To Appear at Computer Vision and Image Understanding (CVIU): Volume 117, Issue 3, March 2013, Pages 250-268

Generative Face Alignment Through 2.5D Active Appearance Models

Pedro Martins, Rui Caseiro, Jorge Batista

Institute of Systems and Robotics, University of Coimbra, DEEC - Polo II, 3030-290 Coimbra, Portugal

Abstract

This work addresses the matching of a 3D deformable face model to 2D images through a 2.5D Active Appearance Models (AAM). We propose a 2.5D AAM that combines a 3D *metric* Point Distribution Model (PDM) and a 2D appearance model whose control points are defined by a *full perspective* projection of the PDM. The advantage is that, assuming a calibrated camera, 3D metric shapes can be retrieved from single view images. Two model fitting algorithms and their computational efficient approximations are proposed: the Simultaneous Forwards Additive (SFA) and the Normalization Forwards Additive (NFA), both based on the Lucas-Kanade framework. The SFA algorithm searches for shape and appearance parameters simultaneously whereas the NFA projects out the appearance from the error image and searches only for the shape parameters. SFA is therefore more accurate. Robust solutions for the SFA and NFA are also proposed in order to take into account the self-occlusion or partial occlusion of the face. Several performance evaluations for the SFA, NFA and theirs efficient approximations were performed. The experiments include evaluating the frequency of converge, the fitting performance in unseen data and the tracking performance in the FGNET Talking Face sequence. All results show that the 2.5D AAM can outperform both the 2D+3D combined models and the 2D standard methods. The robust extensions to occlusion were tested on a synthetic sequence showing that the model can deal efficiently with large head rotation.

1. Introduction

Facial image alignment is a key aspect in many computer vision applications, such as advanced human computer interaction, face recognition, head pose estimation, facial expression analysis, surveillance or realistic graphical animation. Detecting and tracking faces in video is a challenging task due to the non-rigidity structure of faces and also due to the large variability in shape, texture, pose and lighting conditions of their images.

The Active Appearance Model (AAM), introduced by [1], is one of the most effective face alignment technique with respect to fitting accuracy and efficiency. The standard AAMs are intrinsically 2D models, combining a 2D Point Distribution Model (PDM) [2][3] and a 2D appearance model into a single formulation using a fitting process that rely on a precomputed regression matrix.

The AAM has been reformulated with true analytical derived gradients by Matthews *et al.*[4], achieving a better fitting accuracy and real-time performances using the Inverse Compositional (IC) [5] approach. Their solution is probably the fastest introduced so far, where its key to efficiency is that both the Jacobian and the Hessian matrices are constant and can be precomputed. A dual inverse compositional algorithm was also proposed in [6], dealing with both the geometric and photometric transformations in image registration under varying lighting conditions.

Although the excellent performance of the 2D AAM, its convergence ability is severely affected under large 3D head pose variations. To deal with this issue, several solutions have been proposed [7][8][9][10]. View-Based AAM [7] uses multiple 2D AAMs taken from each view, while issues related to self-occlusion are solved by using multiple view-specific templates. Similarly, the solution proposed by [8] uses multiple view appearance models although combined with a sparse 3D PDM. In [11] a IC algorithm for simultaneously fitting a 2D and a 3D PDM to multiple images is proposed. Their fitting methodology, instead of relying on multiple independent optimizations, is formulated in a single-objective optimization by enforcing the same 3D model across all the views. In [9][12], a 3D PDM derived

Email address: `pedromartins@isr.uc.pt` (Pedro Martins)
URL: `http://www.isr.uc.pt/~pedromartins` (Pedro Martins)

from the Candide model [13] is used, being combined with a weak perspective model. In that work, head occlusions are handled by exploiting facial texture symmetry and the model fitting is based on a numerically estimated gradient.

Natural extensions to 3D have also been proposed [12][14][15][16][17], with the 3D Morphable Model (3DMM) [18] one of the most popular. There are several differences between AAMs and 3DMMs. The 3DMMs are built from 3D range scans, therefore are usually constructed to be denser, including several thousands of vertices whereas the AAMs use only a few tens. The appearance model consists of 3D cylindrical folded textures that are densely aligned between all samples in the training set. This huge alignment step involves a modified optical flow, designed to operate on cylindrical coordinates, and smooth interpolation methods to fill in the registration holes. A reflectance model (the Phong model) is also used, i.e. the appearance model also uses surface normals. The large amount of data, due to the density of the 3DMMs, makes the algorithm quite slow, requiring several minutes to fit per frame (50 minutes using a SGI R10000 processor). Efficient 3DMMs, working under a scaled orthographic projection model and based on the IC algorithm, have also been proposed [19]. Still, its Jacobian and Hessian are only locally valid and take an average of 30s per frame to fit, making it impracticable for real-time applications.

This paper addresses the fitting of a 3D shape deformable face model from a single view through 2.5D AAM. The 2.5D model can be viewed as a 3D sparse PDM whose projections define 2D control points for the 2D appearance. This means that 2.5D data has components of both 2D image data and 3D volumetric shape data. Consequently, the 2.5D model combines the advantages of both 3DMMs and 2D AAMs, in particular the robustness to pose changes and the fitting speed. Face alignment on this 2.5D dimensional space will carry an extra level of complexity since the IC approach is invalid in this case [20]. To deal with this problem, Matthews *et al.*[21] proposed a 2D+3D AAM work around by exploiting the 2D and 3D shape models simultaneously. The shape instance generated by the 2D AAM is constrained to be consistent with the projection of a 3D *affine* shape (a 3D PDM is used, build from non-rigid structure from motion [22]). This constraint is formulated as part of the cost function, where a balancing weight is added and the value of this weighting constant is determined manually. In [22] is also showed that any 2D projection of a 3D shape model can be represented by a 2D shape model but at the expense of using up to 6 times more parameters than using a 3D model. However, a weak perspective projection model was used in this demonstration and this property does not hold for the perspective projection model. The solution described in this paper explores the advantages of using a single 3D model to constrain the possible 2D shape projection under the assumption of a full perspective model.

1.1. Paper Contributions

The proposed solution extend the Active Appearance Model approach to deal with matching a 3D face shape model to a single 2D image using a perspective projection model, whereas previous approaches have generally only dealt with scaled orthographic projections. This approach uses a single 3D metric PDM combined with a full perspective model. The use of a full perspective model carries an important advantage over the state of the art solutions. Assuming a calibrated camera, an estimation of the 3D Euclidean shapes can be obtained from a single image and face tracking can be performed by using cameras with short focal length and strong radial distortion (e.g. a low cost webcam). Compared to [21], no balancing weight is required since the approach is based on a single, low dimensional, 3D PDM.

Two algorithms to fit a 3D deformable shape model to a 2D image are proposed. Both algorithms seek to minimize the difference between the projected model and the target image using slightly different strategies: The Simultaneous Forwards Additive (SFA) and the Normalization Forwards Additive (NFA), both based on the Lucas-Kanade forwards additive [23] update step. The SFA algorithm is computationally expensive but more accurate. It searches for shape and appearance parameters simultaneously whereas the NFA projects out the appearance from the error image and searches only for the shape parameters. Although both solutions require evaluating several components per iteration, efficient approximations are proposed leading to an efficient update step. By comparison, our fitting solution is based on analytically derived gradients (”true gradients”) rather than gradients approximated by numerical differences as in [9], genetic algorithms in [16] or generic optimization methods like the simplex in [15]. Finally, real-time performance can be achieving when using the efficient approximations, unlike the 3DMMs [18][19]. Moreover the methods used to acquired 3D dense shapes and textures normally demand very time consuming 3D reconstruction approaches or the use of expensive and cumbersome laser scan hardware.

Expanded solutions for the SFA and NFA are also proposed to handle self and partial occlusion, namely the Robust Simultaneous Forwards Additive (RSFA) and the Robust Normalization Forwards Additive (RNFA). These fitting methods use robust weighting functions that combine outlier estimation with pixel visibility extracted from the 3D pose.

In short, the main contributions in this paper are as follows:

- The use of a 2.5D AAM that combines a 3D metric Point Distribution Model (PDM) and a 2D appearance model whose control points are defined by full perspective projection of the PDM.
- A unique shape model is used where all the six degrees of freedom (6 DOF) are modeled using a simple linear parametric model.
- Two model fitting algorithms and their computationally efficient approximations are proposed: the Simultaneous Forwards Additive (SFA) and the Normalization Forwards Additive (NFA).
- Robust solutions for the SFA and NFA are also proposed in order to take into account head partial and self occlusions.

Other 2D AAM related extensions such as using Light-Invariant theory to deal with external shading [24], multi-band appearance models [25][26][27][28] or modifying the cost function in order to include the previously aligned frame as an additional constraint (SICOV) [29] can be easily incorporated into the proposed algorithms with expected improvements on the overall performance.

1.2. Paper Outline

This paper is organized as follows: **Section** 2 explains the 2.5D parametric model building process. The 3D PDM and 2D appearance models are both described in detail, as well as the full perspective camera model involved. **Section** 3 presents two model fitting algorithms, their respective efficient approximations and also the robust approaches to self and partial occlusion. In **Section** 4 is described how to efficiently evaluate the Jacobian of the warp for both shape and pose parameters, and the 2.5D AAM initial estimate problem is discussed in **Section** 5. Experimental results comparing both robust and non-robust fitting performances are presented in **Section** 6 and the results are discussed. Finally, **Section** 7 summarizes the paper.

As final note, we highlight that this 2.5D AAM image alignment method was first described in [30]. This journal paper describes the technique in more detail and includes the full derivation of the fitting algorithms and both Jacobians of the warp. New experiments and performance evaluation in new data sets are also presented.

2. 2.5D Parametric Models

The aim is to build a 2.5D AAM by combining a 3D metric Point Distribution Model (PDM) with a 2D appearance model whose control points are defined by full perspective projection of the PDM, as shown in figure 1. The 3D PDM is modeled by the shape and pose parameters, $\mathbf{p}$ and $\mathbf{q}$ respectively, that uniquely define a shape s in the 3D space whose projection into the image space sets 2D control points where the generated texture ($\boldsymbol{\lambda}$) is held.

2.1. The Shape Model

The shape of a non-rigid object can be expressed as a linear combination of a set of n basis shapes plus a rigid mean shape vector. This representation is also known as a Point Distribution Model (PDM) [3]. In PDM notation, each 3D v-point shape is defined by the vertex locations of a mesh $s = (X_1, \ldots, X_v, Y_1, \ldots, Y_v, Z_1, \ldots, Z_v)^T$ and the training data consists of a set of annotated images of those shapes (usually by hand). The shapes are then aligned into a common mean shape using a Generalized Procrustes Analysis (GPA) that removes location, scale and rotation effects.

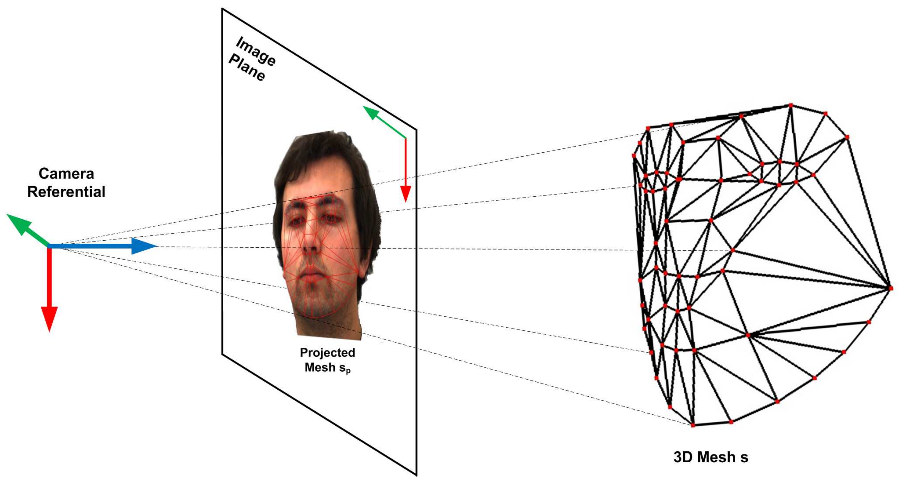


Figure 1: The 2.5D parametric model. The 3D shape model uniquely defines a shape in the 3D space whose projection into the image space sets the 2D control points to the generated texture by the appearance model.

Applying a Principal Components Analysis (PCA) to the aligned shapes, results the linear parametric model $s = s_0 + \Phi\mathbf{p}$, where $\mathbf{p}$ is a vector of shape configuration weights, s_0 is the mean shape (also refereed as the *base mesh*) and the basis $\Phi = [\phi_1 \cdots \phi_n]$ represent the allowed models of deformation. Figure 2 shows the visual representation of the first three modes of variation.

In this work, the 3D PDM, including the full pose variation, is defined by

$$s = s_0 + \sum_{i=1}^{n} p_i \phi_i + \sum_{j=1}^{6} q_j \psi_j^{(t)} + \underbrace{\int_0^{t-1} \sum_{j=1}^{6} q_j \psi_j^{(t)} \partial t}_{s_\psi} . \tag{1}$$

where $\mathbf{p} = (p_1, \ldots, p_n)^T$ are the previous shape parameters, $\mathbf{q} = (q_1, \ldots, q_6)^T$ are the pose parameters and s_ψ is the contribution of pose increments over time t. The first two terms represent the PDM modes of deformation, the third term is the current estimated pose, and the last term (s_ψ) acts as an offset that accumulates pose increments from previous time frames. Note that $\psi_1^{(t)}, \ldots, \psi_6^{(t)}$ are a special set of eigenvectors that are only valid for small changes in pose. With this formulation, the shape model (eq.1) holds the full 6 DOF between the camera referential and the target face by means of incremental pose updates on the current mesh s.

Expressing a rotation of θ radians around an arbitrary axis $\mathbf{w} = (w_x, w_y, w_z)^T$ by the Rodrigues formula

$$\mathbf{R}(\mathbf{w}, \theta) = \mathbf{I}_3 + \hat{\mathbf{w}} \sin(\theta) + \hat{\mathbf{w}}^2(1 - \cos(\theta)), \quad \hat{\mathbf{w}} = \begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix}, \tag{2}$$

the incremental rotation update, based on the linearization of eq.2 and holding the first order terms, is given by

$$\mathbf{R}(\mathbf{w}, \theta) \approx \mathbf{I}_3 + \hat{\mathbf{w}}\theta. \tag{3}$$

By relaxing the constraint that $\mathbf{w}$ is of unit length, the θ coefficient can be dropped from eq.3. According, the pose update that transforms each 3D point $\mathbf{P}_i = (X_i, Y_i, Z_i)$ of the mesh s into $\mathbf{P}'_i$, is given by

$$\mathbf{P}'_i = \mathbf{R}(\mathbf{w})\mathbf{P}_i + \mathbf{T}_i \tag{4}$$

where $\mathbf{T}_i = (t_x, t_y, t_z)^T$ represents the 3D translation components. Defining the pose parameters vector as $\mathbf{q} = [w_x, w_y, w_z, t_x, t_y, t_z]^T$, eq.4 can be written as

$$\mathbf{P}'_i = \begin{bmatrix} 0 & Z_i & -Y_i & 1 & 0 & 0 \\ -Z_i & 0 & X_i & 0 & 1 & 0 \\ Y_i & -X_i & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{q}, \tag{5}$$

that describes how a single mesh point location is updated from $\mathbf{P}_i$ to $\mathbf{P}'_i$ through the pose vector $\mathbf{q}$.

Extending eq.5 to all the 3D mesh points of shape s, the small updates of the pose contribute to the current mesh through an amount of $\sum_{j=1}^{6} \psi_j^{(t)} q_j$. $\Psi = [\ \psi_1^{(t)} \ \ldots \ \psi_6^{(t)}]$ is the extended version of eq.5, incorporating all the v points of the mesh s, and being expressed w.r.t. the *updated* base mesh (which is given by $s_0 + s_\psi$). It can be seen as a special set of *pose eigenvectors* and it is written as

$$\Psi(s_\psi) = \underbrace{\left[\begin{array}{cccccc} 0 & s_0^{z_1}+s_\psi^{z_1} & -s_0^{y_1}-s_\psi^{y_1} & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & s_0^{z_v}+s_\psi^{z_v} & -s_0^{y_v}-s_\psi^{y_v} & 1 & 0 & 0 \\ \hline -s_0^{z_1}-s_\psi^{z_1} & 0 & s_0^{x_1}+s_\psi^{x_1} & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -s_0^{z_v}-s_\psi^{z_v} & 0 & s_0^{x_v}+s_\psi^{x_v} & 0 & 1 & 0 \\ \hline s_0^{y_1}+s_\psi^{y_1} & -s_0^{x_1}-s_\psi^{x_1} & 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ s_0^{y_v}+s_\psi^{y_v} & -s_0^{x_v}-s_\psi^{x_v} & 0 & 0 & 0 & 1 \end{array}\right]}_{\psi_1^{(t)},\ldots,\psi_6^{(t)}}. \tag{6}$$

Since Ψ is a function of s_ψ (as s_0 is constant), it requires being evaluated every time the mesh s is updated.

Finally, the last term of the PDM, s_ψ, consists in the integral form

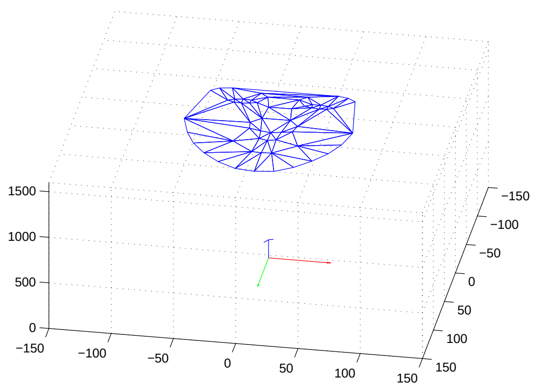
$$s_\psi = \int_0^{t-1} \sum_{j=1}^{6} q_j \psi_j^{(t)} \partial t, \tag{7}$$

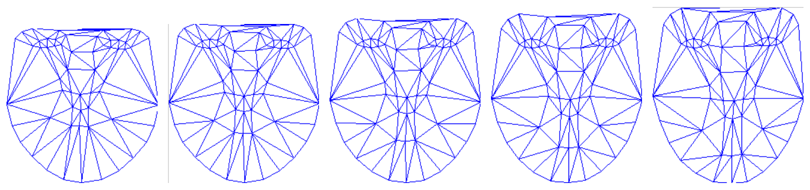
that collects small pose updates over time t. The s_ψ term plays a fundamental role. It overcomes the previous constraint on the incremental pose update so that the 6DOF can be successfully used and it allows updating the base mesh referential (as in eq.6) so that correct head rotations can be modeled.

2.2. The Camera Model

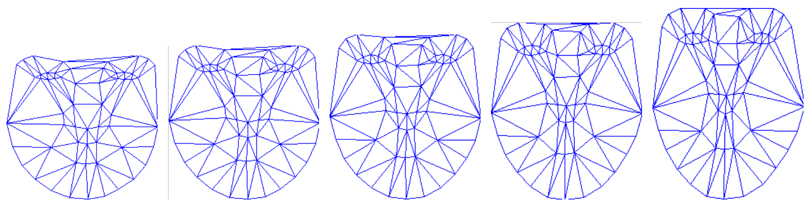
Using a *full perspective* camera, the 3D shape s generated by the PDM (eq.1), is projected into the image space as

$$\begin{bmatrix} w(x_1\cdots x_v) \\ w(y_1\cdots y_v) \\ w\cdots w \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & \alpha_s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \underbrace{\left[\begin{array}{c|c} \mathbf{R}_0 & \mathbf{t}_0 \end{array}\right]}_{\text{Base Pose}} \underbrace{\begin{bmatrix} s^{x_1}\cdots s^{x_v} \\ s^{y_1}\cdots s^{y_v} \\ s^{z_1}\cdots s^{z_v} \\ 1\cdots 1 \end{bmatrix}}_{\text{PDM shape (eq.1)}} \tag{8}$$

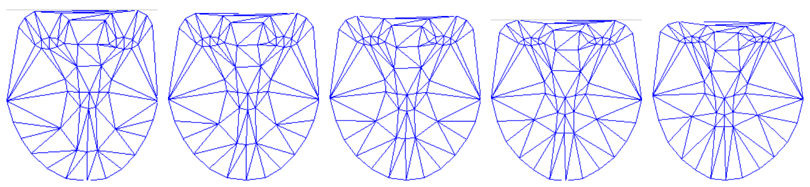




(a) $-3\sigma_1$ (b) $-1.5\sigma_1$ (c) $p_1 = 0$ (d) $+1.5\sigma_1$ (e) $+3\sigma_1$



(f) $-3\sigma_2$ (g) $-1.5\sigma_2$ (h) $p_2 = 0$ (i) $+1.5\sigma_2$ (j) $+3\sigma_2$



(k) $-3\sigma_3$ (l) $-1.5\sigma_3$ (m) $p_3 = 0$ (n) $+1.5\sigma_3$ (o) $+3\sigma_3$

Figure 2: The first three modes of variation of the 3D PDM. On top is shown the 3D base mesh $s_{0\mathbf{p}}$ and the camera frame. The PDM is composed by a mean shape plus a weighted eigenshape contribution. Each row of images shows the 2D image projection of how the shape deforms by spanning the weights $\mathbf{p}_i$ from $-3\sigma_i$ to $3\sigma_i$ $(i = 1, \ldots, n)$. The shape variances, σ_i^2 are captured when applying the PCA in the model building process. The middle column represents the mean shape projection $s_{0\mathbf{p}}$ when $\mathbf{p} = \mathbf{0}$.

where $\mathbf{K}$ is the camera matrix (with f_x, f_y the focal length, c_x, c_y the principal point and α_s the skew parameter) and it is assumed to be known. $\mathbf{R}_0$ and $\mathbf{t}_0$ are the rigid motion components between the camera frame and an extra referential where the PDM is defined. We define this rigid motion as the *base pose*. Both $\mathbf{R}_0$ and $\mathbf{t}_0$ are fixed and estimated during the PDM building process.

2.3. The Texture Model

The texture model is almost identical to the traditional 2D formulation [1], where each training image is texture-warped into a common frame using a warping function $\mathbf{W}$. This function $\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})$ is a piecewise affine warp and is a function of the shape and pose parameters that define the 2D texture control points by means of the perspective projection of the mesh s (using eq.8). The warp is defined for all the projected pixels $\mathbf{x_p}$[1] contained within the *projected base mesh*, $s_{0\mathbf{p}}$, and is given by

$$\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q}) = \mathbf{x}_{\mathbf{p}_i} + \alpha \left(\mathbf{x}_{\mathbf{p}_j} - \mathbf{x}_{\mathbf{p}_i}\right) + \beta \left(\mathbf{x}_{\mathbf{p}_k} - \mathbf{x}_{\mathbf{p}_i}\right), \forall \text{ triangles} \in s_{0\mathbf{p}} \tag{9}$$

where $\mathbf{x}_{\mathbf{p}_i}$, $\mathbf{x}_{\mathbf{p}_j}$, $\mathbf{x}_{\mathbf{p}_k}$ are triangle vertex's coordinates and α, β are the barycentric coordinates [31] for the pixel $\mathbf{x_p}$. The appearance model is obtained by applying a low memory PCA on all the warped training images and it is represented by a base appearance, $\mathbf{A}_0(\mathbf{x_p})$, plus a linear combination of m eigen images $\mathbf{A}_i(\mathbf{x_p})$, as

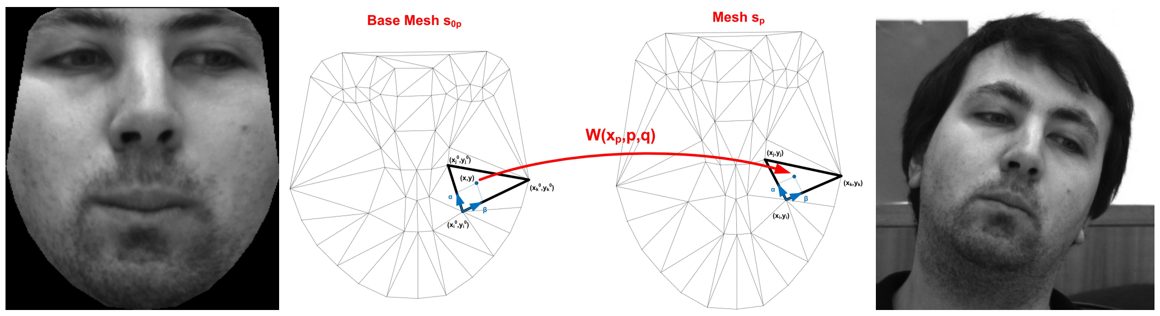
$$\mathbf{A}(\mathbf{x_p}) = \mathbf{A}_0(\mathbf{x_p}) + \sum_{i=1}^{m} \lambda_i \mathbf{A}_i(\mathbf{x_p}), \ \mathbf{x_p} \in s_{0\mathbf{p}} \tag{10}$$

with λ_i being the appearance parameters. To model the gain and illumination offset effects, two extra appearance images are added $\mathbf{A}_{m+1}(\mathbf{x_p}) = \mathbf{A}_0(\mathbf{x_p})$ and $\mathbf{A}_{m+2}(\mathbf{x_p}) = \mathbf{1}$ which imposes the need for orthonormalization [23].

2.4. 3D to 2D Piecewise Affine Warp

The warp function $\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})$, is a piecewise affine warp that is function of the shape and pose parameters. The warp $\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})$ involves a 3D to 2D transformation, i.e. the 3D

[1]During the remaining of the paper, $\mathbf{x_p} = [x, y]^T$ defines a projected 3D point into the 2D image space, by eq.8.



(a) $\mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q}))$ (b) Piecewise affine warp illustration (c) Input image $\mathbf{I}(\mathbf{x_p})$

Figure 3: Piecewise affine warping. The warped image $\mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q}))$ is computed by backwards warping the input image $\mathbf{I}(\mathbf{x_p})$, using the current estimate of the warp $\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})$.

face mesh is generated from shape and pose parameters using eq.1 and then is projected into the image plane by a full perspective model using eq.8. As shown in figure 1, the converted 3D mesh points into 2D define the texture mapping control points. The piecewise affine warp is composed by sets of affine warps between corresponding triangles of the mesh. The base triangles are found by partitioning the convex hull of the projected mean shape, $s_{0\mathbf{p}}$, using the Delaunay triangulation, and each pixel belonging to a given triangle is mapped to its corresponding triangle using barycentric coordinates (see supplementary material section for details).

Figure 3 shows an illustration of this warping procedure. The warped image $\mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q}))$ is computed by backwards warping the input image $\mathbf{I}(\mathbf{x_p})$, therefore preventing holes, using the current estimate of the warp $\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})$. The warp is done for all the pixels $\mathbf{x_p}$ that lie within the projected base mesh $s_{0\mathbf{p}}$.

3. Model Fitting

Fitting the AAM consists in finding the set of parameters, $\mathbf{p}$, $\mathbf{q}$ and $\boldsymbol{\lambda}$ that best describe the face in the target image. Since the Inverse Compositional (IC) approach [5] was proved in [20] to be invalid for the 2.5D case, two algorithms are proposed and described on the paper: the Simultaneous Forwards Additive (SFA) and the Normalization Forwards Additive (NFA), both following the additive formulation proposed by Lucas-Kanade [32][33][34][35].

Both formulations include the 6DOF embedded in the PDM and just like the solutions initially proposed in [32][23], the SFA searches for all the parameters simultaneously whereas the NFA projects out the appearance from the error image. In section 3.3 it is shown how to maintain the fitting efficiency by making a simple approximation, precomputing a couple of terms. The experimental evaluation, as will be shown in section 6, proves that the proposed solution substantially improves the fitting performance.

3.1. Simultaneous Forwards Additive (SFA)

The SFA goal is to minimize the squared difference between the current instance of the appearance and the target warped image. The optimization consists in solving

$$\arg\min_{\mathbf{p},\mathbf{q},\boldsymbol{\lambda}} \sum_{\mathbf{x_p}\in s_{0\mathbf{p}}} \left[\mathbf{A}_0(\mathbf{x_p}) + \sum_{i=1}^{m+2} \lambda_i \mathbf{A}_i(\mathbf{x_p}) - \mathbf{I}(\mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q}))\right]^2 \tag{11}$$

simultaneously for the shape, pose and appearance parameters, $\mathbf{p}$, $\mathbf{q}$ and $\boldsymbol{\lambda}$ respectively. $\mathbf{I}(\mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q}))$ represents the input image $\mathbf{I}(\mathbf{x_p})$ warped by $\mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})$ as defined in section 2.3. The nonlinear optimization in eq.11 can be solved by gradient descent using additive updates to the parameters as

$$\sum_{\mathbf{x}\in s_{0\mathbf{p}}} [\mathbf{A}_0(\mathbf{x_p}) + \sum_{i=1}^{m+2} (\lambda_i + \Delta\lambda_i)\mathbf{A}_i(\mathbf{x_p}) - \mathbf{I}(\mathbf{W}(\mathbf{x_p},\mathbf{p}+\Delta\mathbf{p},\mathbf{q}+\Delta\mathbf{q}))]^2. \tag{12}$$

Expanding and holding the first order Taylor terms gives[2]

$$\sum_{\mathbf{x_p}\in s_{0\mathbf{p}}} \left[\mathbf{A}_0(\mathbf{x_p}) + \sum_{i=1}^{m+2} \lambda_i \mathbf{A}_i(\mathbf{x_p}) + \sum_{i=1}^{m+2} \Delta\lambda_i \mathbf{A}_i(\mathbf{x_p}) - \mathbf{I}(\mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q}))\cdots\right.$$
$$\left.\cdots - \nabla\mathbf{I}\frac{\partial\mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial\mathbf{p}}\Delta\mathbf{p} - \nabla\mathbf{I}\frac{\partial\mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial\mathbf{q}}\Delta\mathbf{q}\right]^2 \tag{13}$$

where $\nabla\mathbf{I}$ $\left(\nabla\mathbf{I} \equiv \nabla\mathbf{I}(\mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})) = (\frac{\partial\mathbf{I}(\mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q}))}{\partial x}, \frac{\partial\mathbf{I}(\mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q}))}{\partial y})\right)$ represents the gradients of the image $\mathbf{I}(\mathbf{x_p})$ evaluated at $\mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})$, before the warp. $\nabla\mathbf{I}$ is computed in the coordinate frame of $\mathbf{I}(\mathbf{x_p})$ and then warped back using the current warp estimate $\mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})$.

[2]The derivation of eq.13 can be found in supplementary material section.

The terms $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{p}}$ and $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{q}}$ are Jacobians of the warp w.r.t. the shape and pose parameters, respectively[3].

Defining the combined parameters vector as $\mathbf{r} = [\mathbf{p}^T \ \mathbf{q}^T \ \boldsymbol{\lambda}^T]^T$ and denoting the $(n+6+m+2)$ Steepest Descent images $\mathbf{SD}(\mathbf{x_p})_{\text{sfa}}$ as

$$\mathbf{SD}(\mathbf{x_p})_{\text{sfa}} = \left[\nabla \mathbf{I} \frac{\partial \mathbf{W}}{\partial \mathbf{p}_1} \ \ldots \ \nabla \mathbf{I} \frac{\partial \mathbf{W}}{\partial \mathbf{p}_n} \ \nabla \mathbf{I} \frac{\partial \mathbf{W}}{\partial \mathbf{q}_1} \ \ldots \ \nabla \mathbf{I} \frac{\partial \mathbf{W}}{\partial \mathbf{q}_6} \ -\mathbf{A}_1(\mathbf{x_p}) \ \ldots \ -\mathbf{A}_{m+2}(\mathbf{x_p}) \right], \tag{14}$$

eq.13 can be written as

$$\sum_{\mathbf{x_p} \in s_{0\mathbf{p}}} \left[\mathbf{A}_0(\mathbf{x_p}) + \sum_{i=1}^{m+2} \lambda_i \mathbf{A}_i(\mathbf{x_p}) - \mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})) - \mathbf{SD}(\mathbf{x_p})_{\text{sfa}} \Delta \mathbf{r} \right]^2. \tag{15}$$

Taking the partial derivative and making-it equal to zero $\left(\frac{\partial(\text{eq.}15)}{\partial \Delta \mathbf{r}} = 0 \right)$ comes the closed from solution for the combined parameters update as

$$\Delta \mathbf{r} = \mathbf{H}_{\text{sfa}}^{-1} \sum_{\mathbf{x_p} \in s_{0\mathbf{p}}} \mathbf{SD}(\mathbf{x_p})_{\text{sfa}}^T \mathbf{E}(\mathbf{x_p})_{\text{sfa}} \tag{16}$$

where

$$\mathbf{H}_{\text{sfa}} = \sum_{\mathbf{x_p} \in s_{0\mathbf{p}}} \mathbf{SD}(\mathbf{x_p})_{\text{sfa}}^T \mathbf{SD}(\mathbf{x_p})_{\text{sfa}} \tag{17}$$

represents the Gauss-Newton approximation to the Hessian matrix and $\mathbf{E}(\mathbf{x_p})_{\text{sfa}}$ represents the error image defined as

$$\mathbf{E}(\mathbf{x_p})_{\text{sfa}} = \mathbf{A}_0(\mathbf{x_p}) + \sum_{i=1}^{m+2} \lambda_i \mathbf{A}_i(\mathbf{x_p}) - \mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})). \tag{18}$$

This procedure is done iteratively and the parameters are additively updated by $\mathbf{r} \leftarrow \mathbf{r} + \Delta \mathbf{r}$ until $\Delta \mathbf{r} \leq \varepsilon$ or a maximum number of iterations is reached.

The SFA is a computationally expensive algorithm since the reevaluation of the image warp $\mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q}))$, the gradients before the warp $\nabla \mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q}))$, the error image $\mathbf{E}(\mathbf{x_p})_{\text{sfa}}$, the Jacobians $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$, $\frac{\partial \mathbf{W}}{\partial \mathbf{q}}$, that depend on $\mathbf{p}$ and $\mathbf{q}$ respectively, the $\mathbf{SD}(\mathbf{x_p})_{\text{sfa}}$ images and the Hessian matrix $\mathbf{H}_{\text{sfa}}$ and its inverse, are required for each iteration. This makes SFA

[3]From now on, $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ and $\frac{\partial \mathbf{W}}{\partial \mathbf{q}}$ will be used as condensed representation for these Jacobians. Section 4 is totally dedicated to evaluate these Jacobians of the warp.

algorithm rather slow but very accurate since it searches for shape, pose and appearance parameters simultaneously. Nevertheless, some components of the Jacobians $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$, $\frac{\partial \mathbf{W}}{\partial \mathbf{q}}$ are constant and can be precomputed (see section 4).

The algorithm 1 summarizes the SFA fitting method. Only at startup, a rough 3D pose estimation is required (the initial $\mathbf{q}$ parameters), taken from a combination of face detector (AdaBoost method [36]) and a 6DOF pose parameters extraction. See section 5 for details. The model starts with the initial shape parameters $\mathbf{p} = \mathbf{0}$ (the mean shape), $\boldsymbol{\lambda} = \mathbf{0}$ (the mean appearance) and $s_\psi = \mathbf{0}$ (zero pose offset).

```
1  Precompute:
2  The 2.5D parametric models: $(s_0, \Phi, \Psi)$ and $(\mathbf{A}_0(\mathbf{x_p}), \mathbf{A}_i(\mathbf{x_p}))$
3  Evaluate $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p})}{\partial \mathbf{x}_k}$ and $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p})}{\partial \mathbf{y}_k}$ for $k = 1, \ldots, v$ (see figure 7)
4  repeat
5      Update pose reference $\Psi(s_\psi)$ with eq.6
6      Warp image $\mathbf{I}(\mathbf{x_p})$ with $\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})$, computing $\mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q}))$
7      Evaluate the gradients $\nabla\mathbf{I}(\mathbf{x_p})$ and warp to $\nabla\mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q}))$
8      Compute the Error image $\mathbf{E}(\mathbf{x_p})_{\mathrm{sfa}}$ using eq.18
9      Evaluate the Jacobian of the warp w.r.t shape $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{p}}$ (eq.41)
10     Evaluate the Jacobian of the warp w.r.t pose $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{q}}$ (eq.44)
11     Compute Steepest Descent images $\mathbf{SD}(\mathbf{x_p})_{\mathrm{sfa}}$ using eq.14
12     Find the Hessian matrix $\mathbf{H}_{\mathrm{sfa}}$ and its inverse with eq.17
13     Compute the parameters updates $\Delta\mathbf{r}$ with eq.16
14     Update parameters $\mathbf{r} \leftarrow \mathbf{r} + \Delta\mathbf{r}$
15     Update pose offset $s_\psi \leftarrow s_\psi + \sum_{j=1}^{6} \psi_j \Delta q_j$
16 until $||\Delta \boldsymbol{r}|| \leq \varepsilon$ or maximum number of iterations reached ;
```

Algorithm 1: Simultaneous Forwards Additive (SFA).

3.2. Normalization Forwards Additive (NFA)

A slightly different algorithm that minimizes the expression in eq.11 is the NFA algorithm. An alternative way of dealing with the linear appearance variation is to project out the appearance images $\mathbf{A}_i(\mathbf{x_p})$ from the error image [23]. Denoting the appearance into a single image by

$$\mathbf{A}(\mathbf{x_p}, \boldsymbol{\lambda}) = \mathbf{A}_0(\mathbf{x_p}) + \sum_{i=1}^{m+2} \lambda_i \mathbf{A}_i(\mathbf{x_p}), \tag{19}$$

eq.11 can be written as

$$\arg\min_{\mathbf{p},\mathbf{q},\boldsymbol{\lambda}} \sum_{\mathbf{x_p} \in s_{0\mathbf{p}}} \left[\mathbf{A}(\mathbf{x_p}, \boldsymbol{\lambda}) - \mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q}))\right]^2. \tag{20}$$

Supposing, by now, that there is no appearance variation, which means that $\mathbf{A}(\mathbf{x_p}, \boldsymbol{\lambda}) = \mathbf{A}_0(\mathbf{x_p})$, the $(n + 6)$ modified $\mathbf{SD}_{\text{nfa}}(\mathbf{x_p})$ images are represented, as

$$\mathbf{SD}(\mathbf{x_p})_{\text{nfa}} = \left[\nabla\mathbf{I}\frac{\partial\mathbf{W}}{\partial\mathbf{p}_1} \;\ldots\; \nabla\mathbf{I}\frac{\partial\mathbf{W}}{\partial\mathbf{p}_n} \; \nabla\mathbf{I}\frac{\partial\mathbf{W}}{\partial\mathbf{q}_1} \ldots \nabla\mathbf{I}\frac{\partial\mathbf{W}}{\partial\mathbf{q}_6}\right]. \tag{21}$$

Applying a first order Taylor expansion to eq.20 results

$$\sum_{\mathbf{x_p} \in s_{0\mathbf{p}}} \left[\mathbf{A}_0(\mathbf{x_p}) - \mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})) - \mathbf{SD}_{\text{nfa}}(\mathbf{x_p}) \begin{bmatrix} \Delta\mathbf{p} \\ \Delta\mathbf{q} \end{bmatrix}\right]^2 \tag{22}$$

and following the same strategy used for the SFA approach, the error image and the Hessian are, respectively, given by

$$\mathbf{E}(\mathbf{x_p})_{\text{lk}} = \mathbf{A}_0(\mathbf{x_p}) - \mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})) \tag{23}$$

and

$$\mathbf{H}_{\text{nfa}} = \sum_{\mathbf{x_p} \in s_{0\mathbf{p}}} \mathbf{SD}(\mathbf{x_p})^T_{\text{nfa}} \mathbf{SD}(\mathbf{x_p})_{\text{nfa}}. \tag{24}$$

Dealing with the full appearance variation $(\mathbf{A}(\mathbf{x_p}, \boldsymbol{\lambda}))$ requires a *normalization* procedure. It is accomplished in the following two steps:

(1) Project the error image, $\mathbf{E}(\mathbf{x})_{\text{lk}}$, into the appearance basis by estimating the $m + 2$ appearance parameters using

$$\lambda_i = \sum_{\mathbf{x_p} \in s_{0\mathbf{p}}} \mathbf{A}_i(\mathbf{x_p}) \mathbf{E}(\mathbf{x_p})_{\text{lk}}, \quad i = 1, \ldots, m + 2 \tag{25}$$

(2) Remove the component of the error image in the direction of $\mathbf{A}_i(\mathbf{x_p})$ finding the normalized error image

$$\mathbf{E}_{\text{nfa}}(\mathbf{x_p}) = \mathbf{E}(\mathbf{x_p})_{\text{lk}} - \sum_{i=1}^{m+2} \lambda_i \mathbf{A}_i(\mathbf{x_p}). \tag{26}$$

The NFA method consists in normalizing the error image (that has appearance $\mathbf{A}(\mathbf{x_p}, \boldsymbol{\lambda})$) so that the component of the error image in the direction $\mathbf{A}_i(\mathbf{x_p})$ is zero. This step has the advantage of estimate the appearance parameters $\boldsymbol{\lambda}$. Finally, the parameters updates are given by

$$\begin{bmatrix} \Delta\mathbf{p} \\ \Delta\mathbf{q} \end{bmatrix} = \mathbf{H}_{\text{nfa}}^{-1} \sum_{\mathbf{x_p} \in s_{0\mathbf{p}}} \mathbf{SD}(\mathbf{x_p})_{\text{nfa}}^{T} \mathbf{E}(\mathbf{x_p})_{\text{nfa}}. \tag{27}$$

The NFA algorithm is less computationally expensive than the SFA, since it projects out the appearance from the error image and searches only for the shape and pose parameters. As shown in algorithm 2, each iteration requires reevaluating the image warp $\mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q}))$, the warped gradients $\nabla\mathbf{I}$, the error image, $\mathbf{E}(\mathbf{x_p})_{\text{lk}}$, the normalized error image $\mathbf{E}(\mathbf{x_p})_{\text{nfa}}$, the Jacobians $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$, $\frac{\partial \mathbf{W}}{\partial \mathbf{q}}$, $\mathbf{SD}(\mathbf{x_p})_{\text{nfa}}$ and the Hessian $\mathbf{H}_{\text{nfa}}^{-1}$. However, note that the $\mathbf{SD}(\mathbf{x_p})_{\text{nfa}}$ images are much smaller in number than the $\mathbf{SD}(\mathbf{x_p})_{\text{sfa}}$, i.e. $(n << m)$, with typical values of n about $10 - 20$ and m about $50 - 80$. The NFA algorithm performs much faster than the SFA.

3.3. Efficient Approximations to SFA and NFA

Some computational load can be reduced by eliminating the need to recompute the image gradients at each iteration. Following the idea proposed by Hager *et al.*[37], and assuming existence of good estimates for all the parameters $\mathbf{p}$, $\mathbf{q}$ and $\boldsymbol{\lambda}$ (in eq.11), the error image $\mathbf{E}(\mathbf{x_p})_{\text{sfa}}$ will be $\approx \mathbf{0}$ and we can say that:

$$\begin{aligned} \left(\mathbf{A}_0(\mathbf{x_p}) + \sum_{i=1}^{m+2} \lambda_i \mathbf{A}_i(\mathbf{x_p})\right) &\approx \mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})) \\ &\Downarrow \\ \underbrace{\left(\nabla\mathbf{A}_0(\mathbf{x_p}) + \sum_{i=1}^{m+2} \lambda_i \nabla\mathbf{A}_i(\mathbf{x_p})\right)}_{\nabla\mathbf{A}_i(\mathbf{x_p}, \boldsymbol{\lambda})} &\approx \nabla\mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})). \end{aligned} \tag{28}$$

```
1  Precompute:
2  The 2.5D parametric models: $(s_0,\ \Phi,\ \Psi)$ and $(\mathbf{A}_0(\mathbf{x_p}), \mathbf{A}_i(\mathbf{x_p}))$
3  Evaluate $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p})}{\partial \mathbf{x}_k}$ and $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p})}{\partial \mathbf{y}_k}$ for $k = 1, \ldots, v$ (see figure 7)
4  repeat
5      Update pose reference $\Psi(s_\psi)$ with eq.6
6      Warp input image $\mathbf{I}$ with $\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})$, computing $\mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q}))$
7      Evaluate the gradients $\nabla\mathbf{I}(\mathbf{x_p})$ and warp to $\nabla\mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q}))$
8      Compute the Error image $\mathbf{E}(\mathbf{x_p})_{\mathrm{lk}}$, eq.23
9      Project-out the error image into $\mathbf{A}_i(\mathbf{x_p})$ basis and estimate the appearance parameters
       $\boldsymbol{\lambda}$ using eq.25
10     Find the normalization error image $\mathbf{E}(\mathbf{x_p})_{\mathrm{nfa}}$ with eq.26
11     Evaluate the Jacobian of the warp w.r.t shape $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{p}}$ (eq.41)
12     Evaluate the Jacobian of the warp w.r.t pose $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{q}}$ (eq.44)
13     Compute Steepest Descent images $\mathbf{SD}(\mathbf{x_p})_{\mathrm{nfa}}$ using eq.21
14     Find the Hessian matrix $\mathbf{H}_{\mathrm{nfa}}$ and its inverse
15     Compute the parameters updates $\begin{bmatrix} \Delta\mathbf{p} \\ \Delta\mathbf{q} \end{bmatrix}$ with eq.27
16     Update parameters $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$ and $\mathbf{q} \leftarrow \mathbf{q} + \Delta\mathbf{q}$
17     Update pose offset $s_\psi \leftarrow s_\psi + \sum_{j=1}^{6} \psi_j \Delta q_j$
18 until $\left\| \begin{matrix} \Delta p \\ \Delta q \end{matrix} \right\| \leq \varepsilon$ or maximum number of iterations reached ;
```

Algorithm 2: Normalization Forwards Additive (NFA).

Under this approximation, the Efficient SFA/NFA Steepest Descent images from eq.14 and eq.21, respectively, can be rewritten as

$$\begin{aligned}\mathbf{SD}(\mathbf{x_p})_{\mathrm{esfa}} = \Bigg[\nabla\mathbf{A}_i(\mathbf{x_p},\boldsymbol{\lambda})\frac{\partial\mathbf{W}}{\partial\mathbf{p}_1}\ \dots\ \nabla\mathbf{A}_i(\mathbf{x_p},\boldsymbol{\lambda})\frac{\partial\mathbf{W}}{\partial\mathbf{p}_n}\ \nabla\mathbf{A}_i(\mathbf{x_p},\boldsymbol{\lambda})\frac{\partial\mathbf{W}}{\partial\mathbf{q}_1}\dots\\ \dots\nabla\mathbf{A}_i(\mathbf{x_p},\boldsymbol{\lambda})\frac{\partial\mathbf{W}}{\partial\mathbf{q}_6}\ -\mathbf{A}_1(\mathbf{x_p})\ \dots\ -\mathbf{A}_{m+2}(\mathbf{x_p})\Bigg],\end{aligned} \tag{29}$$

and

$$\mathbf{SD}(\mathbf{x_p})_{\mathrm{enfa}} = \left[\nabla\mathbf{A}_0(\mathbf{x_p})\frac{\partial\mathbf{W}}{\partial\mathbf{p}_1}\ \dots\ \nabla\mathbf{A}_0(\mathbf{x_p})\frac{\partial\mathbf{W}}{\partial\mathbf{p}_n}\ \nabla\mathbf{A}_0(\mathbf{x_p})\frac{\partial\mathbf{W}}{\partial\mathbf{q}_1}\dots\nabla\mathbf{A}_0(\mathbf{x_p})\frac{\partial\mathbf{W}}{\partial\mathbf{q}_6}\right]. \tag{30}$$

The approximation in eq.28, besides providing extra computation efficiency (the gradients of the template $\nabla\mathbf{A}_0$ can be precomputed when using ENFA and also the gradients of all the eigen faces $\nabla\mathbf{A}_i$ when using ESFA), it has the great advantage of providing better stability to noise sensitivity since it avoids the reevaluation of the gradients in the input image $\nabla\mathbf{I}(\mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q}))$ and at both warps $\frac{\partial\mathbf{I}(\mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q}))}{\partial x}$, $\frac{\partial\mathbf{I}(\mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q}))}{\partial y}$ of each iteration.

Figure 4 shows an example of the ESFA fitting method applied in a video sequence. Each image shows three different views of the 3D mesh and input frame overlaid with its current projection.

The algorithms 5 and 6, shown in Appendix B, summarize the detailed steps of the Efficient versions of the Simultaneous and the Normalization Forwards Additive approaches.

3.4. Robust Fitting

Both SFA and NFA are data driven algorithms and the error image continuously drives the models in further updates. In the case of occlusion, the error image accounts for all the pixels equally (L2 norm) leading the model to diverge. To overcome this problem, occlusion can be modeled as outlier pixels in the appearance model and handled by robust fitting methods [38] [39], namely by Iteratively Reweighted Least Squares (IRLS) where outliers are not accounted for the parameters updates.

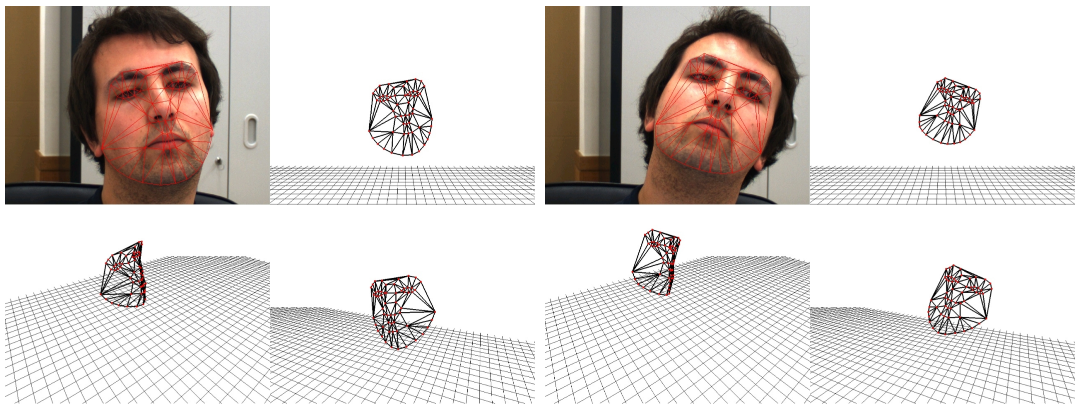


Figure 4: 2.5D AAM fitting using the Efficient Simultaneous Forwards Additive (ESFA) algorithm. Each image shows the input frame overlaid with the projected mesh and three different views of the current 3D mesh s. The full video sequence can be seen at http://www.isr.uc.pt/~pedromartins/Videos/AAM25D.

The robust fitting seeks to minimize

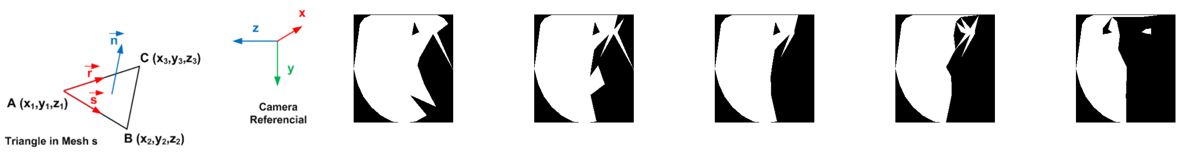
$$\arg\min_{\mathbf{p},\mathbf{q},\boldsymbol{\lambda}} \sum_{\mathbf{x_p}\in s_{0\mathbf{p}}} \rho\left(\left[\underbrace{\mathbf{A}_0(\mathbf{x_p}) + \sum_{i=1}^{m+2}\lambda_i\mathbf{A}_i(\mathbf{x_p}) - \mathbf{I}(\mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q}))}_{\mathbf{E}(\mathbf{x_p})_{\text{sfa}}}\right]^2, \sigma_{\mathbf{x_p}}\right) \tag{31}$$

where $\rho(.)$ is a robust error function that has the purpose of weighting the large errors on $\mathbf{E}(\mathbf{x_p})_{\text{sfa}}$ so that they have less significance in updating the fitting parameters. The vector of scale parameters is defined as $\sigma_{\mathbf{x_p}}$ and can be estimated from the error image, $\mathbf{E}(\mathbf{x_p})_{\text{sfa}}$. The notation $\sigma_{\mathbf{x_p}}$ reflects that each pixel $\mathbf{x_p}$ is treated independently i.e. the decision if a pixel is occluded is not influenced by any other pixel.

3.4.1. Modified Robust Error Function

Several robust error functions can be used, such as the Hubber, the Tukey or the Cauchy function (see [40] for an AAM related comparison). In this work a slightly modified robust error function, based on the Talwar function, is used. The Talwar function assigns a weight of 1 to inliers and 0 to outliers, according to

$$\rho(\mathbf{E}(\mathbf{x_p}),\sigma_{\mathbf{x_p}}) = \begin{cases} 1, & |\mathbf{E}(\mathbf{x_p})| \leq \sigma_{\mathbf{x_p}} \\ 0, & |\mathbf{E}(\mathbf{x_p})| > \sigma_{\mathbf{x_p}}. \end{cases} \tag{32}$$



(a) Back-face Culling (b) 40° (c) 50° (d) 60° (e) 75° (f) 90°

Figure 5: a) Back-face Culling illustration. **n** is the normal vector from a triangle in mesh s and **z** is the view vector from the camera reference. Images (b)(c)(d)(e) show the triangle visibility mask over the projected base mesh, $s_{0\mathbf{p}}$, for a head pan variation of 40°, 50°, 60°, 75° and 90° w.r.t the base pose using the Back-face Culling technique. Non-visible triangles (in black) are not used to update the parameters.

The scale parameter, $\sigma_{\mathbf{x_p}}$, can also be estimated from several ways. Since statistical distribution of the error image is unknown it can be assumed that the error image has a given percentage of outliers (e.g. 5% or 10%) and $\sigma_{\mathbf{x_p}}$ is set such that the largest user defined percentage of error pixels are rejected. Other solution, consists in estimate $\sigma_{\mathbf{x_p}}$ from the fitting error residuals using the Median of Absolute Deviations (MAD). The scale estimation can be moved into the AAM model building process by simply running, in an offline mode, a fitting algorithm for every (unoccluded) training image and then estimate the MAD fitting error. Figure 6 shows robust fitting results using the MAD as an estimate to the scale parameters $\sigma_{\mathbf{x_p}}$.

The 2.5D proposed model has the advantage of being able to estimate the visible areas (say mesh triangles) in the image projection model. The robust function modification consists in using information about the triangles visibility over the projected base mesh (by Back-face Culling) and select the invisible triangles by the camera to be dropped. These occluded triangles are established as outliers and are not taken into consideration in the fitting process. See figure 5.

3.4.2. Robust Fitting Algorithms (RSFA and RNFA)

The derivation of the Robust versions of SFA and NFA algorithms, RSFA and RNFA respectively, is similar to those of section 3, where the RSFA final parameters update is given by

$$\Delta\mathbf{r} = \mathbf{H}_{\text{rsfa}}^{-1} \sum_{\mathbf{x} \in s_{0\mathbf{p}}} \rho(\mathbf{E}(\mathbf{x_p})_{\text{sfa}}^2)\mathbf{SD}(\mathbf{x_p})_{\text{sfa}}^T \mathbf{E}(\mathbf{x_p})_{\text{sfa}} \tag{33}$$

being $\rho(\mathbf{E}(\mathbf{x_p})_{\text{sfa}}^2)$ a weight mask that measures the confidence of each pixel over the base mesh. The Hessian is defined as

$$\mathbf{H}_{\text{rsfa}} = \sum_{\mathbf{x} \in s_{0\mathbf{p}}} \rho(\mathbf{E}(\mathbf{x_p})_{\text{sfa}}^2)\mathbf{SD}(\mathbf{x_p})_{\text{sfa}}^T \mathbf{SD}(\mathbf{x_p})_{\text{sfa}}. \tag{34}$$

Algorithm 3 describes in detail the steps required for the RSFA.

```
1  Precompute:
2  The 2.5D parametric models: $(s_0, \Phi, \Psi)$ and $(\mathbf{A}_0(\mathbf{x_p}), \mathbf{A}_i(\mathbf{x_p}))$
3  Evaluate $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p})}{\partial \mathbf{x}_k}$ and $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p})}{\partial \mathbf{y}_k}$ for $k = 1, \ldots, v$ (see figure 7)
4  repeat
5      Update pose reference $\Psi(s_\psi)$ with eq.6
6      Warp input image $\mathbf{I}$ with $\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})$, computing $\mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q}))$
7      Evaluate the gradients $\nabla\mathbf{I}(\mathbf{x_p})$ and warp to $\nabla\mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q}))$
8      Evaluate triangle visibility by Back Face Culling
9      Compute the Error image $\mathbf{E}(\mathbf{x_p})_{\text{sfa}}$ using eq.18
10     Estimate the weight mask $\rho(\mathbf{E}(\mathbf{x_p})_{\text{sfa}}^2)$
11     Evaluate the Jacobian of the warp w.r.t shape $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{p}}$ (eq.41)
12     Evaluate the Jacobian of the warp w.r.t pose $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{q}}$ (eq.44)
13     Compute Steepest Descent images $\mathbf{SD}(\mathbf{x_p})_{\text{sfa}}$ using eq.14
14     Find the Hessian matrix $\mathbf{H}_{\text{rsfa}}$ and its inverse with eq.34
15     Compute the parameters updates, $\Delta\mathbf{r}$, with eq.33
16     Update parameters $\mathbf{r} \leftarrow \mathbf{r} + \Delta\mathbf{r}$
17     Update pose offset $s_\psi \leftarrow s_\psi + \sum_{j=1}^{6} \psi_j \Delta q_j$
18 until $||\Delta \boldsymbol{r}|| \leq \varepsilon$ or maximum number of iterations reached ;
```

Algorithm 3: Robust Simultaneous Forwards Additive (RSFA).

In the same way, the Robust version of NFA (RNFA) includes a weight mask in the

Steepest Descent images, when evaluating the Hessian matrix,

$$\mathbf{H}_{\text{rnfa}} = \sum_{\mathbf{x}\in s_{0\mathbf{p}}} \rho(\mathbf{E}(\mathbf{x_p})^2_{\text{rnfa}})\mathbf{SD}(\mathbf{x_p})^T_{\text{nfa}}\mathbf{SD}(\mathbf{x_p})_{\text{nfa}} \tag{35}$$

and the parameters updates become

$$\begin{bmatrix} \Delta\mathbf{p} \\ \Delta\mathbf{q} \end{bmatrix} = \mathbf{H}^{-1}_{\text{rnfa}} \sum_{\mathbf{x}\in s_{0\mathbf{p}}} \rho(\mathbf{E}(\mathbf{x_p})^2_{\text{rnfa}})\mathbf{SD}(\mathbf{x_p})^T_{\text{nfa}}\mathbf{E}(\mathbf{x_p})_{\text{rnfa}} \tag{36}$$

with the error image being

$$\mathbf{E}(\mathbf{x_p})_{\text{rnfa}} = \mathbf{A}_0(\mathbf{x_p}) + \sum_{i=1}^{m+2} \lambda_i \mathbf{A}_i(\mathbf{x_p}) - \mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})). \tag{37}$$

Just like in the NFA algorithm, the RNFA requires an appearance normalization step for the error image. As referred in section 3.2, the goal of this normalization step is to make the component of the error image in the direction of $\mathbf{A}_i(\mathbf{x_p})$ to be zero. The NFA method deals with this by simply projecting the error image into the appearance basis $(\mathbf{A}_i(\mathbf{x_p}))$. However the same approach can not be used in the robust version. With the use of a robust error function, $\rho(.)$, the appearance vectors are no longer orthonormal.

A slightly modified solution of the normalization step, initially proposed in [23], can be used. Starting from the error image $\mathbf{E}(\mathbf{x})_{\text{nfa}}$, the goal is to compute the appearance parameters update $\Delta\boldsymbol{\lambda}$ that minimize

$$\sum_{\mathbf{x_p}\in s_{0\mathbf{p}}} \rho(\mathbf{E}(\mathbf{x_p})^2_{\text{rnfa}}) \left(\mathbf{E}_{\text{rnfa}}(\mathbf{x_p}) + \sum_{i=1}^{m+2} \Delta\lambda_i \mathbf{A}_i(\mathbf{x_p}) \right)^2, \tag{38}$$

which has the least squares minimum given by

$$\Delta\boldsymbol{\lambda} = \mathbf{H}^{-1}_{\text{A}} \sum_{\mathbf{x_p}\in s_{0\mathbf{p}}} \rho(\mathbf{E}(\mathbf{x_p})^2_{\text{rnfa}})\mathbf{A}_i(\mathbf{x_p})^T\mathbf{E}(\mathbf{x_p})_{\text{rnfa}} \tag{39}$$

where

$$\mathbf{H}_{\text{A}} = \sum_{\mathbf{x_p}\in s_{0\mathbf{p}}} \rho(\mathbf{E}(\mathbf{x_p})^2_{\text{rnfa}}) \sum_{i=1}^{m+2} \mathbf{A}_i(\mathbf{x_p})^T\mathbf{A}_i(\mathbf{x_p}) \tag{40}$$

is the appearance Hessian.

Algorithm 4 describe the RNFA algorithm steps, including the robust appearance normalization.

```
1  Precompute:
2  The 2.5D parametric models: $(s_0, \Phi, \Psi)$ and $(\mathbf{A}_0(\mathbf{x_p}), \mathbf{A}_i(\mathbf{x_p}))$
3  Evaluate $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p})}{\partial \mathbf{x}_k}$ and $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p})}{\partial \mathbf{y}_k}$ for $k = 1, \ldots, v$ (see figure 7)
4  repeat
5      Update pose reference $\Psi(s_\psi)$ with eq.6
6      Warp input image $\mathbf{I}$ with $\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})$, computing $\mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q}))$
7      Evaluate the gradients $\nabla\mathbf{I}(\mathbf{x_p})$ and warp to $\nabla\mathbf{I}(\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q}))$
8      Evaluate triangle visibility by Back Face Culling
9      Compute the Error image $\mathbf{E}(\mathbf{x_p})_{\mathrm{rnfa}}$ using eq.37
10     Estimate the weight mask $\rho(\mathbf{E}(\mathbf{x_p})^2_{\mathrm{rnfa}})$
11     Find the Hessian appearance $\mathbf{H}_{\mathrm{A}}$ with eq.40
12     Compute the appearance parameters update $\Delta\boldsymbol{\lambda}$ with eq.39
13     Update appearance parameters $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \Delta\boldsymbol{\lambda}$
14     Recompute $\mathbf{E}(\mathbf{x_p})_{\mathrm{rnfa}}$ using eq.37 (normalized error image)
15     Evaluate the Jacobian of the warp w.r.t shape $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{p}}$ (eq.41)
16     Evaluate the Jacobian of the warp w.r.t pose $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{q}}$ (eq.44)
17     Compute Steepest Descent images $\mathbf{SD}(\mathbf{x_p})_{\mathrm{nfa}}$ using eq.21
18     Find the Hessian matrix $\mathbf{H}_{\mathrm{rnfa}}$ and its inverse with eq.35
19     Compute the parameters updates $\begin{bmatrix} \Delta\mathbf{p} \\ \Delta\mathbf{q} \end{bmatrix}$ with eq.36
20     Update parameters $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$ and $\mathbf{q} \leftarrow \mathbf{q} + \Delta\mathbf{q}$
21     Update pose offset $s_\psi \leftarrow s_\psi + \sum_{j=1}^{6} \psi_j \Delta q_j$
22 until $\left\| \begin{matrix} \Delta\boldsymbol{p} \\ \Delta\boldsymbol{q} \end{matrix} \right\| \leq \varepsilon$ or maximum number of iterations reached ;
```
Algorithm 4: Robust Normalization Forwards Additive (RNFA).

3.4.3. Efficient Robust Approximations (ERSFA and ERNFA)

The efficient approximations presented in section 3.3 are also valid for the robust fitting versions. The main changes w.r.t. the standard versions (RSFA and RNFA) are the use of efficient Steepest Descent images in eqs.29 and 30, respectively. See algorithms 7 and 8 in Appendix B for details. Figure 6-top shows some occlusion robust examples using the ERNFA algorithm in a video sequence. Dealing with self-occlusion effects can be seen in figure 6-bottom where the ERSFA algorithm was used.

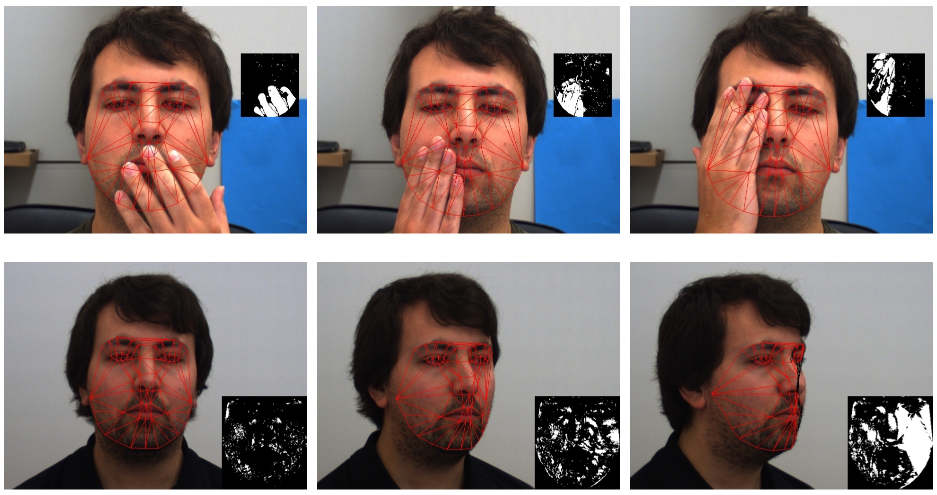


Figure 6: The top images show the robust 2.5D AAM fitting using the ERNFA algorithm. The weight mask $\rho(\mathbf{E}(\mathbf{x_p})^2_{\text{rnfa}})$ is shown on the right. The scale parameters $\sigma_{\mathbf{x_p}}$ were estimated assuming that there always exists 10% of outliers in the error image. On bottom images the ERSFA algorithm was used with $\sigma_{\mathbf{x_p}}$ estimated from the fitting error MAD. Both full video sequences can be seen at http://www.isr.uc.pt/~pedromartins/Videos/AAM25D.

4. The Jacobian of The Warp

The Jacobians of the warp measure the rate of change of the destination in the warp $\mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})$ w.r.t. the parameters $\mathbf{p}$ and $\mathbf{q}$. Two Jacobians must be derived, $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{p}}$ and $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{q}}$, w.r.t. shape and pose parameters, respectively.

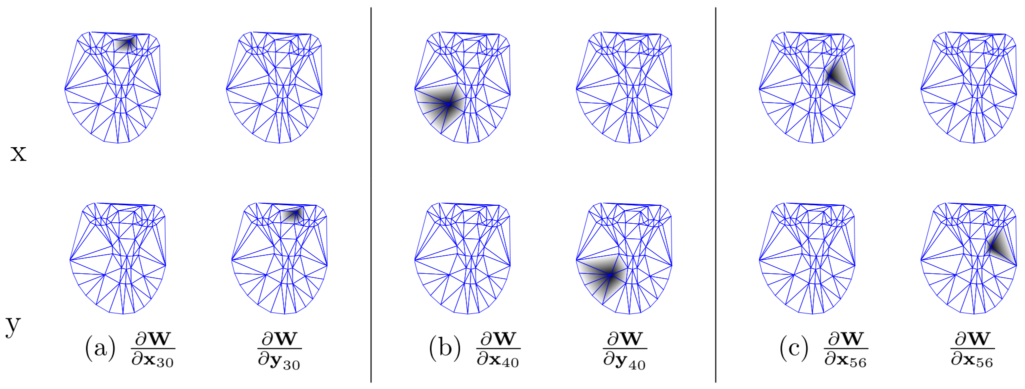

x
y
(a) $\frac{\partial \mathbf{W}}{\partial \mathbf{x}_{30}}$
$\frac{\partial \mathbf{W}}{\partial \mathbf{y}_{30}}$
(b) $\frac{\partial \mathbf{W}}{\partial \mathbf{x}_{40}}$
$\frac{\partial \mathbf{W}}{\partial \mathbf{y}_{40}}$
(c) $\frac{\partial \mathbf{W}}{\partial \mathbf{x}_{56}}$
$\frac{\partial \mathbf{W}}{\partial \mathbf{x}_{56}}$

Figure 7: (a) (b) (c) Shows $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{x}_k}$ and $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{y}_k}$ for the landmarks 30, 40 and 56, respectively. Top and bottom rows represent $\mathbf{W_x}(\mathbf{x_p},\mathbf{p},\mathbf{q})$ and $\mathbf{W_y}(\mathbf{x_p},\mathbf{p},\mathbf{q})$ components. For clarity the shown images are black/white inverted. The location of the vertex has a maximum value and decays linearly to its neighbors. Note the highly sparse matrices shown.

4.1. Jacobian of The Warp for The Shape Parameters

The Jacobian of the warp for the shape parameters can be decomposed by the chain rule as

$$\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{p}} = \sum_{k=1}^{v} \left[\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{x}_k} \frac{\partial \mathbf{x}_k}{\partial \mathbf{p}} + \frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{y}_k} \frac{\partial \mathbf{y}_k}{\partial \mathbf{p}} \right]. \tag{41}$$

Taking eq.9, comes that $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{x}_k} = (1-\alpha-\beta, 0)$ and $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{y}_k} = (0, 1-\alpha-\beta)$. These Jacobians are images w.r.t. a particular vertex and have the same size of the projected base mesh $s_{0\mathbf{p}}$. Figure 7 shows examples of these images for some landmarks (note the x and y components). The Jacobians are only non zero around the neighbor triangles of vertex k^{th}, taking the maximum value of 1 at the vertex location and decaying linearly with a rate of $(1-\alpha-\beta)$ to the other surrounding vertex's.

The remaining terms $\frac{\partial \mathbf{x}_k}{\partial \mathbf{p}_i}$ and $\frac{\partial \mathbf{y}_k}{\partial \mathbf{p}_i}$ are both scalars, found by combining eq.8 and eq.1, as

$$\begin{bmatrix} w\mathbf{x}_k \\ w\mathbf{y}_k \\ w \end{bmatrix} = \underbrace{\mathbf{K} \left[\begin{array}{c|c} \mathbf{R}_0 & \mathbf{t}_0 \end{array} \right]}_{\mathbf{M}_0} \begin{bmatrix} s_0^{x_k} + p_i\phi_i^{x_k} + \sum_{j\neq i}^{n} p_j\phi_j^{x_k} + \sum_{j=1}^{6} q_j\Psi_j^{x_k} + s_\psi^{x_k} \\ s_0^{y_k} + p_i\phi_i^{y_k} + \sum_{j\neq i}^{n} p_j\phi_j^{y_k} + \sum_{j=1}^{6} q_j\Psi_j^{y_k} + s_\psi^{y_k} \\ s_0^{z_k} + p_i\phi_i^{z_k} + \sum_{j\neq i}^{n} p_j\phi_j^{z_k} + \sum_{j=1}^{6} q_j\Psi_j^{z_k} + s_\psi^{z_k} \\ 1 \end{bmatrix}. \tag{42}$$

To compute $\frac{\partial \mathbf{x}_k}{\partial \mathbf{p}_i}$ we take the differential $\frac{\partial}{\partial \mathbf{p}_i}(\frac{w\mathbf{x}_k}{w})$ from eq.42, and do the same for $\frac{\partial \mathbf{y}_k}{\partial \mathbf{p}_i} =$

$\frac{\partial}{\partial \mathbf{p}_i}(\frac{w\mathbf{y}_k}{w})$, resulting in

$$\frac{\partial \mathbf{x}_k}{\partial \mathbf{p}_i} = \frac{\xi_1 \Xi_3 - \Xi_1 \xi_3}{(\Xi_3)^2} \quad \text{and} \quad \frac{\partial \mathbf{y}_k}{\partial \mathbf{p}_i} = \frac{\xi_2 \Xi_3 - \Xi_2 \xi_3}{(\Xi_3)^2} \tag{43}$$

with $i = 1, \ldots, n$ (shape parameters) and $k = 1, \ldots, v$ (landmarks). The $\xi_1, \xi_2, \xi_3, \Xi_1, \Xi_2$ and Ξ_3 are all scalars values defined in Appendix A. Note that the amount $\sum_{j \neq i}^{n} p_j \phi_j$ (non-rigid shape deformation excluding the i^{th} parameter) is constant when taking the i^{th} shape parameter differential.

As previously mentioned, $\frac{\partial \mathbf{x}_k}{\partial \mathbf{p}_i}$ and $\frac{\partial \mathbf{y}_k}{\partial \mathbf{p}_i}$ are both scalars and depend on $\mathbf{p}$ and $\mathbf{q}$ by means of Ξ_1, Ξ_2 and Ξ_3. Reevaluating the Jacobian of the warp for the shape parameters only requires evaluating eqs.43 and multiplying it by the precomputed components $\frac{\partial \mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})}{\partial \mathbf{x}_k}$ and $\frac{\partial \mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})}{\partial \mathbf{y}_k}$ as presented in eq.41. The projection matrix, $\mathbf{M}_0$, is constant and can be precomputed since a calibrated camera was assumed.

4.2. Jacobian of The Warp for The Pose Parameters

The same approach is taken to evaluate the Jacobian of the warp for the pose parameters, that is given by

$$\frac{\partial \mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})}{\partial \mathbf{q}} = \sum_{k=1}^{v} \left[\frac{\partial \mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})}{\partial \mathbf{x}_k} \frac{\partial \mathbf{x}_k}{\partial \mathbf{q}} + \frac{\partial \mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})}{\partial \mathbf{y}_k} \frac{\partial \mathbf{y}_k}{\partial \mathbf{q}} \right]. \tag{44}$$

A chain rule decomposition is used and the new terms $\frac{\partial \mathbf{x}_k}{\partial \mathbf{q}_j}$ and $\frac{\partial \mathbf{y}_k}{\partial \mathbf{q}_j}$, again both scalars, are found by combining eq.8 with eq.1, leading to

$$\begin{bmatrix} w\mathbf{x}_k \\ w\mathbf{y}_k \\ w \end{bmatrix} = \mathbf{M}_0 \begin{bmatrix} s_0^{x_k} + \sum_{i=1}^{n} p_i \phi_i^{x_k} + q_j \psi_j^{x_k} + \sum_{i \neq j} q_i \psi_i^{x_k} + s_\psi^{x_k} \\ s_0^{y_k} + \sum_{i=1}^{n} p_i \phi_i^{y_k} + q_j \psi_j^{y_k} + \sum_{i \neq j} q_i \psi_i^{y_k} + s_\psi^{y_k} \\ s_0^{z_k} + \sum_{i=1}^{n} p_i \phi_i^{z_k} + q_j \psi_j^{z_k} + \sum_{i \neq j} q_i \psi_i^{z_k} + s_\psi^{z_k} \\ 1 \end{bmatrix}. \tag{45}$$

In the same way, $\frac{\partial \mathbf{x}_k}{\partial \mathbf{q}_j} = \frac{\partial}{\partial \mathbf{q}_j}(\frac{w\mathbf{x}_k}{w})$ and $\frac{\partial \mathbf{y}_k}{\partial \mathbf{q}_j} = \frac{\partial}{\partial \mathbf{q}_j}(\frac{w\mathbf{y}_k}{w})$, resulting in

$$\frac{\partial \mathbf{x}_k}{\partial \mathbf{q}_j} = \frac{\xi_4 \Xi_6 - \Xi_4 \xi_6}{(\Xi_6)^2} \quad \text{and} \quad \frac{\partial \mathbf{y}_k}{\partial \mathbf{q}_j} = \frac{\xi_5 \Xi_6 - \Xi_5 \xi_6}{(\Xi_6)^2} \tag{46}$$

with $j = 1, \ldots, 6$ and $k = 1, \ldots, v$. The scalar terms $\xi_4, \xi_5, \xi_6, \Xi_4, \Xi_5, \Xi_6$ are also defined in Appendix A. Just like in section 4.1, the terms $\frac{\partial \mathbf{x}_k}{\partial \mathbf{q}_j}$ and $\frac{\partial \mathbf{y}_k}{\partial \mathbf{q}_j}$ depend both on $\mathbf{p}$ and $\mathbf{q}$ by means of Ξ_4, Ξ_5 and Ξ_6.

Summarizing, both the Jacobians of the warp depend on the current shape and pose parameters, so they are required to be recomputed at every iteration. However, both common components $\frac{\partial \mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})}{\partial \mathbf{x}_k}$ and $\frac{\partial \mathbf{W}(\mathbf{x_p}, \mathbf{p}, \mathbf{q})}{\partial \mathbf{y}_k}$ depend only on the configuration of the projected base mesh, $s_{0_\mathbf{p}}$, and thus can be precomputed and efficiently stored as sparse matrices, reducing the overall computation. At the fitting stage only the computation of $\frac{\partial \mathbf{x}_k}{\partial \mathbf{p}_i}$, $\frac{\partial \mathbf{y}_k}{\partial \mathbf{p}_i}$, $\frac{\partial \mathbf{x}_k}{\partial \mathbf{q}_j}$ and $\frac{\partial \mathbf{y}_k}{\partial \mathbf{q}_j}$, is required, being all scalar values.

5. The Initial Estimate

The 2.5D AAM requires a rough head pose estimation to establish the initial 3D pose parameters $\mathbf{q}$. From the monocular point of view, estimate the head pose consists on recovering the camera position and relative orientation to a known set of 3D points. In this work the 6DOF pose parameters are estimated using a combination of Adaboost [36] face detection with the Pose from Orthography and Scaling with ITerations (POSIT) [41]. The POSIT algorithm estimates the 6DOF given a set of 3D points (a rigid model) and corresponding 2D image projections. The base mesh s_0 is used as the required 3D rigid model and the 2D correspondences are given by the base mesh projection $s_{0\mathbf{p}}$, scale adjusted to the average AdaBoost detection.

Figure 8 shows the different coordinate frames involved in the 2.5D AAM. The camera, the current head position and the base pose referential are shown. The base pose reference, $\mathbf{R}_0, \mathbf{t}_0$, in homogeneous coordinates and represented as $\mathbf{T}_0$, is established during the AAM building process where the training shapes are all aligned into s_0. The pose estimated by the combination of face detection and POSIT is represented as $\mathbf{T}_{POSIT}$. The initial pose $\mathbf{T}_{AAM}$ is the rigid transformation between the base pose reference and the current head position. Note that the AAM fitting solves the pose parameters w.r.t. the updated base pose referential $(s_0 + s_\psi)$ and not to the camera (applying a further base pose transformation is required to get the 3D mesh points w.r.t the camera frame).

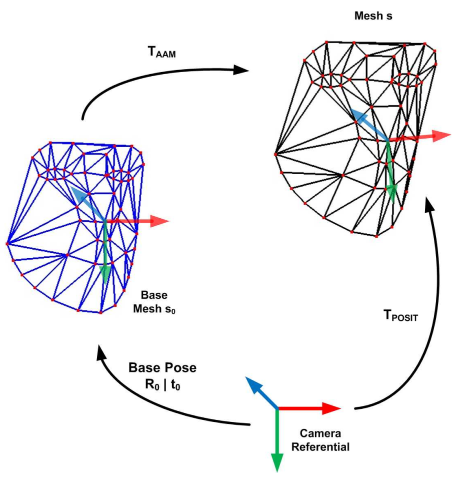


Figure 8: The figure shows the coordinate frames involved in the 2.5D AAM. The base pose $\mathbf{T}_0$ is the transformation between the camera and the base mesh s_0, the $\mathbf{T}_{POSIT}$ is the transformation that results from applying POSIT algorithm and $\mathbf{T}_{AAM}$ is the initial transformation required to startup the fitting algorithm. Note that the AAM fitting solves the pose parameters w.r.t. the updated base pose referential and not to the camera.

As shown in figure 8 the reference frames follow the relationship, $\mathbf{T}_0\mathbf{T}_{AAM} = \mathbf{T}_{POSIT}$, that solving for $\mathbf{T}_{AAM}$, gives

$$\mathbf{T}_{AAM} = \mathbf{T}_0^{-1}\mathbf{T}_{POSIT}. \tag{47}$$

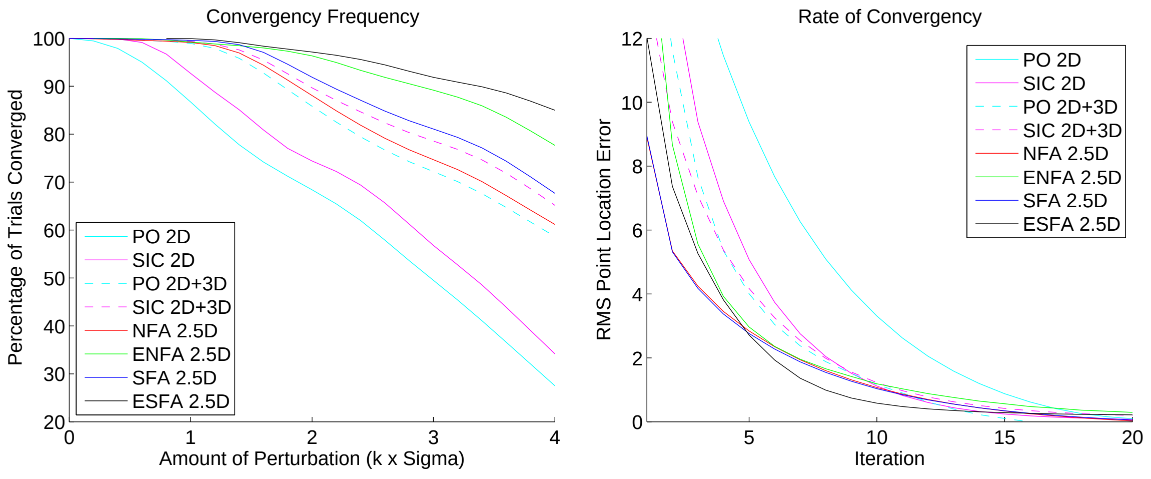
6. Experimental Results

The 3D shape model (PDM) can be acquired by several ways such as using laser range scans, time-of-flight cameras (ToF), Structure from Motion (SfM) techniques and of course multi-camera networks. The 3D PDM in this work was built using a fully calibrated stereo system where the 2D shape on each view was extracted by fitting a 2D AAM [4] with $v = 58$ landmarks (see supplementary material section for details). For evaluation purposes a 2.5D AAM was constructed from a set of 20 individuals collected from our institution. A total of 20 images for each individual (10 left + 10 right) exhibiting several expressions and head poses were used in both shape and texture model building process, as described in section 2. The 2.5D AAM held $n = 12$ shape parameters, $m = 79$ eigenfaces and the projected base mesh has 68970 gray level pixels (i.e. the figure 3-a has size 285×242 pixels).

This evaluation compares the projective 2.5D AAM (NFA, SFA, ENFA, and ESFA algorithms) against the state-of-the-art 2D AAM algorithms (Project Out - PO [4] and Simultaneous Inverse Compositional - SIC [42]) and the combined 2D+3D AAM [21] (2D+3D Project Out and 2D+3D Simultaneous Inverse Compositional). Briefly, the 2D+3D AAM [21] uses two shape models: a 2D PDM and a 3D affine PDM built from Non-Rigid Structure-from-Motion (NRSfM). The optimization goal has two main parts (see eq.39 from [21]): the first part deals with pixel intensity matching by optimizing a standard 2D AAM (shape, similarity and appearance parameters) while the second part is a (heavily weighted) soft constraint that enforces the matching between a 3D PDM (scaled orthographic) projection and the current 2D model instance. This constraint ensures that the 2D model deforms according to a valid 3D face projection. The main differences between the 2.5D model and the 2D+3D AAM are: **(1)** The camera projection models. The 2.5D AAM uses a full perspective projection (allowing to retrieve Euclidean metric 3D shapes) whereas the 2D+3D AAM uses a scaled orthographic projection model. **(2)** The 2.5D model is less complex, using just a single PDM instead of two, and it does not require NRSfM techniques. **(3)** Finally, the 2D+3D AAM requires to manually tune the weight parameter K that balances the two main terms.

To effectively compare the 2.5D AAM approach with the 2D+3D AAM an additional 3D (affine) PDM, built from NRSfM, is required. The NRSfM data consists in short video sequences (around 200 frames) taken from the same 20 individuals exhibiting several facial expressions and pose changes. The 2D SIC algorithm was used to fit all the sequences (around 4000 frames in total). Several well known NRSfM algorithms were tested, namely the Xiao-Kanade's method [22][4] (the same approach used in 2D+3D AAM), the Torresani et al. technique that models the shape by a Linear Dynamical System [44] and the NRSfM that uses Discrete Cosine Transform (DCT) basis [45]. In the 2D+3D AAM experiments it was used the state-of-art NRSfM-DCT technique [45] as it proves to be the most reliable in the conducted experiments. Note that, an extra Procrustes alignment and PCA are required

[4]Code provided by Vincent's Structure from Motion Toolbox [43]



(a) Convergence Frequency (b) Rate of Convergence

Figure 9: Robustness fitting and convergence comparison between 2.5D, 2D+3D [21], 2D algorithms [4]. Best viewed in color.

since no standard basis are given.

6.1. Fitting Robustness and Rate of Convergence

To evaluate the fitting robustness and the rate of convergence of the proposed solutions, the performance evaluation scheme presented in [4][21] was adopted. Figure 9 shows the results obtained by comparing the fitting robustness and rate of convergence of all the non-robust 2.5D algorithms (NFA, SFA, ENFA, ESFA), the 2D+3D algorithms (PO 2D+3D, SIC 2D+3D) and the standard 2D algorithms (PO 2D, SIC 2D).

These experiments measure the performance of the algorithms in two ways: (1) the average frequency of convergence i.e. the number of times each algorithm has converged vs. initial perturbation; (2) the average rate of convergence i.e. the 2D Root Mean Square (RMS) error in the mesh point location vs. iteration number (if convergence was accomplished). For these experiments, each AAM was perturbed from a set of ground truth parameters using independent Gaussian distributions with variance equal to a multiple of a given eigenvalue mode, and tested for convergence. Formally, the parameters disturbance at each experiment was given by

$$\mathbf{p} = \mathbf{p}_{GT} + \mathcal{N}(\mathbf{0}, k\sigma_{\mathbf{p}}) \quad (48)$$

$$\boldsymbol{\lambda} = \boldsymbol{\lambda}_{GT} + \mathcal{N}(\mathbf{0}, k\sigma_{\boldsymbol{\lambda}}) \quad (49)$$

with an increasing factor $k =]0, 0.1, 0.2, \ldots, 3.9, 4]$. The $\sigma_{\mathbf{p}}$ and $\sigma_{\boldsymbol{\lambda}}$ are the standard deviations from the shape and appearance parameters, respectively. The variances $\sigma^2_{\mathbf{p}}$ and $\sigma^2_{\boldsymbol{\lambda}}$ were estimated at the model building process when applying PCA to both shape and texture models.

The ground truth data was generated using the same AAM by a combination of tracking (say fitting in every frame) / manual initialization / visual confirmation on several small sequences taken from each individual. A subset of 20 random selected frames, from each sequence, was used for further testing, accounting a total of 400 frames. For each testing frame a set of 20 trials was generated by perturbing the shape and appearance parameters simultaneously from the ground-truth (20 trials $\times$ 40 noise increasing perturbations experiments per test image). All the algorithms were executed and their convergence ability was evaluated by comparing the final 2D RMS error shape with the ground-truth. A threshold of 1.0 RMS pixels was used to define convergence.

Analyzing figure 9, it can be concluded that both 2.5D and 2D+3D fitting algorithms are more robust than 2D algorithms (PO and SIC) and they converge faster, taking fewer iterations to converge. The 3D PDM is inherently higher dimensional than the 2D PDM, however, it uses less 3D shape parameters than the 2D PDM to represent the same visual phenomenon (our PDM has only 12 shape parameters). The 3D PDM is also less prone to local minima because a 2D model can easily generate physically unfeasible shapes, i.e. spanning the 2D PDM parameters can produce a shape that is not even possible, as described in [22][21]. Figure 9 also shows that our projective 2.5D AAM performs better than the 2D+3D versions[5].

[5]Notice that the methods PO 2D+3D, NFA, ENFA (normalization versions) and SIC 2D+3D, SFA, ESFA (simultaneous versions) should be compared among themselves due to its optimization strategies similarities,

Besides the full perspective model addition, the 2.5D model outperforms the 2D+3D versions, as it has the following advantages: **(1)** The 2.5D AAM is less dimensional, so less prone to local minima, e.g. the NFA solves $(n+6)$ parameters whereas the PO 2D+3D solves $(n_{2D}+4+n+6)$, namely the 2D shape parameters (n_{2D}), the 4 similarity parameters, the 3D shape parameters (n) and the 6 scaled orthographic camera parameters (the scale, 3D rotations and the 2D translations). **(2)** The optimization uses more accurate gradients. The forwards additive approaches when compared with the inverse compositional (in particular the 2D model from the 2D+3D AAM) produce less second order terms in the Taylor series approximation (eq.13). The main optimization neglects more terms if an inverse compositional method is used [34]. This means that our forwards additive 2.5D use gradients that are closer to the "true" gradients, being therefore more accurate and take less iterations to converge (as shown by figure 9-b). **(3)** As previously mentioned, the 2D+3D AAM requires tuning a constant K that weights the 3D affine projection constraint. When K is too small (soft constant) the combined model fits a 2D and a 3D shape independently (the 3D projection and the 2D model do not converge). However, if K is set to be too large, e.g. 10^6, the gradient descent updates (times the inverse of the Hessian) are too small, and the model requires a lot more iterations to converge. In all experiments K was set to $K = 10000$. The 2.5D model does not have this weighting issue. **(4)** The 2D+3D AAM requires to compute Jacobians for the constraints w.r.t. the 2D shape and pose parameters ($\frac{\partial Ft_i}{\partial \mathbf{p}}\mathbf{J_p}$ and $\frac{\partial Ft_i}{\partial \mathbf{q}}\mathbf{J_q}$), which are not required for the 2.5D model. Furthermore, these Jacobians are numerically estimated. **(5)** A minor, but still an advantage, is that our model do not require to inverse compose the warp at each iteration since the parameters update is additive. Both 2D versions and consequently the 2D+3D versions require to inverse compose the warp at each iteration, which still is an approximation (averaging the neighbor triangles) since no true inverse exists [4]. **(6)** Finally, the 2.5D AAM is a lot more simple and easier to implement when compared to the 2D+3D model.

The results also show that the efficient versions (ENFA, ESFA) perform even better than

i.e. to project out the appearance variation optimizing only the shape and pose or optimizing all parameters at once, respectively.

the standard formulations. The main reason for this performance increase is the reduced noise influence that comes out from avoiding the reevaluation of the gradients of the input image in each iteration, as described in section 3.3. The Efficient-SFA, that searches simultaneously for all the parameters, has proved to be the best algorithm w.r.t. convergence speed showing high fitting success rates even from far initial estimates.

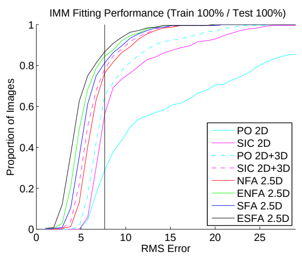
6.2. Performance in Unseen Data

The AAM is a generative (holistic) method as it models the appearance of all image pixels within the face. By synthesizing the expected appearance template it achieves a high registration accuracy on the dataset it was trained for but it performs poorly in unseen data (individuals not captured by the texture PCA). If the appearance of a target individual does not lie in the subspace spanned by $\mathbf{A}_i(\mathbf{x_p})$, the AAM can not generate a valid template and the model fitting will not converge.

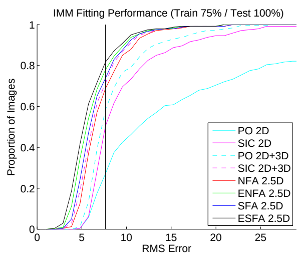
The AAM fitting performance in unseen data was evaluated by running a series of experiments, by changing the amount of training images in the model building process. The IMM [46] database was used, as it consists of 240 annotated images (58 ground truth landmarks) of 40 different human faces presenting different head pose, illumination and facial expression. All the fitting algorithms that previously appeared in section 6.1 were used in this evaluation, namely PO 2D, SIC 2D, PO 2D+3D, SIC 2D+3D, NFA 2.5D, SFA 2.5D, ENFA 2.5D and SFA 2.5D.

Four main experiments were conducted, training all the algorithms (building the AAMs) with the full sized dataset (100% - 240 images), then 75% (180 images), 50% (120 images) and finally only 25% (60 images) of the dataset. Performing a fair comparison requires that all the runs must be evaluated on the same testing data, therefore the full database has been used.

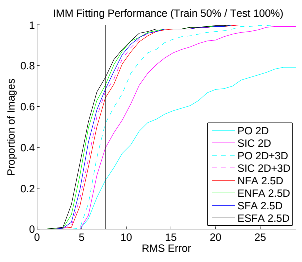
Figures 10-(a)(b)(c)(d) show the fitting performance curves for these four experiments, respectively. These are standard curves that show the percentage of faces that converge with less or equal Root Mean Square (RMS) error amount. As higher the curve is, more percentage of images converge with a given amount of error, hence better the fitting algo-



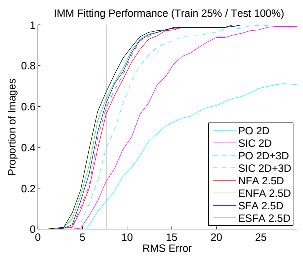
(a) Train 100%, test full dataset



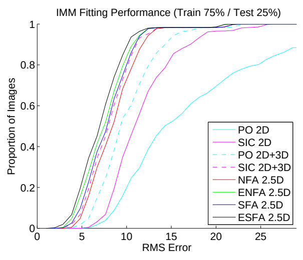
(b) Train 75%, test full dataset



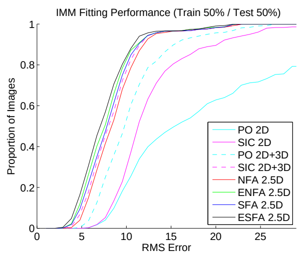
(c) Train 50%, test full dataset



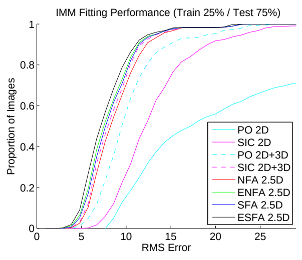
(d) Train 25%, test full dataset



(e) Train 75%, test unseen 25%



(f) Train 50%, test unseen 50%



(g) Train 25%, test unseen 75%

Reference 7.5 RMS	PO 2D	SIC 2D	PO 2D+3D	SIC 2D+3D	NFA 2.5D	ENFA 2.5D	SFA 2.5D	ESFA 2.5D
(a) 100%	28.7	56.2	65.4	78.3	76.2	84.6	81.2	**86.7**
(b) 75%	27.1	50.4	58.3	71.2	68.8	77.9	73.8	**81.7**
(c) 50%	23.8	39.6	51.7	67.1	64.2	70.4	68.3	**74.2**
(d) 25%	13.6	23.3	39.6	59.2	56.2	63.7	61.7	**67.1**

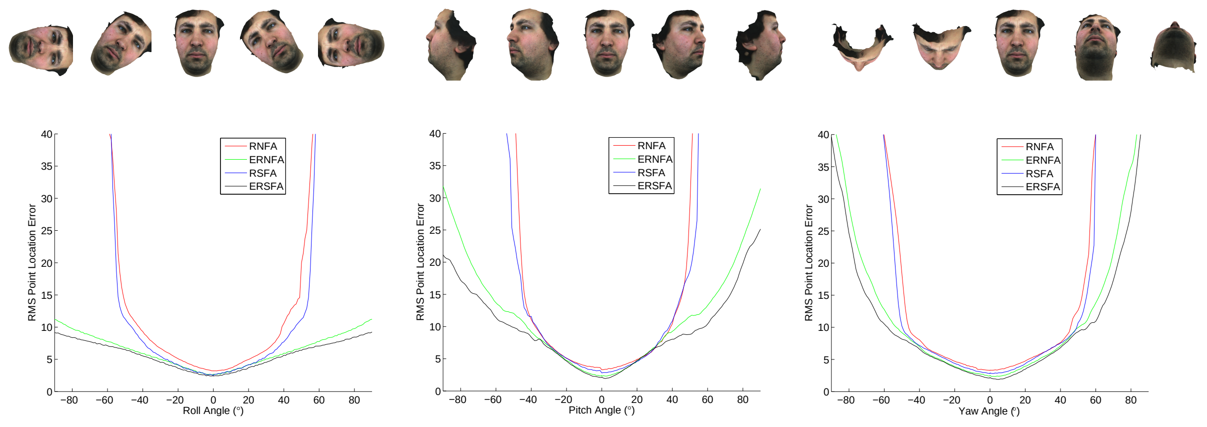
Figure 10: Fitting performances curves on the IMM [46] database using 100%, 75%, 50% and 25% of training images, respectively. Figures (a)(b)(c)(d) show fitting curves for all AAM fitting algorithms using the full dataset as test data. (e)(f)(g) show results for the 75%, 50% and 25% models on just the unseen remainder images of the dataset, respectively. Top images show fitting examples from the IMM database using the ESFA algorithm. The bottom table shows quantitative values taken by sampling the graphics (a)(b)(c)(d) using a fixed RMS error amount (7.5 pixels - represented as the vertical line). Each table entry show how many percentage of images converge with less or equal RMS error that the reference. Note that the table only applies to charts where the same amount of testing data was used (the full dataset).

rithm. Additionally, figures 10-(e)(f)(g), show the fitting performance curves for the 75%, 50% and 25% (training data) models on the respective unseen portion of the dataset (curves with only unseen test data). The table in the same figure shows quantitative values taken by sampling the graphics using a fixed RMS error amount (7.5 pixels - represented as the vertical line in the graphics). This table only applies to the experiments where the same testing data was used.

As expected, all the methods reveals a fitting performance decrease (less images converge for the same RMS value) as the appearance representation power decrease. The relative performance between all the methods are conform to section 6.1. The 2D models have the lower performance (where the Project Out performs the worst), followed by the combined 2D+3D model and then our projective 2.5D versions, where the efficient algorithms perform the better. Accordingly, the simultaneous versions perform better than the error normalization versions mainly due to their improved search strategy (all parameters at once). The overall results show that using a 3D PDM projection effectively increases the performance in unseen data.

6.3. Robust Methods Evaluation

The robust fitting methods proposed in this work intend to improve the performance w.r.t. self occlusion due to 3D head motion. To evaluate these algorithms, namely the RNFA, the RSFA and the efficient versions ERNFA and ERNFA, three synthetic sequences were created. A set of images with an individual standing in near frontal position was used. The current 3D mesh location was found by fitting the 2.5D AAM using ESFA. Then, ranging the 3D mesh from $-90°$ to $90°$ degrees in both roll, pitch and yaw angles, using one degree of resolution, the fixed appearance image was projected into the camera and stored (figure 11-top). Finally, all the robust fitting algorithms were evaluated using these sequences, starting from the frontal position. In all the algorithms the scale parameters, $\sigma_{\mathbf{x_p}}$, were estimated from the fitting error MAD. Figure 11-bottom shows the RMS error in point location for all the algorithms. Once again the Efficient versions of the algorithms (ERNFA and ERSFA) outperform their standard versions (RNFA and RSFA). Also, the ERSFA



(a) Roll (b) Pitch (c) Yaw

Figure 11: Robust algorithms evaluation on the synthetic sequences at top figure. The graphics show the RMS error due to roll, pitch and yaw angles ranging from $-90°$ to $90°$, respectively.

performs slightly better that the ERNFA, as expected, due to the parameters search strategy. These experiments show that, using the efficient algorithms, the model can successfully deal with rotations of almost $\pm 90°$ in roll, pitch and yaw angles, respectively.

6.4. Results on the BU-4DFE Dataset

This section evaluates the quality of the 3D recovered shape when using the 2.5D AAM. The Binghamton University 3D Dynamic Facial Expression Database (BU-4DFE) [47] was used for this evaluation process. The BU-4DFE dataset includes high resolution 3D dense reconstructions of video sequences of several individuals showing the six prototypic facial expressions [48] namely, anger, disgust, happiness, fear, sadness, and surprise. The 3D facial expressions were captured at 25 frames per second where each expression sequence contains about 100 frames (resolution of 1040×1392 per frame).

Due to the generative nature of the AAM, a new BU-4DFE tuned model must be built to run these experiments. To fit every frame of the database the AAM should hold as much shape variation as possible. To accomplished this the training images were composed by the most emotion expressive images of the testing set. These training images were hand annotated using also the 58 landmarks scheme ($v = 58$). Holding 95% of the shape and appearance variance produces a 2.5D AAM with 19 shape parameters, ($n = 19$), and 87 eigenfaces, ($m = 87$). The projected base mesh width was set to 300 pixels, as described in the 3D model building process in supplementary material, resulting on a total of 100500 gray level pixels used by the appearance model.

In this section, only the ESFA algorithm has been used, since it was proved previously to be the most accurate. A subset of the BU-4DFE dataset, consisting in 7 males and 7 females, forming a total of around 8400 frames were used in this evaluation. The ESFA algorithm was applied on every frame of each sequence for all the testing subjects, and the RMS error between the current PDM shape s (the shape that the model fits for) and the ground truth extracted from the BU-4DFE dataset was evaluated.

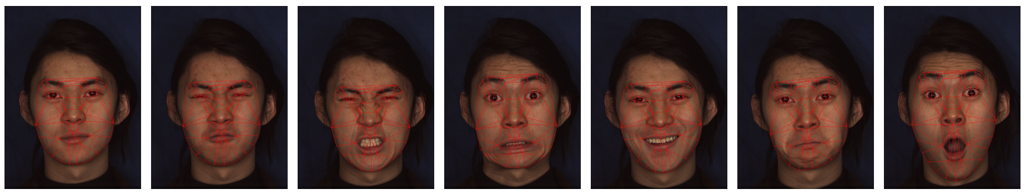
The shape RMS error is given by

$$e_{\text{RMS}}(s) = \sqrt{\frac{1}{v}\sum_{i=1}^{v}\left(s^{x_i} - s_{gt}^{x_i}\right)^2 + \left(s^{y_i} - s_{gt}^{y_i}\right)^2 + \left(s^{z_i} - s_{gt}^{z_i}\right)^2} \tag{50}$$

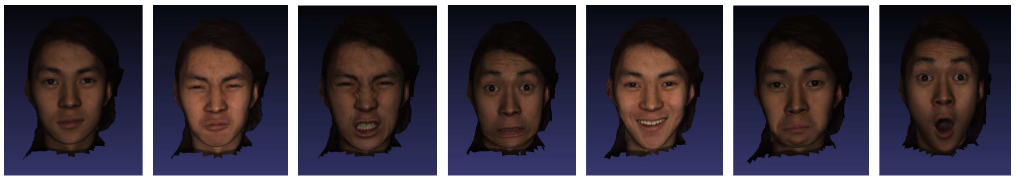
where the ground truth shape, s_{gt}, was extracted from the dense reconstruction by lookup the 3D depth from the 2D image projections found by the 2.5D AAM.

Figure 12 shows examples of the AAM fitting, the corresponding 3D dense reconstruction ground truth and a graphic showing the RMS shape error over time for each emotion sequence (for a single test subject). The evaluation shows that globally during the entire sequence, the fitting error stays low, exhibiting an average error of around 5mm (in 3D space). Typically, in the captured facial expression sequences of the BU-4DFE dataset, each individual starts from a neutral expression, exhibits the emotion until its maximum intensity and then goes back to the neutral state. The graphic shows that the RMS error match this behavior, i.e. the AAM has a lower shape fitting error during the begin and at end of the sequences when the individual displays the neutral emotion. The results also show that the surprise facial expression is the one that holds more fitting error, mainly because it is the emotion that more deforms the face from the neutral state.

Table 1 displays the mean and standard deviations of the RMS shape error over the entire testing subset of the BU-4DFE.



(a) Neutral (b) Angry (c) Disgust (d) Fear (e) Happy (f) Sad (g) Surpr.



(h) Neutral (i) Angry (j) Disgust (k) Fear (l) Happy (m) Sad (n) Surpr.

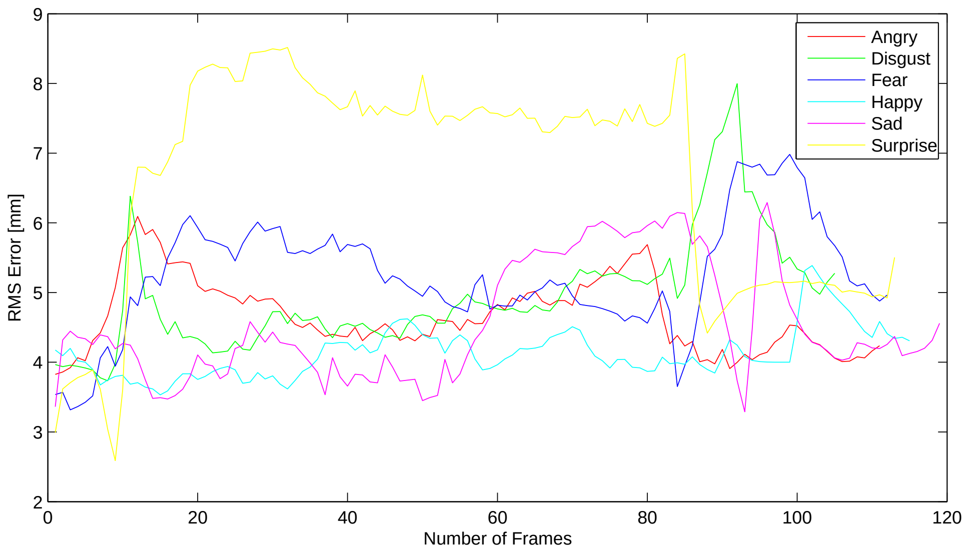


Figure 12: Evaluation of the 3D recovered shape when using the 2.5D AAM. The top a)-g) figures show examples of AAM fitting on a test subject of the BU-4DFE database exhibiting the six basic emotions plus the neutral one. Images h)-n) shows the corresponding 3D dense reconstruction provided by the database. The ground truth, s_{gt}, used in all evaluations is a sparse shape that results from retrieving the 3D data from the dense reconstruction on the 2D projections points found by the AAM (the red mesh at top figures). The bottom graphic show the RMS shape error during each of the facial expressions sequences of the testing individual shown in the top images. The RMS error units are in mm. A 2.5D AAM fitting video showing some examples of BU-4DFE dataset can be seen at http://www.isr.uc.pt/~pedromartins/Videos/AAM25D.

	Angry	Disgust	Fear	Happy	Sad	Surprise	**Overall**
avg	4.51	4.82	4.92	4.78	4.64	6.28	4.99
std	0.42	0.83	0.92	0.36	0.82	1.67	0.83

Table 1: The RMS shape fitting error over a subset of the BU-4DFE dataset consisting of 14 individuals (7 males and 7 females). About 8400 frames were used. The table show the mean and standard deviation found for each facial expression sequence and also for the entire set(overall). The units are in mm.

6.5. Tracking Performance

The tracking performance is evaluated on the challenging FGNet Talking Face (TF) [49] video sequence that holds 5000 frames of video of an individual engaged in a conversation. The full sequence is annotated using 68 landmarks (2D ground truth). Just like in previous sections (6.1 and 6.2), all AAM algorithms are used, namely the PO 2D, SIC 2D, PO 2D+3D, SIC 2D+3D, NFA, SFA, ENFA and SFA. A minor difference from the previous experiments is that a few annotated frames from the TF sequence were added in each AAM so that the appearance model $\mathbf{A}_i(\mathbf{x_p})$ can now include the new individual.

The figure 13 shows the RMS fitting error for all the evaluated methods. Since we are using a 58 landmark scheme and the TF uses 68, the error was only measured over the correspondent landmarks. The quantitative values on the legend box are the mean and standard deviation values for the RMS error.

Globally, as expected, all the 2.5D algorithms (NFA, ENFA, SFA and ESFA) perform better than the 2D algorithms (especially when exists some degree of head pose variation) and slightly better than the 2D+3D algorithm, confirming their relative performance. Again, the efficient versions also express a performance advantage over all the others.

6.6. Computational Performance

Table 2 shows a comparison, in computational cost, between all the evaluated algorithms during the entire section 6. The tables shows approximated fitting times per iteration using a MatLab implementation on a 3GHz Intel i7 CPU with 4GB of RAM running Fedora 14 OS. All the AAM use the same settings mentioned on section 6.1.

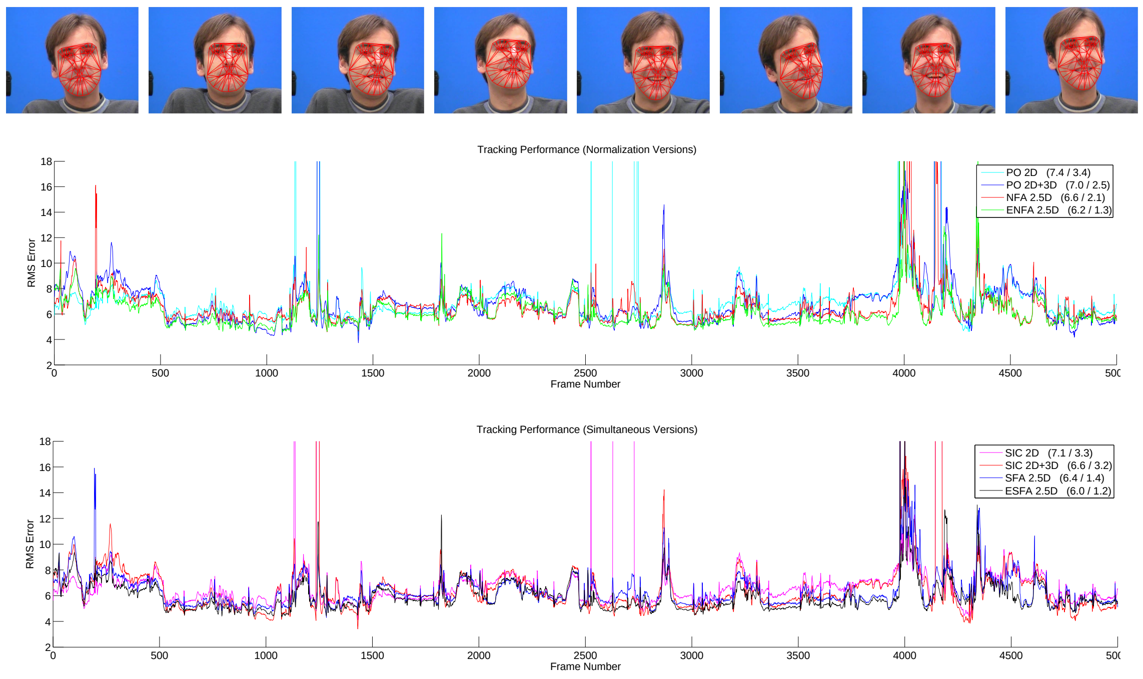


Figure 13: RMS shape error on the Talking Face video sequence. The top images show ESFA fitting examples. The values on legend box are the mean and standard deviation RMS errors, respectively. Best viewed in color.

	Normalization Versions				**Simultaneous Versions**			
	PO 2D	PO 2D+3D	NFA 2.5D	ENFA 2.5D	SIC 2D	SIC 2D+3D	SFA 2.5D	ESFA 2.5D
Time per iteration (ms)	310	340	1780	760	460	550	1820	780

Table 2: Fitting times by iteration on the evaluated algorithms. The present times are in ms taken using a MatLab implementation. Note that the 2.5D methods even being slower, they require less iterations to converge.

The 2D Projected-Out is probably the fastest approach introduced so far, where both the Jacobian and the Hessian matrices are constant and can be precomputed. The 2D + 3D PO only requires to reevaluate the Jacobians for the constraints and parts of the Hessian (most part is constant). The simultaneous extensions (SIC 2D and SIC 2D+3D) are much slower because they must evaluate the SD images, the Hessian and its inverse on a larger set of parameters that now include the appearance parameters. As shown in algorithms 1, 2, 5 and 6 the 2.5D algorithms need to perform image warping (it takes around 1200ms and 220ms in the standard and efficient versions, respectively), recompute the SD images (around 400ms) and the Hessian. Even being slower, they require less iterations to converge as shown in figure 9-b. However a C/C++ version of ESFA achieves near real-time performance (around 10 fps - using a base mesh with almost 70K pixels). Additional speed up can be achieved by reducing the base mesh size.

7. Conclusions

In this paper we presented a novel formulation for 3D facial image alignment from single view 2D images through a 2.5D AAM. The major contribution of the paper lies on the use of a 2.5D AAM that combines a 3D metric PDM with a full perspective projection model that defines the 2D appearance. The 2.5D AAM is able to recover 3D Euclidean shapes by assuming a calibrated camera. Two algorithms and computational efficient approximations are proposed, both based on the Lucas and Kanade framework: the Simultaneous Forwards Additive (SFA) and the Normalization Forwards Additive (NFA). The SFA, when compared with NFA, is the most accurate algorithm and also the most computationally expensive. Their efficient versions have shown a substantial improvement in the fitting performance, being more robust to noise and able to converge from far initial estimates, requiring less computational effort. To make the model able to deal with self or partial occlusion, robust extensions to SFA and NFA were also proposed. Again, their efficient approximations perform much better that the basic versions. Several performance evaluations carried out on real an synthetic data demonstrated that the 2.5D AAM algorithms outperform both the combined 2D+3D AAM and the traditional 2D AAM algorithms and accurately handle face pose variations. Finally, the quality of the 3D retrieved shape was also evaluated. The performed tests on the BU-4DFE [47] database show that the 2.5D AAM is an effective method to recover the 3D Euclidean shape.

Acknowledgements

This work was supported by the Portuguese Science Foundation (FCT) by the project "Dinâmica Facial 4D para Reconhecimento de Identidade" with grant PTDC/EIA-CCO/108791/2008. Pedro Martins and Rui Caseiro also acknowledge the FCT for support through the grants SFRH/BD/45178/2008 and SFRH/BD74152/2010, respectively.

Appendix A. The Jacobian of The Warp Partial Differentials

Defining the elements $m_{0_{ij}}$ of the base projection matrix $\mathbf{M}_0$ as

$$\underbrace{\begin{bmatrix} m_{0_{11}} & m_{0_{12}} & m_{0_{13}} & m_{0_{14}} \\ m_{0_{21}} & m_{0_{22}} & m_{0_{23}} & m_{0_{24}} \\ m_{0_{31}} & m_{0_{32}} & m_{0_{33}} & m_{0_{34}} \end{bmatrix}}_{\mathbf{M}_0} = \mathbf{K} \begin{bmatrix} \mathbf{R}_0 & \mathbf{t}_0 \end{bmatrix} \tag{A.1}$$

the quantities $\xi_1, \ldots, \xi_6$ and $\Xi_1, \ldots, \Xi_6$ are scalar values given by

$$\begin{aligned} \xi_1 &= m_{0_{11}}\phi_i^{x_k} + m_{0_{12}}\phi_i^{y_k} + m_{0_{13}}\phi_i^{z_k} & \xi_4 &= m_{0_{11}}\psi_j^{x_k} + m_{0_{12}}\psi_j^{y_k} + m_{0_{13}}\psi_j^{z_k} \\ \xi_2 &= m_{0_{21}}\phi_i^{x_k} + m_{0_{22}}\phi_i^{y_k} + m_{0_{23}}\phi_i^{z_k} & \xi_5 &= m_{0_{21}}\psi_j^{x_k} + m_{0_{22}}\psi_j^{y_k} + m_{0_{23}}\psi_j^{z_k} \\ \xi_3 &= m_{0_{31}}\phi_i^{x_k} + m_{0_{32}}\phi_i^{y_k} + m_{0_{33}}\phi_i^{z_k} & \xi_6 &= m_{0_{31}}\psi_j^{x_k} + m_{0_{32}}\psi_j^{y_k} + m_{0_{33}}\psi_j^{z_k} \end{aligned} \tag{A.2}$$

$$\begin{bmatrix} \Xi_1 \\ \Xi_2 \\ \Xi_3 \end{bmatrix} = \mathbf{M}_0 \begin{bmatrix} s_0^{x_k} + p_i\phi_i^{x_k} + \sum_{j\neq i}^{n} p_j\phi_j^{x_k} + \sum_{j=1}^{6} q_j\psi_j^{x_k} + s_\psi^{x_k} \\ s_0^{y_k} + p_i\phi_i^{y_k} + \sum_{j\neq i}^{n} p_j\phi_j^{y_k} + \sum_{j=1}^{6} q_j\psi_j^{y_k} + s_\psi^{y_k} \\ s_0^{z_k} + p_i\phi_i^{z_k} + \sum_{j\neq i}^{n} p_j\phi_j^{z_k} + \sum_{j=1}^{6} q_j\psi_j^{z_k} + s_\psi^{z_k} \\ 1 \end{bmatrix} \tag{A.3}$$

$$\begin{bmatrix} \Xi_4 \\ \Xi_5 \\ \Xi_6 \end{bmatrix} = \mathbf{M}_0 \begin{bmatrix} s_0^{x_k} + \sum_{i=1}^{n} p_i\phi_i^{x_k} + q_j\psi_j^{x_k} + \sum_{i\neq j}^{6} q_i\psi_i^{x_k} + s_\psi^{x_k} \\ s_0^{y_k} + \sum_{i=1}^{n} p_i\phi_i^{y_k} + q_j\psi_j^{y_k} + \sum_{i\neq j}^{6} q_i\psi_i^{y_k} + s_\psi^{y_k} \\ s_0^{z_k} + \sum_{i=1}^{n} p_i\phi_i^{z_k} + q_j\psi_j^{z_k} + \sum_{i\neq j}^{6} q_i\psi_i^{z_k} + s_\psi^{z_k} \\ 1 \end{bmatrix}. \tag{A.4}$$

Appendix B. Details on the Efficient Fitting Algorithms

```
1  Precompute:
2  The 2.5D parametric models
3  Evaluate $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p})}{\partial \mathbf{x}_k}$ and $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p})}{\partial \mathbf{y}_k}$ (fig. 7)
4  Gradients of the template $\nabla A_0(\mathbf{x_p})$
5  Gradients of the Eigen images $\nabla A_i(\mathbf{x_p})$
6  repeat
7      Update pose reference $\Psi(s_\psi)$ with eq.6
8      Warp input image $\mathbf{I}(\mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q}))$
9      Error image $\mathbf{E}(\mathbf{x_p})_{\mathrm{sfa}}$ using eq.18
10     Find Jacobian $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{p}}$ (eq.41)
11     Find Jacobian $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{q}}$ (eq.44)
12     Compute efficient SD images $\mathbf{SD}(\mathbf{x_p})_{\mathrm{esfa}}$
       using eq.29
13     Hessian matrix and its inverse
       $\mathbf{H}_{\mathrm{esfa}} = \sum_{\mathbf{x_p}} \mathbf{SD}(\mathbf{x_p})^T_{\mathrm{esfa}} \mathbf{SD}(\mathbf{x_p})_{\mathrm{esfa}}$
14     Parameters updates
       $\Delta\mathbf{r} = \mathbf{H}^{-1}_{\mathrm{esfa}} \sum_{\mathbf{x_p}} \mathbf{SD}(\mathbf{x_p})^T_{\mathrm{esfa}} \mathbf{E}(\mathbf{x_p})_{\mathrm{sfa}}$
15     Update parameters $\mathbf{r} \leftarrow \mathbf{r} + \Delta\mathbf{r}$
16     Update pose offset $s_\psi \leftarrow s_\psi + \sum_{j=1}^{6} \psi_j \Delta q_j$
17 until $||\Delta r|| \leq \varepsilon$ or max. number of iterations
   reached ;
```

Algorithm 5: Efficient SFA.

```
1  Precompute:
2  The 2.5D parametric models
3  Evaluate $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p})}{\partial \mathbf{x}_k}$ and $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p})}{\partial \mathbf{y}_k}$ (fig. 7)
4  Gradients of the template $\nabla A_0(\mathbf{x_p})$
5  repeat
6      Update pose reference $\Psi(s_\psi)$ with eq.6
7      Warp input image $\mathbf{I}(\mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q}))$
8      Error image $\mathbf{E}(\mathbf{x_p})_{\mathrm{lk}}$ using eq.23
9      Estimate $\boldsymbol{\lambda}$ using eq.25
10     Normalized Error image $\mathbf{E}(\mathbf{x_p})_{\mathrm{nfa}}$
11     Find Jacobian $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{p}}$ (eq.41)
12     Find Jacobian $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{q}}$ (eq.44)
13     Compute efficient SD images $\mathbf{SD}(\mathbf{x_p})_{\mathrm{enfa}}$
       using eq.30
14     Hessian matrix and its inverse
       $\mathbf{H}_{\mathrm{enfa}} = \sum_{\mathbf{x_p}} \mathbf{SD}(\mathbf{x_p})^T_{\mathrm{enfa}} \mathbf{SD}(\mathbf{x_p})_{\mathrm{enfa}}$
15     $\begin{bmatrix} \Delta\mathbf{p} \\ \Delta\mathbf{q} \end{bmatrix} = \mathbf{H}^{-1}_{\mathrm{enfa}} \sum_{\mathbf{x_p}} \mathbf{SD}(\mathbf{x_p})^T_{\mathrm{enfa}} \mathbf{E}(\mathbf{x_p})_{\mathrm{efa}}$
16     Update $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$ and $\mathbf{q} \leftarrow \mathbf{q} + \Delta\mathbf{q}$
17     Update pose offset $s_\psi \leftarrow s_\psi + \sum_{j=1}^{6} \psi_j \Delta q_j$
18 until $\left\| \begin{matrix} \Delta p \\ \Delta q \end{matrix} \right\| \leq \varepsilon$ or max. number of
   iterations reached ;
```

Algorithm 6: Efficient NFA.

```
1  Precompute:
2  The 2.5D parametric models
3  Evaluate $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p})}{\partial \mathbf{x}_k}$ and $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p})}{\partial \mathbf{y}_k}$ (fig. 7)
4  Gradients of the template $\nabla A_0(\mathbf{x_p})$
5  Gradients of the Eigen images $\nabla A_i(\mathbf{x_p})$
6  repeat
7      Update pose reference $\Psi(s_\psi)$ with eq.6
8      Warp input image $\mathbf{I}(\mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q}))$
9      Evaluate triangle visibility
10     Error image $\mathbf{E}(\mathbf{x_p})_{\text{sfa}}$ using eq.18
11     Estimate the weight mask $\rho(\mathbf{E}(\mathbf{x_p})^2_{\text{sfa}})$
12     Find Jacobian $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{p}}$ (eq.41)
13     Find Jacobian $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{q}}$ (eq.44)
14     Compute efficient SD images $\mathbf{SD}(\mathbf{x_p})_{\text{esfa}}$
       using eq.29
15     Weighted Hessian matrix $\mathbf{H}_{\text{ersfa}} =$
       $\sum_{\mathbf{x_p}} \rho(\mathbf{E}(\mathbf{x_p})^2_{\text{sfa}})\mathbf{SD}(\mathbf{x_p})^T_{\text{esfa}}\mathbf{SD}(\mathbf{x_p})_{\text{esfa}}$
16     Parameters updates $\Delta\mathbf{r} =$
       $\mathbf{H}^{-1}_{\text{ersfa}} \sum_{\mathbf{x_p}} \rho(\mathbf{E}(\mathbf{x_p})^2_{\text{sfa}})\mathbf{SD}(\mathbf{x_p})^T_{\text{esfa}}\mathbf{E}(\mathbf{x_p})_{\text{sfa}}$
17     Update parameters $\mathbf{r} \leftarrow \mathbf{r} + \Delta\mathbf{r}$
18     Update pose offset $s_\psi \leftarrow s_\psi + \sum_{j=1}^{6} \psi_j \Delta q_j$
19 until $||\Delta \boldsymbol{r}|| \leq \varepsilon$ or max. number of iterations
   reached ;
```
Algorithm 7: Efficient Robust SFA.

```
1  Precompute:
2  The 2.5D parametric models
3  Evaluate $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p})}{\partial \mathbf{x}_k}$ and $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p})}{\partial \mathbf{y}_k}$ (fig. 7)
4  Gradients of the template $\nabla A_0(\mathbf{x_p})$
5  repeat
6      Update pose reference $\Psi(s_\psi)$ with eq.6
7      Warp input image $\mathbf{I}(\mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q}))$
8      Evaluate triangle visibility
9      Error image $\mathbf{E}(\mathbf{x_p})_{\text{rnfa}}$ using eq.37
10     Estimate the weight mask $\rho(\mathbf{E}(\mathbf{x_p})^2_{\text{rnfa}})$
11     Hessian Appearance $\mathbf{H}_{\text{A}}$ with eq.40
12     Update app. parameters $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \Delta\boldsymbol{\lambda}$
13     Recompute $\mathbf{E}(\mathbf{x_p})_{\text{rnfa}}$ using eq.37
14     Find Jacobian $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{p}}$ (eq.41)
15     Find Jacobian $\frac{\partial \mathbf{W}(\mathbf{x_p},\mathbf{p},\mathbf{q})}{\partial \mathbf{q}}$ (eq.44)
16     Compute efficient SD images $\mathbf{SD}(\mathbf{x_p})_{\text{enfa}}$ using
       eq.30
17     Weighted Hessian matrix $\mathbf{H}_{\text{ernfa}} =$
       $\sum_{\mathbf{x_p}} \rho(\mathbf{E}(\mathbf{x_p})^2_{\text{rnfa}})\mathbf{SD}(\mathbf{x_p})^T_{\text{enfa}}\mathbf{SD}(\mathbf{x_p})_{\text{enfa}}$
18     $\begin{bmatrix} \Delta\mathbf{p} \\ \Delta\mathbf{q} \end{bmatrix} =$
       $\mathbf{H}^{-1}_{\text{ernfa}} \sum_{\mathbf{x_p}} \rho(\mathbf{E}(\mathbf{x_p})^2_{\text{rnfa}})\mathbf{SD}(\mathbf{x_p})^T_{\text{enfa}}\mathbf{E}(\mathbf{x_p})_{\text{rnefa}}$
19     Update $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$ and $\mathbf{q} \leftarrow \mathbf{q} + \Delta\mathbf{q}$
20     Update pose offset $s_\psi \leftarrow s_\psi + \sum_{j=1}^{6} \psi_j \Delta q_j$
21 until $\left\| \begin{matrix} \Delta \boldsymbol{p} \\ \Delta \boldsymbol{q} \end{matrix} \right\| \leq \varepsilon$ or max. number of iterations
   reached ;
```
Algorithm 8: Efficient Robust NFA.

References

[1] T.F.Cootes, G.J.Edwards, C.J.Taylor, Active appearance models, IEEE Trans Pattern Anal Mach Intell 23 (6) (2001) 681–685.

[2] T.F.Cootes, C.J.Taylor, D.H.Cooper, J.Graham, Active shape models-their training and application, Comput Vis Image Underst 61 (1) (1995) 38–59.

[3] T.F.Cootes, C.J.Taylor, Statistical models of appearance for computer vision, Tech. rep., Imaging Science and Biomedical Engineering, University of Manchester (2004).

[4] I.Matthews, S.Baker, Active appearance models revisited, Int J Comput Vis 60 (1) (2004) 135–164.

[5] S.Baker, I.Matthews, Equivalence and efficiency of image alignment algoritms, in: IEEE Conference on Computer Vision and Pattern Recognition, 2001, pp. 1090–1097.

[6] A.Bartoli, Groupwise geometric and photometric direct image registration, IEEE Trans Pattern Anal Mach Intell 30 (12) (2008) 2098–2108.

[7] T.F.Cootes, G.V.Wheeler, K.N.Walker, C.J.Taylor, View-based active appearance models, Image Vis Comput 20 (2002) 657–664.

[8] P.Mittrapiyanuruk, G.N.DeSouza, A.C.Kak, Accurate 3d tracking of rigid objects with occlusion using active appearance models, in: IEEE Workshop on Moition and Video Computing, 2005, pp. 90–95.

[9] F.Dornaika, J.Ahlberg, Fast and reliable active appearance model search for 3d face tracking, in: International Conference on Model-based Imaging, Rendering, Image Analysis and Graphical Special Effects, 2003, pp. 113–122.

[10] C. Butakoff, A. Frangi, Multi-view face segmentation using fusion of statistical shape and appearance models, Comput Vis Image Underst 114 (3) (2010) 311–321.

[11] C.Hu, J.Xiao, I.Matthews, S.Baker, J.Cohn, T.Kanade, Fitting a single active appearance model simultaneously to multiple images, in: British Machine Vision Conference, 2004.

[12] F.Dornaika, J.Ahlberg, Fitting 3d face models for tracking and active appearance model training, Image Vis Comput 24 (2006) 1010–1024.

[13] J.Ahlberg, Candide-3 - an updated parameterized face, Tech. Rep. LiTH-ISY-R-2326, Dept. of Electrical Engineering, Linköping University, Sweden (2001).

[14] S.E.Ayala-Raggi, L.Altamirano-Robles, J.Cruz-Enriquez, Automatic face interpretation using fast 3d illumination-based aam models, Comput Vis Image Underst 115 (2) (2011) 194–210.

[15] A.Sattar, Y.Aidarous, S.Le.Gallou, R.Seguier, Face alignment by 2.5d active appearance model optimized by simplex, in: International Conference on Computer Vision Systems (ICVS), 2007.

[16] A.Sattar, Y.Aidarous, R.Seguier, Gagm-aam: A genetic optimization with gaussian mixtures for active appearance models, in: IEEE International Conference on Image Processing, 2008, pp. 3220–3223.

[17] A.Sattar, R.Seguier, Mvaam (multi-view active appearance model) optimized by multi-objective genetic

algorithm, in: IEEE International Conference on Automatic Face and Gesture Recognition, 2008, pp. 1–8.

[18] V.Blanz, T.Vetter, A morphable model for the synthesis of 3d faces, in: Comput Graph (ACM) SIGGRAPH, 1999, pp. 187–194.

[19] S.Romdhani, T.Vetter, Efficient, robust and accurate fitting of a 3d morphable model, in: IEEE International Conference on Computer Vision, 2003, pp. 59–66.

[20] S.Baker, R.Patil, K.M.Cheung, I.Matthews, Lucas kanade 20 years on: A unifying framework: Part 5, Tech. Rep. CMU-RI-TR-04-64, CMU Robotics Institute (November 2004).

[21] I.Matthews, J.Xiao, S.Baker, 2d vs. 3d deformable face models: Representational power, construction, and real-time fitting, Int J Comput Vis 75 (1) (2007) 93–113.

[22] J.Xiao, J.Chai, T.Kanade, A closed-form solution to non-rigid shape and motion recovery, in: European Conference on Computer Vision, 2004.

[23] S.Baker, R.Gross, I.Matthews, Lucas kanade 20 years on: A unifying framework: Part 3, Tech. Rep. CMU-RI-TR-03-35, CMU Robotics Institute (November 2003).

[24] D.Pizarro, J.Peyras, A.Bartoli, Light-invariant fitting of active appearance models, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[25] M.B.Stegmann, R.Larsen, Multi-band modelling of appearance, in: International Workshop on Generative Model-Based Vision, 2002.

[26] M.Zhou, Y.Wang, X.Feng, X. Wang, A robust texture preprocessing for aam, in: International Conference on Computer Science and Software Engineering, 2008.

[27] I.M.Scott, T.F.Cootes, C.J.Taylor, Improving appearance model matching using local image structure, in: Information Processing in Medical Imaging, 2003, pp. 258–269.

[28] R.Larsen, M. Stegmann, S.Darkner, S.Forchhammer, T.F.Cootes, B.K.Ersbøll, Texture enhanced appearance models, Comput Vis Image Underst 106 (1) (2007) 20–30.

[29] X.Liu, F.W.Wheeler, P.H.Tu, Improved face model fitting on video sequences, in: British Machine Vision Conference, 2007.

[30] P.Martins, R.Caseiro, J.Batista, Face alignment through 2.5d active appearance models, in: British Machine Vision Conference, 2010.

[31] O. Bottema, Topics in Elementary Geometry, Springer, 2008.

[32] S.Baker, I.Matthews, Lucas-kanade 20 years on: A unifying framework, Int J Comput Vis 56 (1) (2004) 221–255.

[33] B.Lucas, T.Kanade, An iterative image registration technique with an application to stereo vision (darpa), in: DARPA Image Understanding Workshop, 1981, pp. 121–130.

[34] P.Lucey, S.Lucey, M.Cox, S.Sridharan, J.F.Cohn, Comparing object alignment algorithms with ap-

pearance variation: Forward-additive vs inverse-composition, in: IEEE International Workshop on Multimedia Signal Processing - MMSP, 2008, pp. 337–342.

[35] C.W.Chen, C.C.Wang, 3d active appearance model for aligning faces in 2d images, in: IEEE/RSJ International Conference on Intelligent Robots and Systems - IROS, 2008.

[36] P.Viola, M.Jones, Robust real-time object detection, Int J Comput Vis 57 (2) (2002) 137–154.

[37] G.Hager, P.Belhumeur, Efficient region tracking with parametric models of geometry and illumination, IEEE Trans Pattern Anal Mach Intell 20 (10) (1998) 1025–39.

[38] R.Gross, I.Matthews, S.Baker, Active appearance models with occlusion, Image Vis Comput 24 (6) (2006) 593–604.

[39] M.G.Roberts, T.F.Cootes, J.E.Adams, Robust active appearance models with iteratively rescaled kernels, in: British Machine Vision Conference, Vol. 1, 2007, pp. 302–311.

[40] B. Theobald, I.Matthews, S.Baker, Evaluating error functions for robust active appearance models, in: IEEE International Conference on Automatic Face and Gesture Recognition, 2006, pp. 149–154.

[41] D. DeMenthon, L. Davis, Model-based object pose in 25 lines of code, Int J Comput Vis 15 (1995) 123–141.

[42] R.Gross, I.Matthews, S.Baker, Generic vs. person specific active appearance models, Image Vis Comput 23 (1) (2005) 1080–1093.

[43] V. Rabaud, Vincent's Structure from Motion Toolbox, http://vision.ucsd.edu/~vrabaud/toolbox/.

[44] L.Torresani, A.Hertzmann, C.Bregler, Learning non-rigid 3d shape from 2d motion, in: Neural Information Processing Systems, 2003.

[45] I.Akhter, Y.Sheikh, S.Khan, T.Kanade, Nonrigid structure from motion in trajectory space, in: Neural Information Processing Systems, 2008.

[46] M.Nordstrom, M.Larsen, J.Sierakowski, M.Stegmann, The IMM face database - an annotated dataset of 240 face images, Tech. rep., Technical University of Denmark, DTU (May 2004).
URL http://www2.imm.dtu.dk/pubdb/p.php?3160

[47] L.Yin, X.Chen, Y.Sun, T.Worm, M.Reale, A high-resolution 3d dynamic facial expression database, in: IEEE International Conference on Automatic Face and Gesture Recognition, 2008, pp. 17–19.

[48] T. Dalgleish, M. Power, Handbook of Cognition and Emotion, John Wiley & Sons Ltd, 1999.

[49] FGNet, Talking face video (2004).
URL www-prima.inrialpes.fr/FGnet/data/01-TalkingFace/talking_face.html