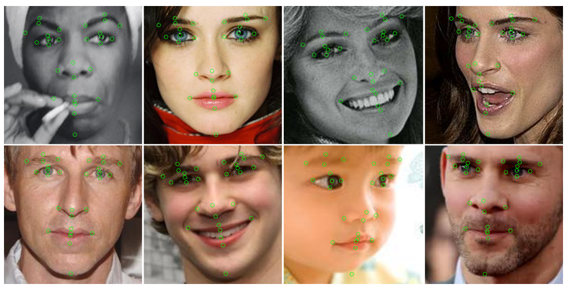


Figure 9: Image examples from the LFPW dataset. The dataset contains both colour and grayscale images of people from different ethnicities and ages, labelled with the 29 facial landmarks shown in the image.

downloaded for this work. In this dataset, the hidden landmarks are also labelled at their estimated location, with an extra parameter specifying whether each landmark is occluded or not. Because of the 6 landmarks corresponding to the ears being imprecisely annotated in most of the images, these are discarded, using the remaining 29 landmarks.

In Figure 9 some examples of face images drawn from LFPW dataset are shown. The dataset contains images at various resolutions, qualities and sizes, as in the case of AFLW, but the range of pose variations is wider. In LFPW, the pitch and yaw pose rotation angles of the faces range from −90 to +90 degrees *wrt.* the frontal view, although extreme rotations have a very low occurrence.

Pointing '04

Pointing '04 [17] is a dataset of grayscale face images taken in a controlled environment where each face image is labelled with the *pitch* and *yaw* pose angles. This dataset does not contain a ground truth for the face landmarks, and thus cannot be used to train any of the proposed algorithms. Instead, it is used to evaluate the accuracy of the pose recovery part of the algorithm.

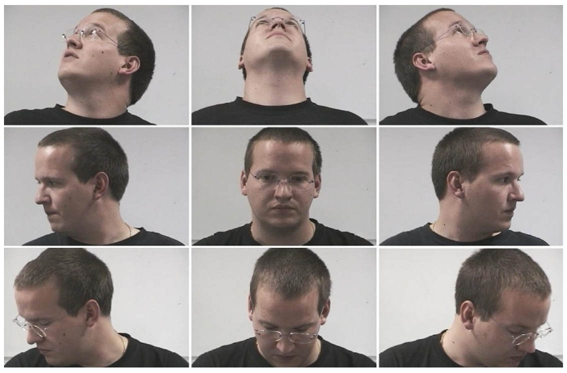


Figure 10: Image examples from the Pointing '04 *dataset. The* yaw *varies from the left to the right, while the change in pitch varies from the top to the bottom.*

The pose variations range from −90 to +90 degrees *wrt.* the frontal view for both *pitch* and *yaw*, with variations of 15 degrees between images. The dataset contains 15 individuals, with two series recorded for each one and each series containing 94 different poses. This amounts to a total of $15 \cdot 2 \cdot 94 = 2820$ images. Some example poses are shown for one of the individuals in Figure 10.

4.2 Geometry fitting experiments

In order to measure the accuracy of the geometry fitting, as well as to compare the accuracies between the original SDM algorithm, the proposed parametric approach and both proposed algorithms for 3D fitting, a series of experiments are performed. The SDM algorithm is implemented from the paper, since the authors made availabe only a testing implementation, from which it is not possible to perform training.

First the parameters of the algorithms are set through a grid search, afterwards using the best parameter combination found in order to compare the algorithms mean errors and Confidence Intervals (CI), determining the most accurate approach. In order to do so, for each parameter combination a ten-fold cross-validation is performed over the AFLW and LFPW datasets separately, selecting the parameters minimizing the average error for each model and dataset. The considered parameters are:

- Number of data augmentations
- Number of cascade steps

- Number of initializations

The *number of data augmentations* corresponds to the number of initial fits with random scaling, rotation and displacement parameters generated for each training image. The *number of training instances* equals $||L^{aug}|| = ||L^{train}|| \cdot N_{aug}$. On the other hand, the *number of cascade steps* corresponds to the number of linear regressors applied in series to find the final image face alignment.

Finally, the *number of validation initializations* is selected, which is a validation parameter that does not affect the training process. For this reason, it is not considered when optimizing the model parameters through the search grid, and is independently optimized over the validation partitions once the final model is selected. This parameter determines the number of initial samples at different rotation angles evenly distributed between -90 and $+90$ that are used to fit the model to an image. When performing the grid search on the original algorithm, 10 testing initializations are considered.

With the final models selected, the validation results of the best parameter combinations at each model are used to evaluate the error distribution through the *Cumulative Error Distribution* (CED) curve, also checking the qualitative characteristics of the wrongly fitted images. Finally, the runtime of the different approaches are compared between them with the selected optimal parameters. In order to do so, all images are processed 32 times and their average runtime is obtained in order to account for runtime variations between executions, afterwards calculating the average image processing time of a single image.

Ten-fold cross-validation

The ten-fold cross-validation is a common approach to evaluate the performance of machine learning algorithms. This methodology uses most of the data for training at each iteration, while still being able to use all of the data to evaluate the performance at the end.

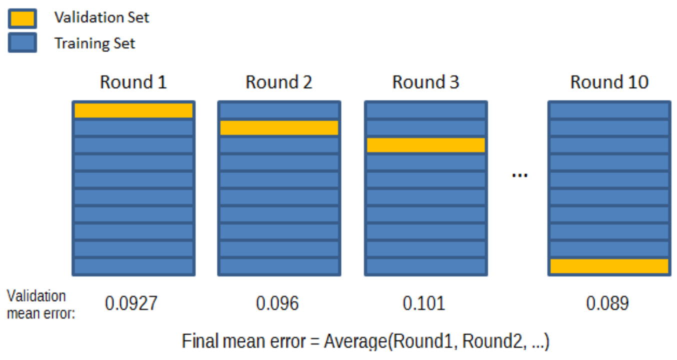


Figure 11: Example of a 10-fold cross-validation data partition. At each fold, one of the partitions is selected for validation and the rest are used to train the algorithm. Finally the results of each fold validation are merged, obtaining the validation results over the whole dataset.

The methodology consists on dividing the data into n equal-sized partitions, in the case of 10-fold cross-validation into 10 partitions, and then performing the algorithm training and validation n times. At each iteration, a different partition is selected for validation, using the rest for training. After validating the algorithm over all of the partitions, the validation results can be merged. This process is illustrated in Figure 11.

While this method is more expensive computationally, since the algorithm must be trained n times, it allows for the use of all the data for validation and most of it for training at each iteration. When evaluating the methods proposed in this work, this gives as a result an error measure with a lower bias and a narrower variance.

Error measure

The error measure used for the geometry fitting is the Normalized Mean Euclidean Distance (NMED) seen in Equation 16. In this equation l is the number of landmarks and L_i are the coordinates of a specific landmark. This measure uses the mean euclidean distance $d_2(x, y)$ between the fitted geometry L and their corresponding ground truth locations in L^*, normalized by the intra-ocular distance of the fitted face in the ground truth.

$$E_{norm} = \frac{\sum_{i=1}^{l} d_2(L_i, L_i^*)}{l \cdot d_2(L_{eyer}^*, L_{eyel}^*)} \tag{16}$$

3D ground truth

In order to measure the geometry fitting accuracies of the two proposed extensions for the 3D geometry fitting, first the 3D ground truth of the landmarks in both the AFLW and LFPW datasets are required. Since both datasets are labelled with only the 2D landmarks, the third dimension is extrapolated by using the iterative alignment algorithm between a 3D ASM and a 2D rigid geometry described in Section 3.3. In this case, the 3D ASM corresponding to the set of landmarks of the specific dataset is aligned against the ground truth 2D landmarks.

This gives as a result a best-fit between the 3D ASM model and the 2D ground truth, but with the new 3D landmarks not matching exactly tose of the 2D shape. In order to solve this problem, the original x_i and y_i coordinates are kept, only adding the new estimated z_i coordinates to each landmark L_i.

4.3 Pose recovery experiments

The pose recovery is evaluated by using the best models obtained for the two 3D fitting algorithms, namely the *3D alignment* and the *3D regression* methods, over the *Pointing '04* dataset. In order to evaluate and compare the pose recovery accuracies, first the mean fitting error over all the poses and its CI is obtained for the *pitch* and *yaw* for each method, performing a Wilkoxon rank sum statistical test to check whether one method can be considered better than the other in determining a specific pose angle. Afterwards, two surface plots are generated for each method, one for the *pitch* mean error and the other for the *yaw* mean error relative to the image ground truth pitch and yaw. This representation is useful for visualizing how the error is distributed along the different poses.

The error measure used for these tests is the error rate $E_\theta = (\theta - \theta*)/\pi$, measured as the angular difference between the measured angle θ and the ground truth angle θ_* divided by the maximum possible error, which is of π radians.

5 Results

In this section the results of the experiments previously explained in Section 4 are given, first analysing the results for the parameter selection in both AFLW and LFPW datasets for the 2D and 3D methods in Section 5.1. Once the best parameters are found for each method, a comparison between the two 2D geometry fitting methods is made in Section 5.2, with the 3D methods being compared in Section 5.3. In both cases the comparison is made for the geometry fitting accuracies and the execution times. The 2D and 3D approaches are analysed separately because of the error measure being defined as the euclidean distance in a 2D space in the first case, and in a 3D space in the second, which makes it not possible to directly compare the results between 2D and 3D approaches.

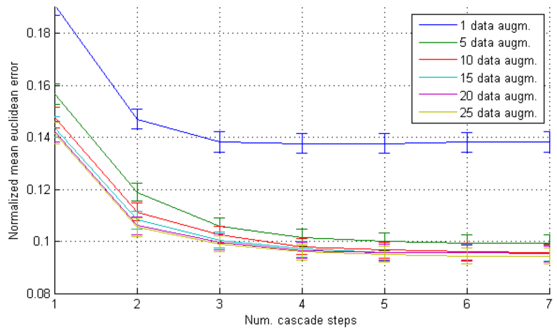
Afterwards, the head pose recovery accuracy of the 3D fitting methods is evaluated in Section 5.4. Finally, a qualitative evaluation of the results for both 2D and 3D approaches is performed in Section 5.5.

5.1 Parameter selection

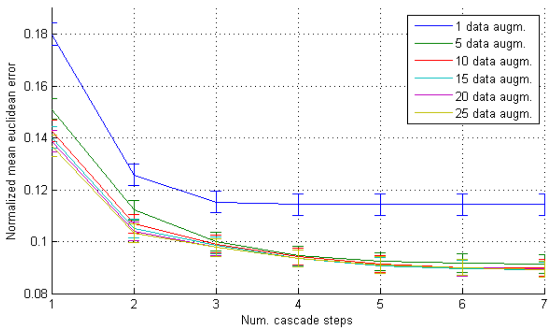
The parameter selection for both 2D approaches (*Supervised Descent Method* and *Parametric*) and 3D approaches (*3D alignment* and *3D regression*) is performed in two steps. First the number of data augmentations and cascade steps are selected through a grid search, since these are correlated, afterwards selecting the number of test initializations given the trained model. In order to perform both tasks, 10-fold cross-validation is used for each parameter combination, obtaining a validation error.

2D geometry fitting methods

When selecting the *data augmentations* and *cascade steps* parameters over the AFLW dataset, it is shown in Figure 12 that the error decreases for any given number of time steps as the number of data boosts increases, but the reduction of the error is very small when the data augmentation is bigger than 10-fold. In the same fashion, as the number of cascade steps increases, the error diminishes regardless of the number of data augmentations, but it tends to converge after about 5 cascade steps. As the number of augmentations increases, it takes the algorithm more cascade steps to reach convergence, allowing for the error to be further reduced.



(a) Supervised Descent Method



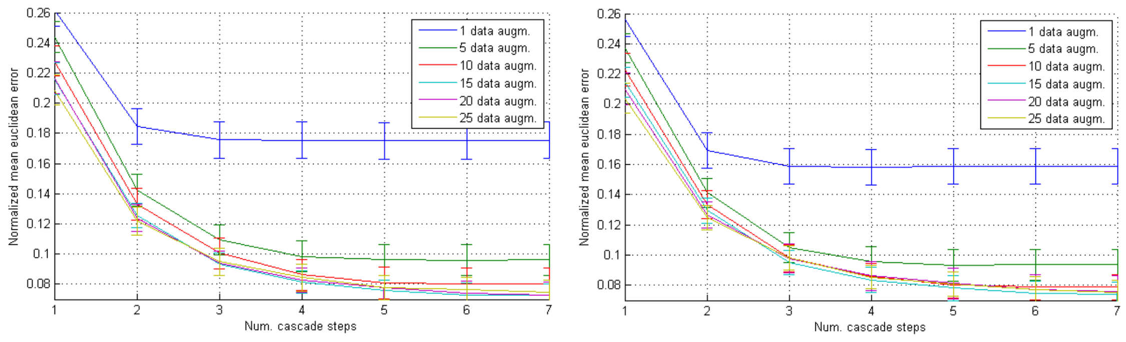
(b) Parametric approach

Figure 12: Grid search results for the selection of the ***data augmentations*** *and* ***cascade steps*** *parameters for both the* ***Supervised Descent Method*** *(a) and* ***Parametric*** *(b) algorithms when analysing the AFLW dataset. The parameter errors are given by the average error and its 95% CI.*

In order to select the optimal parameters for the algorithm, it is useful to look at the precise error values for the grid search, which can be found in Appendix A in the case of the AFLW dataset. From these tables, it can be seen that in the case of SDM the best parameter values are **6**

cascade steps and **25 data augmentations**. For the Parametric approach, the best values are found for **6 cascade steps** and **15 data augmentations**. It is important to note that while the error stabilizes past this point as the number of data augmentations increases, in the case of the Parametric approach the error keeps decreasing as more cascade steps are added, but the decrease is very slow and it is not worth the extra computational time. The reason why the error keeps decreasing in the Parametric approach while it doesn't in the original SDM method is explained later in this section.

In the case of the LFPW dataset, as it can be seen in Figure 13, the *data augmentations* parameter has a greater impact on the overall accuracy of the algorithm. This is due to the LFPW dataset being smaller (it contains 836 images compared to the 2359 images of AFLW), which makes the use of boosting more critical in order to consider a wider range of variations between the initial and target poses.



(a) Supervised Descent Method *(b) Parametric approach*

Figure 13: Grid search results for the selection of the ***data augmentations*** *and* ***cascade steps*** *parameters for both the* ***Supervised Descent Method*** *(a) and* ***Parametric*** *(b) algorithms when analysing the LFPW dataset. The parameter errors are given by the average error and its 95% CI.*

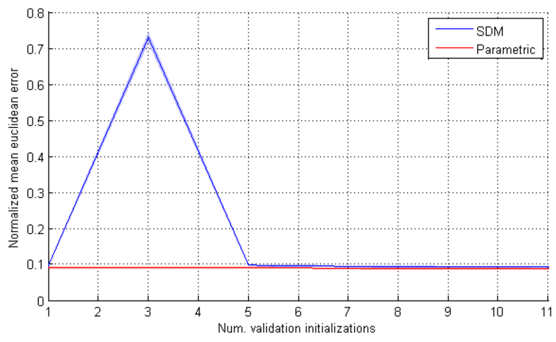
By looking at the precise error values of the grid search in Appendix B, it can be seen that for the LFPW dataset the best training parameters are **6 cascade steps** and **15 data augmentations** for both SDM and the Parametric approach. In both cases the error is still reduced as the number of cascade steps increases, but the reduction is very small and is not worth the extra computational time.

For both datasets, the proposed *Parametric* approach tends to reduce the error both faster and for a bigger number of cascade steps compared to SDM. This is due to the optimization of the pose parameters along with the weights of an ASM instead of directly optimizing the shape. This parametric representation compacts the shape representation to a smaller number of parameters, which in turn means the number of weights at each linear regressor is smaller. Since a smaller number of weights has to be optimized with the same available training data, the algorithm has a greater tendency to generalization, preventing the algorithm from over-fitting.

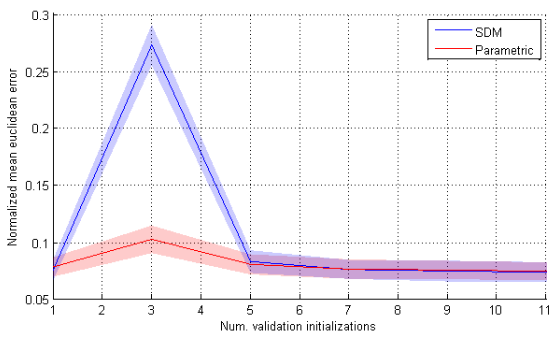
With the optimal parameters for the model training selected, it is now possible to select the optimal value for the number of initializations. The average errors of each final model relative to the number of validation initializations are shown in Figure 14 for both the AFLW and LFPW datasets.

It can be seen from these plots that in both datasets the proposed *Parametric* approach is less sensible to using a small number of initializations, with the original SDM algorithm giving high error values when using a small number of them. This implies that the extreme rotations of -90 and +90 degrees found when only 3 initializations are used, the SDM approach tends to converge to local minima far away from the real face pose, with the parametric approach partially solving this problem.

The precise mean error values and CI for the *initializations* parameter selection are found in



(a) *AFLW dataset*



(b) *LFPW dataset*

Figure 14: Average errors with their CI relative to the number of data initializations for the ***SDM*** *and* ***Parametric*** *approaches over both the AFLW (a) and LFPW (b) datasets.*

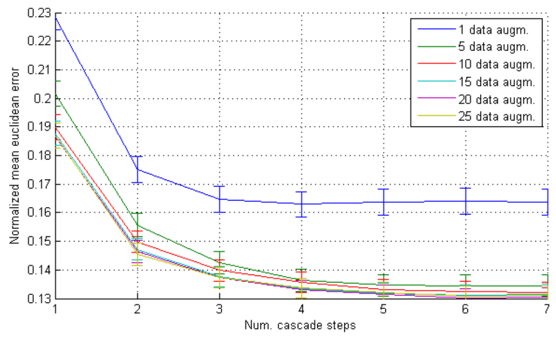
Appendix C. From these values, it can be seen that at about 5 initializations the mean error of all the algorithms starts to fall below the mean error of using a single initialization, but decrease is slow, with a small improvement with respect to using a single initialization. The extra computational time is not worth the use of many initializations, since the time required for processing an image is multiplied by a factor i, the number of initializations.

The selected parameters for each 2D geometry fitting algorithm are the following ones:

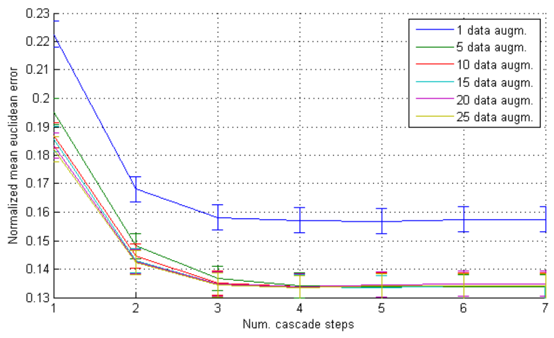
	AFLW		LFPW	
	SDM	Parametric	SDM	Parametric
Data augmentations	6	6	6	6
Cascade steps	25	15	15	15
Initializations	1	1	1	1

3D geometry fitting methods

In the case of parameter selection for the 3D geometry fitting methods, the same trend seen for the 2D geometry fitting algorithms is observed for the *cascade steps* and *data augmentations* parameters. As the number of data boosts increases, the mean error decreases regardless of the number of cascade steps, with the decrease being very small when the *data augmentation* is greater than 10-fold. The error also takes longer to converge as the number of data augmentations increases.



(a) *3D alignment*



(b) *3D regression*

Figure 15: Grid search results for the selection of the ***data augmentations*** *and* ***cascade steps*** *parameters for both the* ***3D alignment*** *(a) and* ***3D regression*** *(b) algorithms when analysing the AFLW dataset. The parameter errors are given by the average error and its 95% CI.*

When observing the trend of the fitting error for both the AFLW and LFPW dataset in Figures 15-16 some differences are observed between the two considered approaches. Firstly, the *3D alignment* approach follows a similar trend to the 2D approaches for the convergence of 2D values, converging at lower error values as the number of data boosts increases. In the case of *3D regression*, though, past a certain number of data boosts the average errors converge at about the same value as the number of cascade steps increases.

Another difference is the rate of the error descent. The *3D alignment* approach has a smoother descent curve relative to the number of cascade steps. *3D regression*, on the other hand, shows a stepper descent curve, converging faster initially. This means a more accurate result can be obtained with *3D regression* with a fast approach using only the first cascade steps.

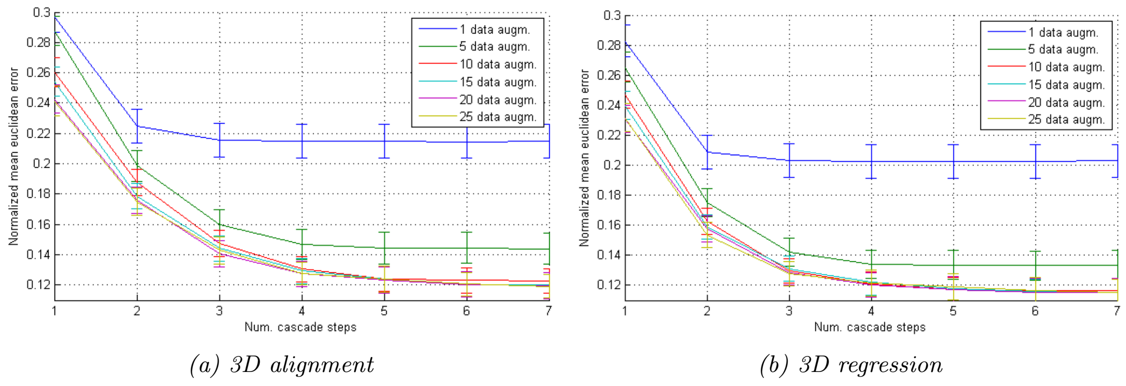


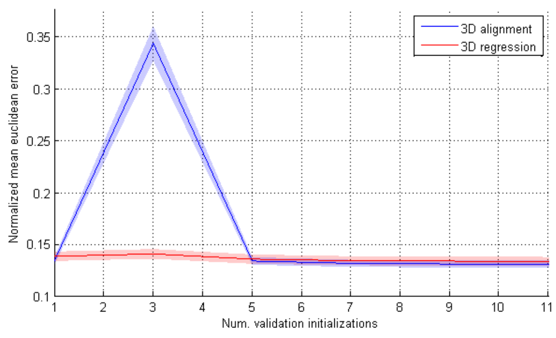
*Figure 16: Grid search results for the selection of the **data augmentations** and **cascade steps** parameters for both the **3D alignment** (a) and **3D regression** (b) algorithms when analysing the LFPW dataset. The parameter errors are given by the average error and its 95% CI.*

In order to select the best parameter values for both algorithms over the AFLW dataset, the precise values for the mean error are found in Appendix A. In the case of *3D alignment*, the lowest average error is found for **6 cascade steps** and **20 data augmentations**. For *3D regression*, the lowest average error is found for **5 cascade steps** and **15 data augmentations**, converging faster than *3D alignment*.

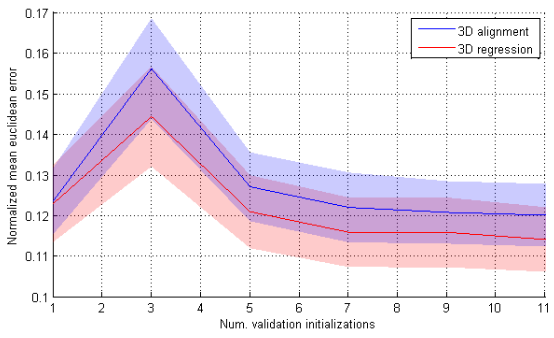
In the case of the LFPW dataset, the mean errors of the grid search are found in Appendix B. In this dataset, the selected parameters for both approaches are **6 cascade steps** and **20 data augmentations**. The error keeps decreasing in both cases if more cascade steps are used, but at a small rate.

When comparing the error relative to the *initializations* parameter for both the *3D alignment* and *3D regression* methods, as shown in Figure 17, a trend similar to the one found between the 2D methods is observed. For a small number of initializations, the error first increases, reducing again at about 5 initializations, point at whitch the error decreases below the one found for a single initialization. This trend is less marked in the case of the *3D regression* approach.

The precise mean error values and their CI for the *initializations* parameter selection are found in Appendix C. From these values it can be seen that the error decreases as the number of initializations increases past 5, but it does not get significantly lower than the mean error obtained for a single initialization. The only exception is for the *3D regression* algorithm over the LFPW dataset when 11 or more initializations are used, point at which the error is better than for a single initialization with a 95% Confidence Interval. Even though, because the error difference is small and the computational cost in order to achieve this small reduction of the average error would be 11-fold that of using a single initialization, the *initializations* parameter is set to 1 for all of the algorithms and datasets.



(a) *AFLW dataset*



(b) *LFPW dataset*

Figure 17: Average errors with their CI relative to the number of data initializations for the ***3D alignment*** *and* ***3D regression*** *approaches over both the AFLW (a) and LFPW (b) datasets.*

The selected parameters for each 3D geometry fitting algorithm are the following ones:

	AFLW		LFPW	
	3D alignment	3D regression	3D alignment	3D regression
Data augmentations	6	5	6	6
Cascade steps	20	15	20	20
Initializations	1	1	1	1

5.2 Accuracies comparison between 2D methods

When comparing the mean fitting error of the 2D shape alignment methods over the AFLW dataset, as it is shown in Table 3, the *Parametric* approach obtains better results than those of SDM, with an error of 0.0928 ± 0.0031 compared to an error of 0.0973 ± 0.0032 for SDM. The Wilkoxon rank sum test shows that this difference is significant, with a probability of only $1.7 \cdot 10^{-10}$ of the obtained sample distributions belonging to the same population distribution.

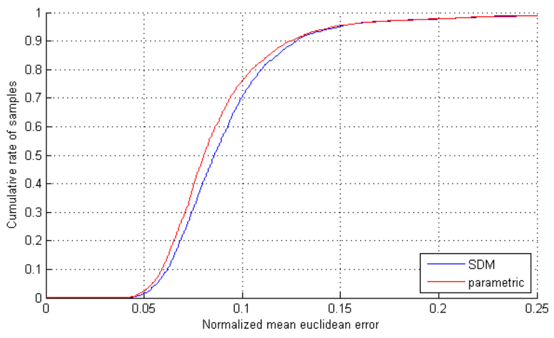
On the other hand, in the case of the LFPW dataset, SDM has slightly better results, with a mean error of 0.0753 ± 0.0090 compared to a mean error of 0.0784 ± 0.0085 for the parametric approach. In this case, though, the difference is not significant, with a probability of 0.2671 of both sample distributions belonging to the same population distribution.

	SDM	Parametric	p-value	Independence
AFLW	0.0973 ± 0.0032	0.0928 ± 0.0031	$1.7 \cdot 10^{-10}$	yes
LFPW	0.0763 ± 0.0090	0.0774 ± 0.0085	0.2671	no

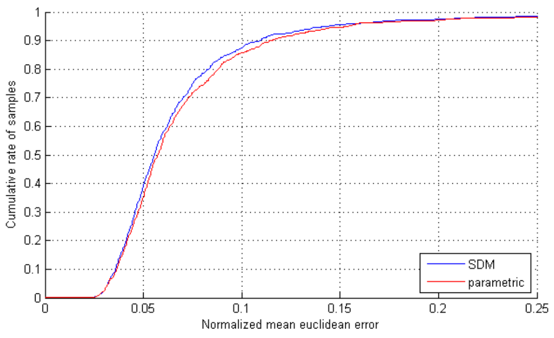
Table 3: Comparison between the average errors of the Supervised Descent Method *and the* Parametric *approaches over the AFLW and LFPW datasets. The given* p-values *correspond to the probability of the error samples of both methods belonging to the same error distribution, with the independence field specifying if both samples belong to different distributions with a 95% confidence.*

The *Cumulative Error Distribution* (CED) curves for the SDM and *Parametric* approaches over both datasets are shown in Figure 18. These plots show that both SDM and the Parametric approach obtain the same accuracies in the best and worst-case scenarios, with a NMED error of about 0.045 in the best case in the AFLW dataset, and of about 0.03 in the LFPW dataset. 95% of the images are detected with an error lower than 0.15 for both datasets and algorithms.

In the case of the AFLW dataset, where the Parametric approach has been shown to have a lower average error which is statistically significant, the CED curve shows that this improve-



(a) AFLW dataset



(b) LFPW dataset

Figure 18: CED curves for the Supervised Descent Method *and* Parametric *approaches over the AFLW (a) and LFPW (b) datasets.*

ment comes from a more accurate fitting of the correctly fitted faces, with the new approach not increasing the ratio of correct fits.

When evaluating the runtime of each algorithm, the following average run-times are obtained for each image at each dataset:

	AFLW		LFPW	
	SDM	Parametric	SDM	Parametric
Average error	0.0973 ± 0.0032	0.0928 ± 0.0031	0.0763 ± 0.0090	0.0774 ± 0.0085
Temporal cost	0.0133	0.0135	0.0200	0.0200

Table 4: Average runtime in seconds of the Supervised Descent Method *and* Parametric *approaches over the AFLW and LFPW datasets.*

From these results can be seen that the proposed *Parametric* approach obtains approximately the same accuracies, or slightly better in the case of the AFLW dataset, while maintaining the same average runtime. While this new approach does not obtain great improvements on the algorithm accuracy, it does provide a reformulation of the original SDM approach which allows for the implementation of a 3D geometry recovery algorithm which is much faster than the conventional two-step approach, as seen in Section 5.3.

5.3 Accuracies comparison between 3D methods

The obtained average errors for the *3D alignment* and *3D regression* methods are shown in Figure 5, where a Wilkoxon rank sum test is used to compare the sample error distributions of both methods at each dataset. In the case of the AFLW dataset, a statistically significant difference has been found for the error distributions with a p-value of 0.0134. The *3D alignment* method obtains a slightly lower average NMED error of 0.1340 ± 0.0036 compared to an average NMED error of 0.1387 ± 0.0045.

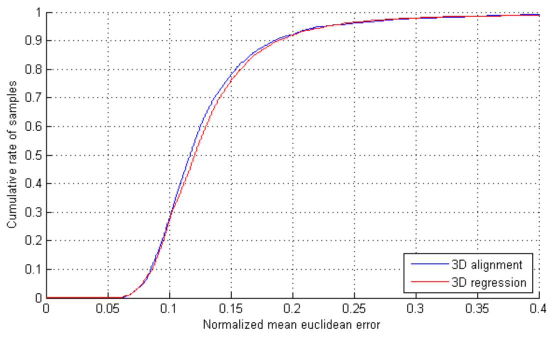
In the case of the LFPW dataset, there is no statistically significant difference between both algorithms average errors, with a probability of 0.3412 of both error sample distributions belonging to the same population distribution. Over this dataset, the *3D alignment* method obtains an average NMED error of 0.1235 ± 0.0081 compared to an average NMED error of 0.1229 ± 0.0094 for the *3D regression* method.

The CED curves of both approaches over each of the datasets are shown in Figure 19. Both algorithms align 95% of the samples with a NMED lower than 0.23, and have a minimum error of about 0.065 for the best fit images in the case of the AFLW dataset. For the LFPW dataset, both algorithms obtain a minimum error of 0.05. In the case of the AFLW dataset, where the *3D*

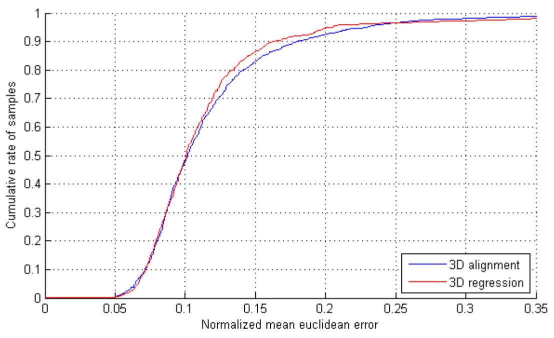
	3D alignment	3D regression	p-value	Independence
AFLW	0.1340 ± 0.0036	0.1387 ± 0.0045	0.0134	yes
LFPW	0.1235 ± 0.0081	0.1229 ± 0.0094	0.3412	no

Table 5: Comparison between the average errors of the 3D alignment *and the* 3D regression *approaches over the AFLW and LFPW datasets. The given* p-values *correspond to the probability of the error samples of both methods belonging to the same error distribution, with the independence field specifying if both samples belong to different distributions with a 95% confidence.*

alignment approach obtains a statistically significant smaller error, the curve shows that it is due to a small increase in the accuracy of the correctly fit faces, but no new faces are detected.



(a) AFLW dataset



(b) LFPW dataset

Figure 19: CED curves for the 3D alignment *and* 3D regression *approaches over the AFLW (a) and LFPW (b) datasets.*

The temporal cost of the algorithms, on the other hand, are shown in Table 6. In the case of the AFLW dataset, the *3D regression* approach has an average runtime of 0.0113 s, with the *3D alignment* approach taking 0.0268 s, 2.37 times more. A similar ratio is found for the LFPW dataset, with the *3D regression* approach running for 0.0163 s seconds on average, while the *3D alignment* approach takes 0.0338 s, 2.07 times more.

	AFLW		LFPW	
	3D alignment	3D regression	3D alignment	3D regression
Average error	0.1340 ± 0.0036	0.1387 ± 0.0045	0.1235 ± 0.0081	0.1229 ± 0.0094
Temporal cost	0.0268	0.0113	0.0338	0.0163

Table 6: Average runtime in seconds of the 3D alignment *and* 3D regression *approaches over the AFLW and LFPW datasets.*

The difference in the runtime rate of the algorithm is due to the increase in the cost of calculating the SIFT features in both algorithms. In the AFLW dataset 21 facial landmarks are taken into account, while in the case of the LFPW dataset 29 are used. As the number of landmarks increases, this common step in both algorithms requires more time, while the step performing 3D ASM alignment to the 2D landmarks in the *3D alignment* approach remains approximately constant. Even though, by directly avoiding the need of performing this additional step in *3D regression*, this second method will always be faster.

5.4 Pose recovery comparison

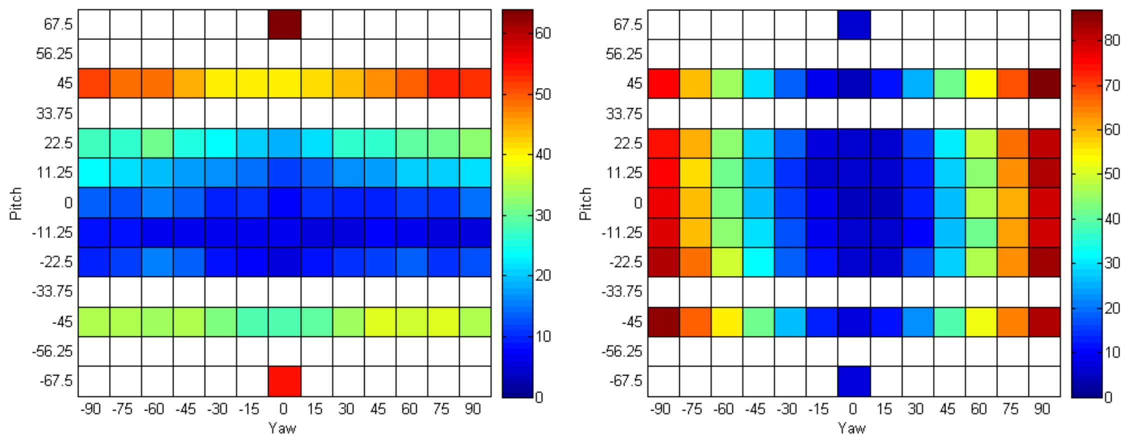
In this section the pose recovery accuracies are evaluated over the *Pointing '04* dataset. In order to do so, the parameters previously selected in Section 5.3 for the 3D geometry fitting methods are used to train the *3D alignment* and *3D regression* models over the full LFPW dataset.

Table 7 shows the average errors obtained for the pitch and yaw as the difference in degrees between the ground truth and predicted angles. As it can be seen, the *3D regression* method obtains lower error values compared to *3D alignment* for both pitch and yaw. These differences are found to be statistically significant in both cases when compared by performing a Wilkozon rank sum test. Even though, the error is found to be quite high for both algorithms, whith an average error of 20.20 and 22.69 degrees for pitch for *3D regression* and *3D alignment* respectively, and of 30.12 and 37.42 for yaw.

	3D alignment	3D regression	p-value	Independence
Pitch error	22.6883 ± 0.6071	20.1984 ± 0.5874	$1.70 \cdot 10^{-9}$	yes
Yaw error	37.4263 ± 1.0112	30.1245 ± 0.9006	$3.64 \cdot 10^{-22}$	yes

Table 7: Comparison between the average pitch and yaw pose recovery errors for the 3D alignment *and* 3D regression *methods over the* Pointing '04 *dataset.*

This does not represent a high fitting error in all cases, but rather the error depends on the pose to be predicted. This is illustrated in Figure 20 for the *3D alignment* algorithm, and in Figure 21 for the *3D regression* algorithm. From these figures it can be seen that for both methods the fitting is good for a range between -30 and +30 degrees for the yaw angle, and for a range between -22.5 and 11.25 for pitch.



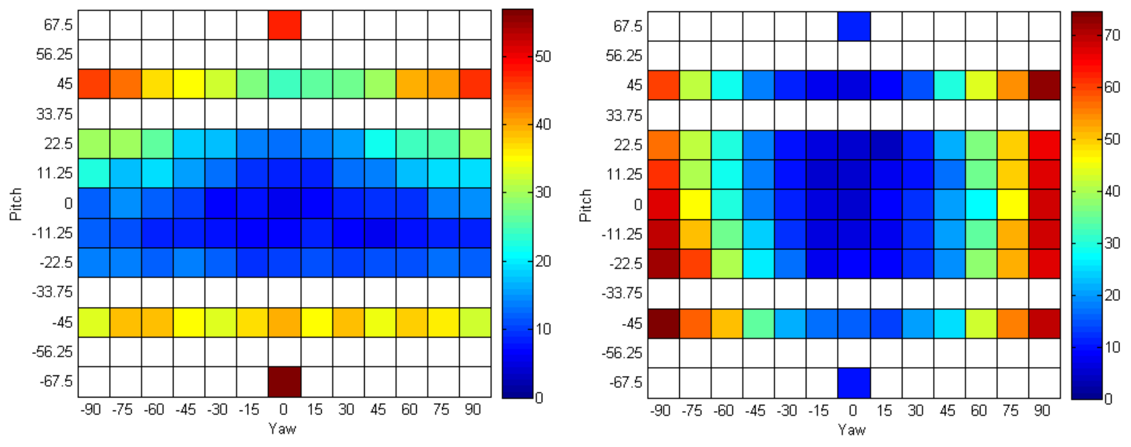
(a) Average pitch errors *(b) Average yaw errors*

Figure 20: Pose recovery errors for the 3D alignment *method over pitch (a) and yaw (b) relative to the ground truth pitch and yaw values.*

When comparing the error distribution between algorithms, it is found that the *3D regression* approach is capable of correctly recovering the pose for a wider range of both yaw and pitch. This is due to the *3D regression* approach direclty regressing the pose when fitting the geometry, while *3D alignment* obtains the 3D geometry and pose from the 2D geometry.

When fitting the 3D geometry through the regression of a 2D geometry, if the geometry is not copletely adjusted to the image, the resulting landmarks can conform an invalid projection of the 3D shape. At the second step of the *3D aligment* approach, the 3D ASM is aligned to this geometry, but since the 2D geometry cannot be completely explained by the 3D ASM, the restricted camera model is forced to partially explain the discrepances. This results in an invalid 3D projection which can be drastically different from the one that would be obtained with a correclty fit 2D geometry. In the case of *3D regression*, since the restricted camera model parameters and 3D ASM weights are directly fit to the image through a cascade of linear regressors, the pose is approximated along with the geometry. This gives a good approximation of the head pose even if the geometry is not completely adjusted to the image.

The precise error values relative to the head pose for *3D alignment* and *3D regression* are found in Appendices D-E respectively. From these values it can be seen that both methods



(a) Average pitch errors *(b) Average yaw errors*

Figure 21: Pose recovery errors for the 3D regression *method over pitch (a) and yaw (b) relative to the ground truth pitch and yaw values.*

perform poorly for pitch angles out of the -22.5 to +11.25 degrees range, and for yaw angles out of the -45 to +45 degrees range. This is likely not a problem with the algorithms, but rather a problem with the data used for training them. Neither the AFLW nor the LFPW datasets contain many examples of images past these ranges, and thus the models could not be trained in order to take into account these cases. Since the *Pointing '04* dataset is labeled only with the pitch and yaw values and does not contain geometric information, it could not be used for training neither.

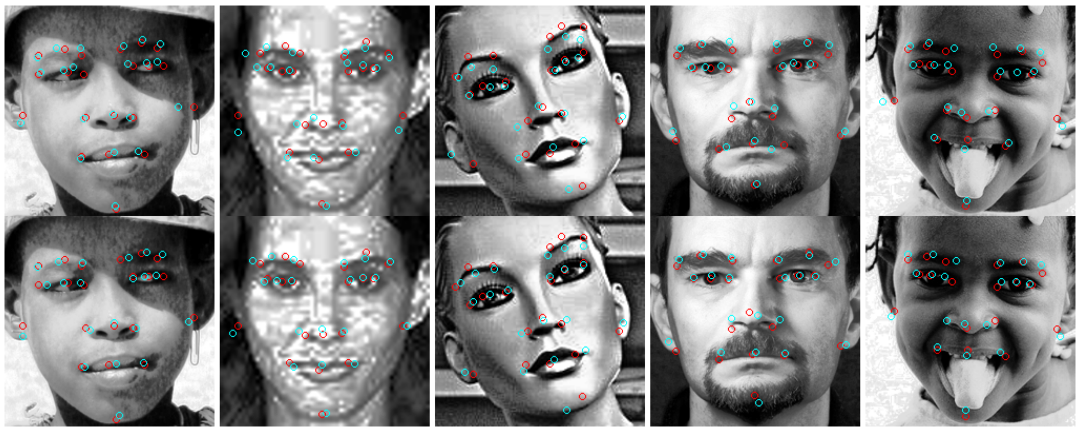
5.5 Qualitative results evaluation

In this section a qualitative evaluation of the 2D and 3D algorithms is performed by looking at sample processed images. Also, the *SDM* method is compared against the *Parametric* parametric approach, and *3D alignment* against *3D regression*.

In the case of the 2D algorithms, some geometry fitting examples over both AFLW and LFPW datasets are shown in Figure 22. From these results, it is shown that an advantage of the *Parametric* approach is its ability to better fit the landmark at the chin. Since this landmark is located at a face region which in most cases has few differences in appearance to other locations around it, it is difficult to precisely locate the region. In the case of the *Parametric* approach, though, since an ASM is used for fitting, the landmark location is forced inside a certain area given the locations of the other landmarks. The same happens for the two other contour landmarks in the case of the AFLW dataset.

It is also seen in the case of the second image for the AFLW dataset that both 2D methods are capable of fitting low-quality images with small resolutions and intensity problems. The AFLW dataset also contains many labelling imprecisions, but both algorithms are in many cases capable of fitting the landmarks more precisely than the ground truth. This is due to the learning process using many images, that on average have the landmarks centered at the correct location, which allows the algorithms to learn a correct regression even with this labelling noise. This is one of the main reasons why both algorithms obtain a higher average NMED error in the AFLW dataset. Both algorithms are also capable of correctly estimating occluded landmarks, as it is shown in some examples of the LFPW dataset where sunglasses are used or there is hair partially covering one of the eyes.

In the case of the 3D fitting algorithms, some fitting examples of which over the AFLW and LFPW datasets are shown in Figure 23, the main observed difference between both 3D methods is the ability of *3D regression* to better estimate the face orientation even when the detected


(a) AFLW dataset

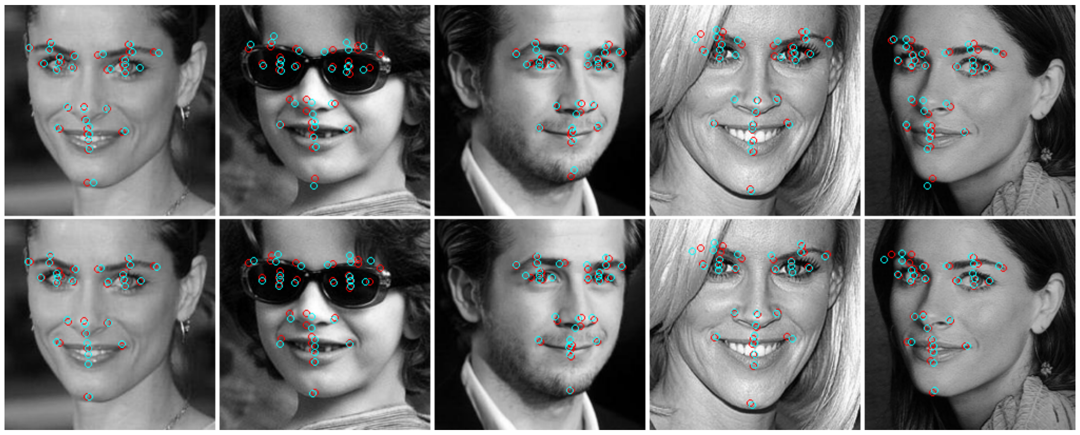
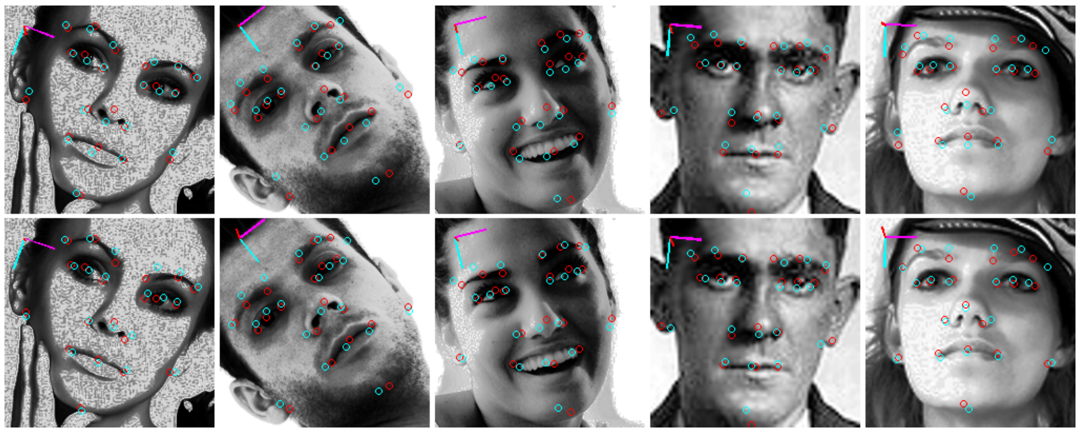

(b) LFPW dataset

Figure 22: Sample image fittings over the AFLW (a) and LFPW (b) datasets for both the Supervised Descent Method *(upper row) and* Parametric *(lower row) methods. The red circles correspond to the ground truth landmark locations, and the cyan circles to the fitted landmarks.*

landmarks are approximately the same. This is specially true in the case of pitch, but also happens for the yaw angle. The reason for this difference is that the *3D alignment* algorithm looks for a best fit between the 3D ASM model and the detected 2D landmarks. This in many cases results in a precise fit of most of the landmarks, except for one or few landmarks which are incorrectly aligned in favour of more precisely locating the rest. In some cases, the incorrectly aligned landmarks are the ones giving the strongest visual clue for the head pose, as it is the case of the nose tip. When looking at the sample images, most of the landmarks are placed at about the same location for both methods, but the nose tip has a higher displacement from the ground truth in the case of *3D alignment*.


(a) AFLW dataset

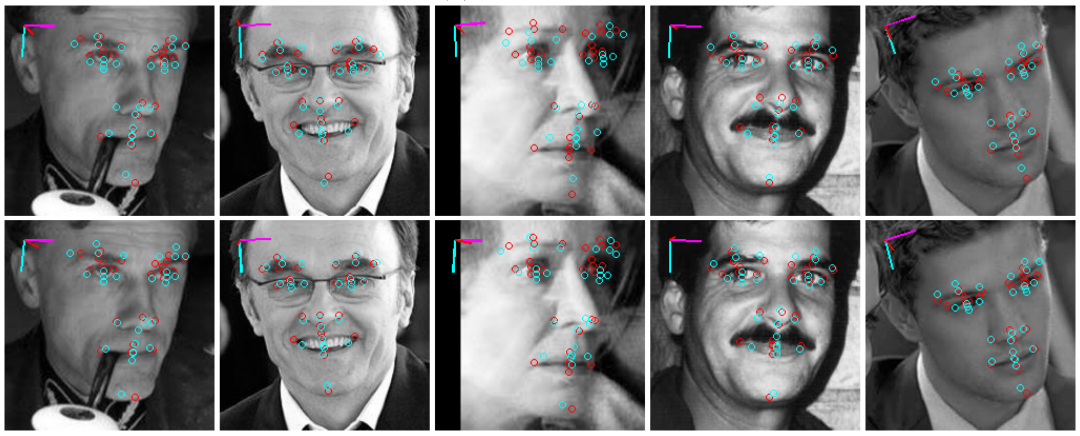

(b) LFPW dataset

Figure 23: Sample image fittings over the AFLW (a) and LFPW (b) datasets for both the 3D alignment *(upper row) and* 3D regression *(lower row) methods. The red circles correspond to the ground truth landmark locations, and the cyan circles to the fitted landmarks.*

6 Conclusions

While in some works ASM models are considered to perform poorly compared to the state of the art in shape regression [31], this work has shown that similar accuracies are obtained when adapting one of those techniques (Supervised Descent Method [38]) in order to regress an ASM instead of directly regressing the geometry. Furthermore, the proposed parametric approach has obtained an average fitting error which is 5% lower in the case of the AFLW dataset while maintaining approximately the same accuracy in the case of LFPW.

The main reason why the parametric approach outperforms SDM is because of the number of principal components required for fitting the face being much lower, passing from 42 values for the 2D shape representation to $21 + 5 = 26$ when using ASM in the case of the AFLW dataset, and from 58 to $23 + 5 = 28$ in the case of LFPW. This means the weights matrix is much smaller when using this parametric representation, but still the same amount of training data is available for calculating these weights. As a result, the algorithm can generalize better. Also, since the parametric approach direclty gives the scale of the current fitting estimate at each cascade step, it provides an easier formulation for scaling the simplified SIFT descriptor windows accordingly, providing scale-invariance. It is also possible that the reduction in the number of parameters required to describe the shape, discarding the lowest ranked eigenvectors, allows the ASM model to ignore the labeling imprecision errors of the ground truth.

The proposed parametric approach has also been extended in order to directly fit the 3D face geometry to 2D face images. This approach is usually done in a two-step process, which comprises the recovery of the 2D geometry and the alignment of a 3D ASM model to the recovered geometry. When comparing the two-step *3D alignment* approach to the direct *3D regresson* of the geometry, *3D regression* is found to have the same average error over the LFPW dataset, but an error 4% higher in the case of AFLW. Even though, this small increase of the error is compensated by a smaller runtime, with direct *3D regression* being 2.37 times faster over the AFLW dataset, and 2.07 times faster over LFPW.

Another advantage of the proposed *3D regression* approach is its ability to better estimate the 3D head pose compared to *3D alignment* in cases where the face geometry could not be precisely recovered. This is shown in Section 5.4, where extreme head poses are more accurately predicted by *3D regression*. This happens because of the two-step *3D alignment* algorithm having to adjust a 3D ASM to the 2D landmarks. When the landmarks are not precisely located in the image, these can take an invalid face configuration which cannot be completely described by the 3D ASM. As a result, the restricted camera model is forced to partially account for the error in predicting the face shape, obtaining an invalid head pose as a result. In the case of *3D regression*, the 3D pose parameters are regressed together with the 3D ASM weights, giving a good approximation to the head pose even if the face landmarks cannot be precisely adjusted to the image.

List of Figures

List of Tables

A Normalized mean euclidean errors for the number of data augmentations and cascade steps grid search over the AFLW dataset

Dataset: AFLW
Algorithm: Supervised Descent Method

	1	5	10	15	20	25
1	0.1911 ± 0.0041	0.1567 ± 0.0040	0.1477 ± 0.0040	0.1439 ± 0.0040	0.1423 ± 0.0040	0.1417 ± 0.0041
2	0.1471 ± 0.0038	0.1188 ± 0.0034	0.1113 ± 0.0034	0.1083 ± 0.0034	0.1061 ± 0.0034	0.1054 ± 0.0035
3	0.1382 ± 0.0039	0.1057 ± 0.0033	0.1025 ± 0.0032	0.1005 ± 0.0031	0.0997 ± 0.0031	0.0990 ± 0.0031
4	0.1377 ± 0.0039	0.1015 ± 0.0033	0.0980 ± 0.0032	0.0971 ± 0.0031	0.0966 ± 0.0031	0.0959 ± 0.0031
5	0.1377 ± 0.0039	0.0999 ± 0.0033	0.0966 ± 0.0032	0.0957 ± 0.0031	0.0958 ± 0.0031	0.0950 ± 0.0030
6	0.1381 ± 0.0039	0.0993 ± 0.0033	0.0959 ± 0.0033	0.0957 ± 0.0031	0.0955 ± 0.0031	0.0943 ± 0.0031
7	0.1383 ± 0.0039	0.0992 ± 0.0034	0.0957 ± 0.0033	0.0954 ± 0.0031	0.0952 ± 0.0031	0.0944 ± 0.0031

Dataset: AFLW
Algorithm: Parametric

	1	5	10	15	20	25
1	0.1802 ± 0.0044	0.1512 ± 0.0041	0.1426 ± 0.0041	0.1401 ± 0.0041	0.1388 ± 0.0041	0.1370 ± 0.0041
2	0.1257 ± 0.0041	0.1121 ± 0.0037	0.1067 ± 0.0036	0.1052 ± 0.0036	0.1038 ± 0.0036	0.1031 ± 0.0036
3	0.1153 ± 0.0041	0.1000 ± 0.0036	0.0991 ± 0.0034	0.0986 ± 0.0033	0.0978 ± 0.0034	0.0978 ± 0.0034
4	0.1144 ± 0.0041	0.0947 ± 0.0036	0.0943 ± 0.0033	0.0937 ± 0.0031	0.0936 ± 0.0032	0.0936 ± 0.0032
5	0.1143 ± 0.0041	0.0923 ± 0.0036	0.0913 ± 0.0033	0.0907 ± 0.0031	0.0909 ± 0.0031	0.0910 ± 0.0031
6	0.1143 ± 0.0041	0.0918 ± 0.0036	0.0900 ± 0.0032	0.0896 ± 0.0031	0.0899 ± 0.0031	0.0900 ± 0.0031
7	0.1143 ± 0.0041	0.0915 ± 0.0036	0.0899 ± 0.0033	0.0892 ± 0.0030	0.0894 ± 0.0031	0.0892 ± 0.0031

Dataset: AFLW
Algorithm: 3D alignment

	1	5	10	15	20	25
1	0.2286 ± 0.0047	0.2017 ± 0.0045	0.1899 ± 0.0044	0.1879 ± 0.0043	0.1869 ± 0.0044	0.1870 ± 0.0043
2	0.1752 ± 0.0045	0.1556 ± 0.0041	0.1498 ± 0.0038	0.1472 ± 0.0039	0.1465 ± 0.0038	0.1454 ± 0.0038
3	0.1646 ± 0.0045	0.1425 ± 0.0038	0.1398 ± 0.0036	0.1377 ± 0.0036	0.1373 ± 0.0035	0.1373 ± 0.0036
4	0.1629 ± 0.0045	0.1363 ± 0.0038	0.1356 ± 0.0036	0.1335 ± 0.0035	0.1331 ± 0.0034	0.1336 ± 0.0034
5	0.1637 ± 0.0045	0.1346 ± 0.0038	0.1332 ± 0.0036	0.1318 ± 0.0035	0.1314 ± 0.0034	0.1321 ± 0.0034
6	0.1640 ± 0.0045	0.1345 ± 0.0038	0.1325 ± 0.0036	0.1311 ± 0.0035	0.1302 ± 0.0034	0.1310 ± 0.0034
7	0.1638 ± 0.0045	0.1345 ± 0.0038	0.1323 ± 0.0036	0.1314 ± 0.0035	0.1305 ± 0.0033	0.1312 ± 0.0034

Dataset: AFLW
Algorithm: 3D regression

	1	5	10	15	20	25
1	0.2225 ± 0.0045	0.1953 ± 0.0045	0.1870 ± 0.0045	0.1857 ± 0.0045	0.1836 ± 0.0045	0.1822 ± 0.0045
2	0.1682 ± 0.0044	0.1480 ± 0.0043	0.1447 ± 0.0043	0.1429 ± 0.0043	0.1427 ± 0.0043	0.1424 ± 0.0043
3	0.1582 ± 0.0044	0.1367 ± 0.0043	0.1351 ± 0.0043	0.1348 ± 0.0043	0.1347 ± 0.0043	0.1343 ± 0.0043
4	0.1572 ± 0.0044	0.1343 ± 0.0044	0.1338 ± 0.0043	0.1337 ± 0.0044	0.1339 ± 0.0043	0.1335 ± 0.0043
5	0.1568 ± 0.0044	0.1339 ± 0.0043	0.1342 ± 0.0044	0.1334 ± 0.0044	0.1346 ± 0.0044	0.1340 ± 0.0043
6	0.1575 ± 0.0045	0.1337 ± 0.0043	0.1342 ± 0.0044	0.1339 ± 0.0044	0.1348 ± 0.0044	0.1341 ± 0.0043
7	0.1575 ± 0.0045	0.1338 ± 0.0043	0.1342 ± 0.0044	0.1341 ± 0.0044	0.1349 ± 0.0044	0.1340 ± 0.0043

B Normalized mean euclidean errors for the number of data augmentations and cascade steps grid search over the LFPW dataset

Dataset:	LFPW
Algorithm:	Supervised Descent Method

	1	5	10	15	20	25
1	0.2621 ± 0.0109	0.2439 ± 0.0105	0.2278 ± 0.0100	0.2164 ± 0.0106	0.2168 ± 0.0103	0.2088 ± 0.0102
2	0.1844 ± 0.0117	0.1424 ± 0.0103	0.1331 ± 0.0104	0.1257 ± 0.0082	0.1239 ± 0.0088	0.1219 ± 0.0091
3	0.1758 ± 0.0121	0.1094 ± 0.0102	0.1005 ± 0.0103	0.0932 ± 0.0075	0.0939 ± 0.0081	0.0951 ± 0.0090
4	0.1753 ± 0.0121	0.0983 ± 0.0103	0.0863 ± 0.0103	0.0817 ± 0.0075	0.0828 ± 0.0082	0.0844 ± 0.0091
5	0.1752 ± 0.0121	0.0961 ± 0.0103	0.0808 ± 0.0104	0.0756 ± 0.0073	0.0775 ± 0.0083	0.0780 ± 0.0081
6	0.1752 ± 0.0121	0.0960 ± 0.0103	0.0801 ± 0.0106	0.0731 ± 0.0077	0.0742 ± 0.0082	0.0765 ± 0.0084
7	0.1753 ± 0.0121	0.0963 ± 0.0103	0.0801 ± 0.0106	0.0728 ± 0.0077	0.0730 ± 0.0083	0.0745 ± 0.0085

Dataset:	LFPW
Algorithm:	Parametric

	1	5	10	15	20	25
1	0.2564 ± 0.0114	0.2371 ± 0.0096	0.2228 ± 0.0108	0.2147 ± 0.0099	0.2103 ± 0.0107	0.2041 ± 0.0097
2	0.1695 ± 0.0117	0.1412 ± 0.0097	0.1333 ± 0.0093	0.1294 ± 0.0084	0.1268 ± 0.0086	0.1250 ± 0.0082
3	0.1590 ± 0.0117	0.1051 ± 0.0099	0.0979 ± 0.0087	0.0951 ± 0.0080	0.0978 ± 0.0095	0.0983 ± 0.0081
4	0.1584 ± 0.0118	0.0957 ± 0.0101	0.0855 ± 0.0088	0.0836 ± 0.0082	0.0861 ± 0.0097	0.0854 ± 0.0078
5	0.1586 ± 0.0118	0.0932 ± 0.0102	0.0803 ± 0.0085	0.0781 ± 0.0082	0.0812 ± 0.0100	0.0806 ± 0.0081
6	0.1586 ± 0.0118	0.0935 ± 0.0102	0.0788 ± 0.0085	0.0746 ± 0.0081	0.0770 ± 0.0100	0.0774 ± 0.0082
7	0.1585 ± 0.0118	0.0935 ± 0.0102	0.0788 ± 0.0085	0.0739 ± 0.0082	0.0759 ± 0.0102	0.0751 ± 0.0083

Dataset:	LFPW
Algorithm:	3D alignment

	1	5	10	15	20	25
1	0.2967 ± 0.0103	0.2877 ± 0.0094	0.2603 ± 0.0094	0.2541 ± 0.0096	0.2425 ± 0.0092	0.2413 ± 0.0098
2	0.2249 ± 0.0109	0.1988 ± 0.0102	0.1875 ± 0.0088	0.1786 ± 0.0085	0.1759 ± 0.0089	0.1749 ± 0.0092
3	0.2156 ± 0.0111	0.1595 ± 0.0100	0.1474 ± 0.0087	0.1442 ± 0.0084	0.1406 ± 0.0084	0.1430 ± 0.0089
4	0.2151 ± 0.0111	0.1467 ± 0.0101	0.1306 ± 0.0083	0.1293 ± 0.0083	0.1276 ± 0.0083	0.1276 ± 0.0077
5	0.2148 ± 0.0111	0.1443 ± 0.0102	0.1241 ± 0.0081	0.1236 ± 0.0081	0.1233 ± 0.0086	0.1238 ± 0.0086
6	0.2145 ± 0.0111	0.1445 ± 0.0103	0.1231 ± 0.0082	0.1204 ± 0.0082	0.1204 ± 0.0085	0.1205 ± 0.0078
7	0.2148 ± 0.0111	0.1439 ± 0.0102	0.1228 ± 0.0082	0.1200 ± 0.0082	0.1195 ± 0.0086	0.1192 ± 0.0079

Dataset:	LFPW
Algorithm:	3D regression

	1	5	10	15	20	25
1	0.2828 ± 0.0107	0.2653 ± 0.0100	0.2470 ± 0.0091	0.2400 ± 0.0095	0.2309 ± 0.0093	0.2315 ± 0.0093
2	0.2087 ± 0.0110	0.1749 ± 0.0094	0.1628 ± 0.0087	0.1588 ± 0.0082	0.1576 ± 0.0085	0.1534 ± 0.0084
3	0.2030 ± 0.0111	0.1422 ± 0.0093	0.1295 ± 0.0082	0.1310 ± 0.0083	0.1281 ± 0.0080	0.1278 ± 0.0084
4	0.2027 ± 0.0111	0.1342 ± 0.0094	0.1206 ± 0.0082	0.1219 ± 0.0085	0.1206 ± 0.0081	0.1216 ± 0.0087
5	0.2026 ± 0.0111	0.1336 ± 0.0097	0.1172 ± 0.0083	0.1175 ± 0.0084	0.1172 ± 0.0090	0.1190 ± 0.0088
6	0.2027 ± 0.0111	0.1331 ± 0.0097	0.1164 ± 0.0082	0.1157 ± 0.0085	0.1156 ± 0.0092	0.1166 ± 0.0088
7	0.2029 ± 0.0111	0.1335 ± 0.0097	0.1164 ± 0.0082	0.1154 ± 0.0085	0.1151 ± 0.0093	0.1154 ± 0.0089

C Normalized mean euclidean errors for the number of validation initializations parameter selection.

AFLW dataset

	SDM	Parametric	3D alignment	3D regression
1	0.0974 ± 0.0032	0.0928 ± 0.0031	0.1340 ± 0.0036	0.1387 ± 0.0045
3	0.7277 ± 0.0159	0.0928 ± 0.0031	0.3422 ± 0.0165	0.1409 ± 0.0046
5	0.0996 ± 0.0036	0.0925 ± 0.0033	0.1341 ± 0.0036	0.1359 ± 0.0045
7	0.0963 ± 0.0031	0.0909 ± 0.0032	0.1319 ± 0.0034	0.1344 ± 0.0043
8	0.0948 ± 0.0030	0.0899 ± 0.0032	0.1313 ± 0.0034	0.1340 ± 0.0044
11	0.0945 ± 0.0030	0.0896 ± 0.0031	0.1313 ± 0.0035	0.1337 ± 0.0043

LFPW dataset

	SDM	Parametric	3D alignment	3D regression
1	0.0763 ± 0.0090	0.0774 ± 0.0085	0.1235 ± 0.0081	0.1229 ± 0.0094
3	0.2718 ± 0.0177	0.1025 ± 0.0120	0.1560 ± 0.0123	0.1443 ± 0.0122
5	0.0832 ± 0.0099	0.0807 ± 0.0088	0.1271 ± 0.0084	0.1210 ± 0.0089
7	0.0763 ± 0.0082	0.0766 ± 0.0085	0.1221 ± 0.0085	0.1160 ± 0.0085
8	0.0749 ± 0.0091	0.0758 ± 0.0080	0.1208 ± 0.0077	0.1159 ± 0.0085
11	0.0739 ± 0.0088	0.0747 ± 0.0078	0.1201 ± 0.0077	0.1142 ± 0.0079

D Average pose fitting errors for each ground truth orientation using the 3D alignment method.

Pitch average errors

	-90	-75	-60	-45	-30	-15	0	15	30	45	60	75	90
-67.5	-	-	-	-	-	-	54.66	-	-	-	-	-	-
-56.25	-	-	-	-	-	-	-	-	-	-	-	-	-
-45	34.23	34.65	33.36	34.07	31.50	29.01	28.99	29.94	33.92	37.09	36.52	37.26	34.26
-33.75	-	-	-	-	-	-	-	-	-	-	-	-	-
-22.5	9.77	11.04	15.52	13.07	8.20	7.25	5.79	8.27	10.61	11.42	13.71	10.64	12.84
-11.25	8.66	8.17	6.50	6.07	6.21	6.24	6.98	5.80	5.82	7.24	6.08	5.84	5.61
0	13.33	12.17	15.15	13.19	9.83	10.19	7.86	10.30	9.38	9.34	11.18	10.92	14.25
11.25	23.63	21.62	19.33	17.99	16.54	14.17	11.16	13.97	16.35	17.89	20.77	20.18	21.74
22.5	28.02	26.40	30.47	25.36	23.08	21.00	18.60	21.22	26.21	26.47	29.46	30.76	32.59
33.75	-	-	-	-	-	-	-	-	-	-	-	-	-
45	51.98	48.83	48.68	44.84	40.66	40.73	40.82	41.91	43.28	46.98	49.16	53.43	52.96
56.25	-	-	-	-	-	-	-	-	-	-	-	-	-
67.5	-	-	-	-	-	-	64.11	-	-	-	-	-	-

Yaw average errors

	-90	-75	-60	-45	-30	-15	0	15	30	45	60	75	90
-67.5	-	-	-	-	-	-	7.02	-	-	-	-	-	-
-56.25	-	-	-	-	-	-	-	-	-	-	-	-	-
-45	84.65	67.80	54.79	40.83	26.92	13.16	6.88	11.07	21.79	39.23	52.38	64.99	82.35
-33.75	-	-	-	-	-	-	-	-	-	-	-	-	-
-22.5	82.12	65.57	49.65	32.07	17.77	12.15	6.29	5.90	16.38	28.27	47.52	62.83	84.05
-11.25	77.82	59.58	43.59	28.60	16.94	8.32	5.48	5.76	10.40	25.89	42.03	61.84	79.62
0	77.16	59.27	42.34	26.87	15.13	8.51	5.12	5.17	12.33	26.98	46.47	61.06	79.85
11.25	75.10	55.88	40.83	25.98	13.79	6.43	5.54	6.30	12.91	27.47	44.68	63.69	81.72
22.5	73.64	59.96	43.90	27.25	17.72	7.64	5.67	5.68	15.59	30.78	48.81	65.65	80.48
33.75	-	-	-	-	-	-	-	-	-	-	-	-	-
45	75.47	59.16	45.48	28.89	18.26	9.14	5.04	11.37	24.74	41.69	53.65	68.57	87.01
56.25	-	-	-	-	-	-	-	-	-	-	-	-	-
67.5	-	-	-	-	-	-	6.20	-	-	-	-	-	-

E Average pose fitting errors for each ground truth orientation using the 3D regression method.

Pitch average errors

	-90	-75	-60	-45	-30	-15	0	15	30	45	60	75	90
-67.5	-	-	-	-	-	-	56.84	-	-	-	-	-	-
-56.25	-	-	-	-	-	-	-	-	-	-	-	-	-
-45	32.95	39.05	38.39	34.85	33.31	36.48	39.17	34.82	38.72	34.39	37.90	35.82	32.54
-33.75	-	-	-	-	-	-	-	-	-	-	-	-	-
-22.5	13.44	13.86	11.84	13.89	12.32	9.44	10.03	11.00	9.89	11.31	10.98	12.59	12.11
-11.25	12.36	11.19	8.63	8.25	7.27	6.89	6.77	8.04	6.74	6.21	7.28	8.44	8.51
0	12.25	14.43	12.26	10.24	6.62	7.28	5.83	7.00	8.76	9.56	9.26	13.91	14.41
11.25	22.57	17.71	18.88	15.84	12.75	9.03	8.25	8.86	12.44	14.05	17.52	19.02	19.45
22.5	29.32	29.46	25.94	18.58	16.92	14.05	12.58	13.63	15.64	21.82	24.81	25.36	30.27
33.75	-	-	-	-	-	-	-	-	-	-	-	-	-
45	45.77	42.98	37.19	34.69	32.24	28.04	24.81	25.93	27.37	30.06	39.30	40.16	46.66
56.25	-	-	-	-	-	-	-	-	-	-	-	-	-
67.5	-	-	-	-	-	-	47.27	-	-	-	-	-	-

Yaw average errors

	-90	-75	-60	-45	-30	-15	0	15	30	45	60	75	90
-67.5	-	-	-	-	-	-	9.79	-	-	-	-	-	-
-56.25	-	-	-	-	-	-	-	-	-	-	-	-	-
-45	74.76	57.27	50.41	34.19	21.56	16.64	16.07	13.29	20.53	25.07	43.09	55.59	69.86
-33.75	-	-	-	-	-	-	-	-	-	-	-	-	-
-22.5	72.05	60.03	40.54	26.04	17.46	7.64	8.84	8.97	11.74	19.07	37.46	51.66	67.36
-11.25	69.66	51.29	35.53	23.97	12.25	6.96	6.62	7.53	12.84	20.48	34.02	51.74	68.83
0	66.59	45.61	30.35	18.12	11.20	6.54	5.06	8.64	12.10	20.23	27.41	46.66	68.35
11.25	61.72	40.68	28.33	17.57	9.94	5.73	4.72	7.27	10.41	16.91	35.95	49.68	67.57
22.5	56.62	41.11	29.53	18.29	9.70	5.94	4.82	4.61	10.56	21.36	37.18	49.89	66.44
33.75	-	-	-	-	-	-	-	-	-	-	-	-	-
45	59.61	41.91	29.04	18.04	10.96	7.31	6.92	9.16	14.89	29.70	43.85	54.47	73.18
56.25	-	-	-	-	-	-	-	-	-	-	-	-	-
67.5	-	-	-	-	-	-	10.53	-	-	-	-	-	-

References

[1] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004.

[2] Adrien Bartoli, Daniel Pizarro, and Marco Loog. Stratified generalized procrustes analysis. *International journal of computer vision*, 101(2):227–253, 2013.

[3] Peter N Belhumeur, David W Jacobs, D Kriegman, and Neeraj Kumar. Localizing parts of faces using a consensus of exemplars. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 545–552. IEEE, 2011.

[4] Luis Miguel Bergasa, Jesús Nuevo, Miguel A Sotelo, Rafael Barea, and María Elena Lopez. Real-time system for monitoring driver vigilance. *Intelligent Transportation Systems, IEEE Transactions on*, 7(1):63–77, 2006.

[5] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194. ACM Press/Addison-Wesley Publishing Co., 1999.

[6] Volker Blanz and Thomas Vetter. Face recognition based on fitting a 3d morphable model. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(9):1063–1074, 2003.

[7] Andrea Giuseppe Bottino and Sandro Cumani. A fast and robust method for the identification of face landmarks in profile images. *WSEAS Transactions on Computers*, 7:1250–1259, 2008.

[8] Lisa M Brown and Ying-Li Tian. Comparative study of coarse head pose estimation. In *Motion and Video Computing, 2002. Proceedings. Workshop on*, pages 125–130. IEEE, 2002.

[9] Xavier P Burgos-Artizzu, Pietro Perona, and Piotr Dollár. Robust face landmark estimation under occlusion. ICCV, 2013.

[10] Chen Cao, Yanlin Weng, Shun Zhou, Yiying Tong, and Kun Zhou. Facewarehouse: a 3d facial expression database for visual computing. 2013.

[11] Xudong Cao, Yichen Wei, Fang Wen, and Jian Sun. Face alignment by explicit shape regression. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2887–2894. IEEE, 2012.

[12] Timothy F Cootes, Gareth J Edwards, Christopher J Taylor, et al. Active appearance models. *IEEE Transactions on pattern analysis and machine intelligence*, 23(6):681–685, 2001.

[13] Timothy F Cootes, Christopher J Taylor, David H Cooper, and Jim Graham. Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.

[14] Matthias Dantone, Juergen Gall, Gabriele Fanelli, and Luc Van Gool. Real-time facial feature detection using conditional regression forests. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2578–2585. IEEE, 2012.

[15] Piotr Dollár, Peter Welinder, and Pietro Perona. Cascaded pose regression. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1078–1085. IEEE, 2010.

[16] Gareth J Edwards, Christopher J Taylor, and Timothy F Cootes. Interpreting face images using active appearance models. In *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*, pages 300–305. IEEE, 1998.

[17] Nicolas Gourier, Daniela Hall, and James L Crowley. Estimating face orientation from robust detection of salient facial structures. In *FG Net Workshop on Visual Observation of Deictic Gestures*, pages 1–9. FGnet (IST–2000–26434) Cambridge, UK, 2004.

[18] Richard Hartley and Andrew Zisserman. Restricted camera estimation. In *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[19] Thanarat Horprasert, Yaser Yacoob, and Larry S Davis. Computing 3d head orientation from a monocular image sequence. In *25th Annual AIPR Workshop on Emerging Applications of Computer Vision*, pages 244–252. International Society for Optics and Photonics, 1997.

[20] Varun Jain and JamesL. Crowley. Head pose estimation using multi-scale gaussian derivatives. 7944:319–328, 2013.

[21] Qiang Ji. 3d face pose estimation and tracking from a monocular camera. *Image and vision computing*, 20(7):499–511, 2002.

[22] Qiang Ji and Xiaojie Yang. Real-time eye, gaze, and face pose tracking for monitoring driver vigilance. *Real-Time Imaging*, 8(5):357–377, 2002.

[23] Michael Jones and Paul Viola. Fast multi-view face detection. *Mitsubishi Electric Research Lab TR-20003-96*, 3:14, 2003.

[24] Martin Kostinger, Paul Wohlhart, Peter M Roth, and Horst Bischof. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. pages 2144–2151, 2011.

[25] Martin Lades, Jan C Vorbruggen, Joachim Buhmann, Jörg Lange, Christoph von der Malsburg, Rolf P Wurtz, and Wolfgang Konen. Distortion invariant object recognition in the dynamic link architecture. *Computers, IEEE Transactions on*, 42(3):300–311, 1993.

[26] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[27] Iain Matthews and Simon Baker. Active appearance models revisited. *International Journal of Computer Vision*, 60(2):135–164, 2004.

[28] Stephen Milborrow and Fred Nicolls. Locating facial features with an extended active shape model. In *Computer Vision–ECCV 2008*, pages 504–513. Springer, 2008.

[29] Erik Murphy-Chutorian, Anup Doshi, and Mohan Manubhai Trivedi. Head pose estimation for driver assistance systems: A robust algorithm and experimental evaluation. In *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 709–714. IEEE, 2007.

[30] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[31] Shaoqing Ren, Xudong Cao, Yichen Wei, and Jian Sun. Face alignment at 3000 fps via regressing local binary features. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2014.

[32] Sami Romdhani and Thomas Vetter. Efficient, robust and accurate fitting of a 3d morphable model. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 59–66. IEEE, 2003.

[33] Henry A Rowley, Shumeet Baluja, and Takeo Kanade. Rotation invariant neural network-based face detection. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 38–44. IEEE, 1998.

[34] Sujith Srinivasan and Kim L Boyer. Head pose estimation using view based eigenspaces. In *Pattern Recognition, International Conference on*, volume 4, pages 40302–40302. IEEE Computer Society, 2002.

[35] Rainer Stiefelhagen, Lei Yang, and Alex Waibel. Modeling focus of attention for meeting indexing based on multiple cues. *Neural Networks, IEEE Transactions on*, 13(4):928–938, 2002.

[36] Hugh R Wilson, Frances Wilkinson, Li-Ming Lin, and Maja Castillo. Perception of head orientation. *Vision research*, 40(5):459–472, 2000.

[37] Laurenz Wiskott, J-M Fellous, N Kuiger, and Christoph Von Der Malsburg. Face recognition by elastic bunch graph matching. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):775–779, 1997.

[38] Xuehan Xiong and Fernando De la Torre. Supervised descent method and its applications to face alignment. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 532–539. IEEE, 2013.

[39] Xiangxin Zhu and Deva Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2879–2886. IEEE, 2012.