# Efficient, Robust and Accurate Fitting of a 3D Morphable Model

Sami Romdhani Thomas Vetter
University of Basel, Computer Science Department
Bernoullistrasse 16, CH - 4056 Basel, Switzerland
{sami.romdhani, thomas.vetter}@unibas.ch

## Abstract

*3D Morphable Models, as a means to generate images of a class of objects and to analyze them, have become increasingly popular. The problematic part of this framework is the registration of the model to an image, a.k.a. the fitting. The characteristic features of a fitting algorithm are its efficiency, robustness, accuracy and automation. Many accurate algorithms based on gradient descent techniques exist which are unfortunately short on the other features. Recently, an efficient algorithm called Inverse Compositional Image Alignment (ICIA) algorithm, able to fit 2D images, was introduced. In this paper, we extent this algorithm to fit 3D Morphable Models using a novel mathematical notation which facilitates the formulation of the fitting problem. This formulation enables us to avoid a simplification so far used in the ICIA, being as efficient and leading to improved fitting precision. Additionally, the algorithm is robust without sacrificing its efficiency and accuracy, thereby conforming to three of the four characteristics of a good fitting algorithm.*

## 1. Introduction

3D Morphable Models, as a means to generate images of a class of objects and to analyze them, have become increasingly popular (e.g [19, 6, 16]). For instance, Blanz *et al.*, use a 3D Morphable Model to analyze human face images across wide variations of pose and illumination [5, 20]. It was shown in [5] that, provided the fitting of the model to the face image is accurate, the identification performances are very high (more than 97% across large variations of pose and illumination). A fitting algorithm, i.e. an algorithm aiming at registering the model with an input image, is characterized by the following four features:

- **Efficient**: The efficiency of the fitting is clearly dependent on applications. Security applications, for instance, requires fast algorithm (i.e. near real time).
- **Robust** against non-Gaussian noise. The assumption of normality of the difference between the image synthesized by the model and the input image is generally violated due to the presence of accessories or artifacts (glasses, hair, specular highlight).
- **Accurate**, as we have already pointed out, the accuracy is crucial for the application which is to use the fitting results (and, generally, the level of accuracy required depends thereon.)
- **Automatic**: The fitting should require as little human intervention as possible, optimally, no initialization.

An algorithm capable of any of the four aforementioned features is difficult to set up. An algorithm capable of *all* four features is the holy grail of model based computer vision. Accurate algorithms exist already since a few years (e.g. [5]) but they are short on the three other features. Recently, Baker and Matthews [2] presented an efficient algorithm for fitting 2D correspondence-based models, called Inverse Compositional Image Alignment (ICIA). They derived a first order approximation thereof to fit Flexible Appearance Models (i.e. sparse correspondence based 2D models). In this paper, we extent this algorithm to 3D Morphable Models. This 3D extension requires the introduction of a new notation (see Section 2) which separates the image frame from an object centered reference frame. The algorithm presented in Section 3 has several advantages over the original ICIA: Due to the new formalism for describing correspondence based models, the algorithm is not a first order approximation of the ICIA as was presented in [2]. This leads to a different inverse shape composition which is more accurate. We also present a method for making the algorithm robust without sacrificing its efficiency and its accuracy (Section 4). So, our algorithm is capable of the first three features, leaving its automation for future improvements.

## 2. 3D Morphable Models

Morphable Models are not new in computer vision [8, 9] and originated from the understanding that any object is characterized by two entities: its shape and its color. This is the main difference between Morphable Models and older approaches such as Eigenfaces [23] which only model the color. In Morphable Models, the shape and the color are modeled separately. 2D as well as 3D Morphable Models parameterize in a consistent manner the 2D surface of the whole set of modeled objects (e.g. human faces). The object centered reference frame, on which both the shape and the texture of the 2D surface are parameterized, is denoted by $(u, v)$. As the objects are parameterized on the same reference frame, the set of objects is closed under addition: the addition of two valid objects is a valid object. Hence we can form linear combinations of objects and the class of objects is said to form a *Linear Objects Class* [24]. Then the 3D shape of an object is defined by a function $s(u, v)$ mapping the $(u, v)$ coordinate system into the three-dimensional $(x, y, z)$ coordinate system. Similarly the color, also called texture, is defined by the function $t(u, v)$ mapping the $(u, v)$ space into the RGB color space. The distinction, which is lacking in [2], between the reference frame $(u, v)$ and the $(x, y, z)$ coordinate system helps clarifying the explanation of the fitting algorithm. In the remaining of this section the shape and texture models are detailed.

### 2.1. Generative Shape Model

**3D Shape modeling** The representation of the 3D shape is discrete: it is a mapping from $(u_i, v_i)$ to $(x_i, y_i, z_i)$ for $i = 1, \ldots, N_v$, where $N_v$ is the number of vertices of the model. We represent this shape by the matrix $\mathbf{S}_{3\times N_v}$ for which each column corresponds to one of the reference points $(u_i, v_i)$. The shapes of the Linear Object Class are modeled by a linear combination of modes of variation $\mathbf{S}^k$ obtained by applying PCA on a set of example shapes:

$$\mathbf{S}_{3\times N_v} = \mathbf{S}^0 + \sum_{k=1}^{N_s} \alpha_k \cdot \mathbf{S}^k, \tag{1}$$

where $N_s$, the number of shape dimensions, is typically in the region of one hundred. A continuous shape is obtained by interpolation across neighboring vertices defined by a triangle list as is common in Computer Graphics.

**Shape projection to the image frame** The projection of the vertices of a 3D shape to the 2D image frame $(x, y)$, using a weak perspective transformation with focal length $f$, rotation matrix $\mathbf{R}$, and translation $\mathbf{t}$, is denoted by $\mathbf{P}$:

$$\mathbf{P}_{2\times N_v} = f \cdot \mathbf{R}_{2\times 3} \cdot \left( \mathbf{S}^0 + \sum_{k=1}^{N_s} \alpha_k \cdot \mathbf{S}^k \right) + \mathbf{t}_{2\times 1} \cdot \mathbf{1}_{1\times N_v} \tag{2}$$

A weak perspective is required, as opposed to a full perspective projection, to facilitate the second step of the shape composition (i.e. the recovery of $\boldsymbol{\alpha}$ and the projection parameters from $\mathbf{P}$) explained in Section 3.1. For ease of explanation, the weak perspective transformation parameters are denoted by the vector $\boldsymbol{\rho} = [f \, \mathrm{vec}(\mathbf{R})^{\mathrm{T}} \mathbf{t}^{\mathrm{T}}]^{\mathrm{T}}$, where $\mathrm{vec}\mathbf{R}$ vectorizes $\mathbf{R}$ by stacking its columns, and $\boldsymbol{\alpha}$ is the vector whose elements are the $\boldsymbol{\alpha}_k$. In the remaining of the paper, the projection of the vertex $i$ to the image frame $(x, y)$ is denoted by the vector valued function $\mathbf{p}(u_i, v_i; \boldsymbol{\alpha}, \boldsymbol{\rho}) = \mathbf{P}_{\cdot,i}$. This function is clearly continuous in $\boldsymbol{\alpha}$, and $\boldsymbol{\rho}$. To provide continuity in the $(u, v)$ space as well, we use a triangle list and interpolate between neighboring vertices. Note that only $N_{vv}$ vertices, a subset of the $N_v$ vertices, are visible after the 2D projection (the remaining vertices are hidden by self-occlusion). We call this subset the domain of the shape projection $\mathbf{p}(u_i, v_i; \boldsymbol{\alpha}, \boldsymbol{\rho})$ and denote it by $\Omega(\boldsymbol{\alpha}, \boldsymbol{\rho})$.

In conclusion, the shape modeling and its projection provides a mapping from the parameter space $\boldsymbol{\alpha}, \boldsymbol{\rho}$ to the image frame $(x, y)$ via the reference frame $(u, v)$.

### 2.2. Shape Inverse

So far, we detailed the generative shape projection, i.e. the mapping from the parameter space $\boldsymbol{\alpha}, \boldsymbol{\rho}$ to the image frame $(x, y)$. In a fitting algorithm, we are interested in solving the inverse problem, i.e. recovering the parameters from an image. We will see in the next section, that the fitting algorithm requires two tools: the inverse shape projection and an algorithm which separates the model parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\rho}$ from a set a vertices projection, $\mathbf{P}$ (i.e. the inversion of Equation (2)).

**Inverse shape projection and Projections Composition** The inverse shape projection maps an image point $(x, y)$ to the reference frame $(u, v)$. Let us denote the composition of a shape projection and its inverse by the symbol $\circ$, hence, $\mathbf{p}(u, v; \boldsymbol{\alpha}, \boldsymbol{\rho}) \circ \mathbf{p}^{-1}(x, y; \boldsymbol{\alpha}, \boldsymbol{\rho})$ is equal to $\mathbf{p}(\mathbf{p}^{-1}(x, y; \boldsymbol{\alpha}, \boldsymbol{\rho}); \boldsymbol{\alpha}, \boldsymbol{\rho})$, but we prefer the former notation for clarity. The inverse shape projection is defined according to the following axiom, which specifies that under the same set of parameters the shape projection composed with its inverse is equal to the identity:

**Axiom 1**

$$\mathbf{p}(u, v; \boldsymbol{\alpha}, \boldsymbol{\rho}) \circ \mathbf{p}^{-1}(x, y; \boldsymbol{\alpha}, \boldsymbol{\rho}) = (x, y),$$
$$\mathbf{p}^{-1}(x, y; \boldsymbol{\alpha}, \boldsymbol{\rho}) \circ \mathbf{p}(u, v; \boldsymbol{\alpha}, \boldsymbol{\rho}) = (u, v),$$

Due to the discretization of the shape, it is not easy to express analytically $\mathbf{p}^{-1}$ as a function of $\mathbf{p}$, but it can be computed using the triangle list: The domain of the plane $(x, y)$ for which there exists an inverse under the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\rho}$, denoted by $\Psi(\boldsymbol{\alpha}, \boldsymbol{\rho})$, is the range of $\mathbf{p}(u, v; \boldsymbol{\alpha}, \boldsymbol{\rho})$. Such a point of $(x,y)$ lies in a single triangle under the projection $\mathbf{p}(u,v;\boldsymbol{\alpha},\boldsymbol{\rho})$. So, the point in $(u,v)$ under the inverse projection has the same relative position in this triangle in the $(u,v)$ space. This process is depicted in Figure 1.

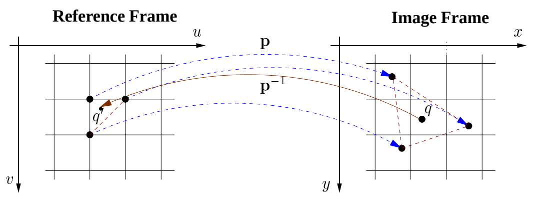

**Figure 1. The inverse shape function $\mathbf{p}^{-1}(x,y;\boldsymbol{\alpha},\boldsymbol{\rho})$ maps the point $q$ (defined in the $(x,y)$ coordinate system), onto the point $q'$ in $(u,v)$. This is done by recovering the triangle which would contain the pixel $q$ under the mapping $\mathbf{p}(u,v;\boldsymbol{\alpha},\boldsymbol{\rho})$. Then the relative position of $q$ in that triangle is the same as the relative position of $q'$ in the same triangle in the $(u,v)$ space.**

Baker *et al.*, in [2], do not make the distinction between the reference and the image frames. A consequence of this, is that they require the set of warps to be closed under inversion (which is not required for our formulation). This leads them to a first order approximation of the inverse shape projection (called inverse warping in their nomenclature): $\mathbf{p}^{-1}(x,y;\boldsymbol{\alpha}) = \mathbf{p}(u,v;-\boldsymbol{\alpha})$ (Note that there are no projection parameters $\boldsymbol{\rho}$, as they use a 2D model). This does not agree with our Axiom (1): As shown in Figure 2, a point from $(u,v)$, $q'$, is mapped under $\mathbf{p}(u,v;\boldsymbol{\alpha})$ to $q$ in $(x,y)$. Hence, to agree with the axiom, this point $q$ must be warped back to $q'$ under $\mathbf{p}^{-1}(x,y;\boldsymbol{\alpha})$. So, the displacement in $q$ which should be inverted is the one from $q'$. However, in Baker *et al.* [2], the displacement function $\mathbf{p}$ is inverted at the point $q$, leading to the point $b$, instead of $q'$. This is due to the fact that the distinction between the two coordinates systems is not made. This approximation is less problematic for a sparse-correspondence model as used by Baker for which the triangles are quite large (see Image (b) of Figure 2 of [2]), because the chances that both $q$ and $q'$ fall in the same triangle are much higher than in our dense correspondence model for which the triangles are much tinier. When $q$ and $q'$ fall in the same triangle, then their displacements are similar to a first order approximation, due to the linear interpolation inside triangles, and the error made during composition is small. This explains the good result obtained by Baker.

**Recovery of $\boldsymbol{\alpha}$ and $\boldsymbol{\rho}$ from $\mathbf{P}$** To recover the model parameters, $\boldsymbol{\alpha}, f, \mathbf{R}$ and $\mathbf{t}$ from correspondences between a set of model vertices and their projection in the image frame,

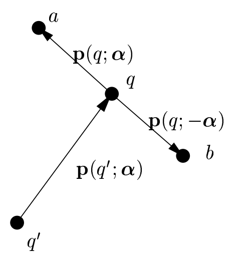

**Figure 2. First order approximation of the inverse shape projection defined by Baker and Matthews in [2] which violates Axiom 1.**

Equation (2) must be inverted. Because it is a weak perspective projection, this equation is a bilinear combination between the shape parameters $\boldsymbol{\alpha}$ and the 3D rotation matrix $\mathbf{R}$. The recovery is performed by the closed form solution detailed in [21] which presents a novel *selective* approach addressing this problem more accurately than the former method based on SVD factorization [3]. The rotation matrix and the focal length are recovered using Equation (15) of [21], the translation using Equation (16) of [21], and then the shape parameters $\boldsymbol{\alpha}$ are recovered by inverting the linear system of equations using the estimated rotation matrix and focal length.

## 2.3. Texture mapping and its inverse

Similarly to the shape model, the texture (or color) model is also defined on the same $(u,v)$ reference frame. The RGB texture is discretized in the same manner, and again assuming that the set of textures back-warped on the reference shape form a Linear Object Class and after applying PCA to a set of example textures:

$$\mathbf{T}_{3\times N_v} = \mathbf{T}^0 + \sum_{k=1}^{N_t} \beta_k \cdot \mathbf{T}^k, \tag{3}$$

where $N_t$ is the number of texture dimensions (again in the region of one hundred) and each rows of the matrix $\mathbf{T}$ holds the values of one color channel. Also, similarly to the shape we denote the color of a vertex $i$ by the vector valued function $\mathbf{t}(u_i,v_i;\boldsymbol{\beta})$, which is extended to the continuous function $\mathbf{t}(u,v;\boldsymbol{\beta})$ by using the triangle list and interpolating. In the fitting algorithm, a texture inverse, $\mathbf{t}^{-1}(u,v;\boldsymbol{\beta})$, is required:

$$\mathbf{t}^{-1}(\mathbf{t}(u_i,v_i);\boldsymbol{\beta}) = \mathbf{t}(u_i,v_i) - \sum_{k=1}^{N_t} \beta_k \cdot \mathbf{T}^k_{\cdot,i}, \tag{4}$$

where $\mathbf{T}^k_{\cdot,i}$ denotes the $i^{\text{th}}$ column of the matrix $\mathbf{T}^k$. This definition is chosen for the texture inverse because then a texture composed with its inverse, under the same set of parameters is equal to the mean texture: $\mathbf{t}^{-1}(\mathbf{t}(u_i, v_i; \boldsymbol{\beta}); \boldsymbol{\beta}) = \mathbf{T}^0_{\cdot,i}$.

Synthesizing the image of an object is performed by warping a texture from the reference to the image frames using a shape projection:

$$I(x_j, y_j; \boldsymbol{\alpha}, \boldsymbol{\rho}, \boldsymbol{\beta}) = \mathbf{t}(u, v; \boldsymbol{\beta}) \circ \mathbf{p}^{-1}(x_j, y_j; \boldsymbol{\alpha}, \boldsymbol{\rho}) \quad (5)$$

where $j$ runs over the pixels for which a shape inverse exist as defined in Section 2.1.

## 3. Image alignment algorithm

In this section, we extent the iterative Inverse Compositional Image Alignment (ICIA) algorithm of Baker and Matthews [2] to fit 3D Morphable Models, using the formalism stated in Section 2. The key feature of this algorithm is its efficiency due to the pre-computation of the derivatives. As our definition of the inverse shape projection is different from [2], its composition with the current shape estimate is also different. For brevity, we denote by $\boldsymbol{\gamma}$ the stacking of the shape and projection parameters, $\boldsymbol{\gamma} = [\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{\rho}^{\mathrm{T}}]^{\mathrm{T}}$. The cost function iteratively minimized in the ICIA algorithm is a sum of squares of the difference between two textures. It is, hence, defined in the reference frame $(u, v)$. The first texture is the model texture. It is the result of a shape projection update (with parameters $\Delta\boldsymbol{\gamma}$) composed with a texture update (with parameters $\Delta\boldsymbol{\beta}$). The second texture of the difference is extracted from the input image to be fitted $I(x, y)$ using the shape projection with the current shape parameters $\boldsymbol{\gamma^c}$. This texture then undergoes an inverse texture mapping using the current texture parameters $\boldsymbol{\beta^c}$. The cost function to be minimized is the following:

$$C(\Delta\boldsymbol{\gamma}, \Delta\boldsymbol{\beta}, \boldsymbol{\gamma^d}, \boldsymbol{\gamma^c}, \boldsymbol{\beta^c}, I) = \frac{1}{2} \cdot \sum_{u_i, v_i \in \Omega(\boldsymbol{\gamma^d})} \big(\mathbf{t}(u, v; \Delta\boldsymbol{\beta}) \circ \mathbf{p}^{-1}(x, y; \boldsymbol{\gamma^d}) \circ \mathbf{p}(u_i, v_i; \boldsymbol{\gamma^d} + \Delta\boldsymbol{\gamma}) - \mathbf{t}^{-1}(I(x, y) \circ \mathbf{p}(u_i, v_i; \boldsymbol{\gamma^c}); \boldsymbol{\beta^c})\big)^2 \quad (6)$$

where the parameters superscripted by $^d$ refer to the parameters at which the derivatives are computed and the parameters superscripted by $^c$ refer to the current parameters. The cost function $C$ is to be minimized with respect to the model parameters update $\Delta\boldsymbol{\gamma}$ and $\Delta\boldsymbol{\beta}$. There are two novelties with respect to Baker's formulation: The first is the presence of the inverse shape projection $\mathbf{p}^{-1}(x, y; \boldsymbol{\gamma^d})$ between the shape update and the texture update. This inverse shape projection must be present because it is not possible to compose a shape (projecting to the image frame) with a texture (whose domain is the reference frame). The second novelty is the update of the texture parameters as well, $\Delta\boldsymbol{\beta}$.

For comparison, the cost function of the common additive approach, used for instance by Blanz *et al.* [6] follows:

$$C(\Delta\boldsymbol{\gamma}, \Delta\boldsymbol{\beta}, \boldsymbol{\gamma^c}, \boldsymbol{\beta^c}, I) = \frac{1}{2} \cdot \sum_{x_j, y_j \in \Psi(\boldsymbol{\gamma^c})} \big(I(x_j, y_j) - \mathbf{t}(u, v; \boldsymbol{\beta^c} + \Delta\boldsymbol{\beta}) \circ \mathbf{p}^{-1}(x_j, y_j; \boldsymbol{\gamma^c} + \Delta\boldsymbol{\gamma})\big)^2 \quad (7)$$

Let us now compute the derivative of the ICIA cost function (Equation (6)) with respect to the shape and projection parameter update $\Delta\boldsymbol{\gamma}$ at the point $(0, 0, \boldsymbol{\gamma^d}, \boldsymbol{\gamma^c}, \boldsymbol{\beta^c}, I)$. In the remaining, we will omit the dependent variables, assuming that it is clear that the dependents of $\mathbf{p}$ and $\mathbf{t}$ are $(u, v)$ and the dependents of $\mathbf{p}^{-1}$ and $I$ are $(x, y)$.

$$\frac{\partial C}{\partial \Delta\gamma_k} = \sum_i \left.\frac{\partial (t(0) \circ \mathbf{p}^{-1}(\boldsymbol{\gamma^d}) \circ \mathbf{p}_i(\boldsymbol{\gamma^d} + \Delta\boldsymbol{\gamma}))}{\partial \Delta\gamma_k}\right|^{\mathrm{T}}_{\Delta\boldsymbol{\gamma}=0} \cdot \left[\mathbf{t}(0) \circ \mathbf{p}^{-1}(\boldsymbol{\gamma^d}) \circ \mathbf{p}_i(\boldsymbol{\gamma^d}) - \mathbf{t}^{-1}(I \circ \mathbf{p}_i(\boldsymbol{\gamma^c}); \boldsymbol{\beta^c})\right] \quad (8)$$

Note that, using Equation (5) and the chain rule:

$$\mathbf{t}(0) \circ \mathbf{p}^{-1}(\boldsymbol{\gamma^d}) = I^d(x, y; \boldsymbol{\alpha^d}, \boldsymbol{\rho^d}, 0) \quad (9)$$

$$\frac{\partial (t(0) \circ \mathbf{p}^{-1}(\boldsymbol{\gamma^d}) \circ \mathbf{p}_i(\boldsymbol{\gamma}))}{\partial \gamma_k} = \nabla I^d \cdot \frac{\partial \mathbf{p}_i(\boldsymbol{\gamma})}{\partial \gamma_k} \quad (10)$$

We refer to the second factor of the right member of Equation (8) in squared brackets as the *texture error* at the vertex $i$, $\mathbf{e}_i$. The texture error, $\mathbf{e}$, is a column vector of length $3N_{vv}$; it is a difference of two terms: The first one is the mean texture (the projection and inverse projection cancel each other using Axiom 1). The second term is the image to be fitted mapped to the reference frame $(u, v)$ using the current shape and projection parameters and inverse-texture mapped with the current texture parameters. At the optimum (and if the face image can be fully explained by the model), this term is equal to the mean texture, and hence the texture error is null. The first factor of (8) is the element of the shape Jacobian, $\mathbf{J^s}$, at the row $i$ and column $k$. The dimensions of the shape Jacobian matrix are $3N_{vv} \times N_s$. The key feature of the ICIA algorithm is that the Jacobian depend only on $\boldsymbol{\gamma^d}$ which is constant across iterations, as opposed to the Jacobian of the additive formulation (Equation (7)) that depends on the current parameter estimate $\boldsymbol{\gamma^c}$ and $\boldsymbol{\beta^c}$.

The derivatives with respect to the texture parameters update $\Delta\boldsymbol{\beta}$ take a similar form to those of the shape parameters. They are simpler, however, as the texture Jacobian is simply equal to the linear texture model: $\mathbf{J^t}_{\cdot,k} = \mathrm{vec}(\mathbf{T}^k)$. The combined Jacobian of the shape and texture model is then: $\mathbf{J} = [\mathbf{J^s}\mathbf{J^t}]$. The Gauss approximation of the Hessian is $\mathbf{H} = \mathbf{J}^{\mathrm{T}}\mathbf{J}$, leading to the Gauss-Newton update:

$$\begin{pmatrix} \Delta\boldsymbol{\gamma} \\ \Delta\boldsymbol{\beta} \end{pmatrix} = -\mathbf{H}^{-1} \cdot \mathbf{J}^{\mathrm{T}} \cdot \mathbf{e} \quad (11)$$

### 3.1. Shape projection composition

In the ICIA algorithm the shape projection update is *composed* with the current shape projection estimate. This composition, detailed in this section, is performed in two steps. The first step computes the correspondences after composition between the model vertices and the image pixels, i.e. it yields a mapping from $(u_i, v_i)$ to $(x, y)$, denoted by $\mathbf{p}^*(u_i, v_i)$. The second step maps this set of vertices-pixels correspondence to the shape model, yielding the model parameters after composition, $\boldsymbol{\alpha}^*$ and $\boldsymbol{\rho}^*$.

**Correspondences after composition** The update obtained after an iteration is a transformation of the reference frame $(u, v)$: $\mathbf{p}^{-1}(x, y; \boldsymbol{\gamma}^{\boldsymbol{d}}) \circ \mathbf{p}(u_i, v_i; \boldsymbol{\gamma}^{\boldsymbol{d}} + \Delta\boldsymbol{\gamma})$ (see Equation (6)). It is this transformation that must be composed with the current shape projection to obtain the new correspondences: The result of the shape projection composition is a shape projection, $\mathbf{p}^*(u_i, v_i)$, mapping the points of the reference frame $(u_i, v_i)$ to the image frame, $(x_i^*, y_i^*)$, equal to:

$$\mathbf{p}^*(u_i, v_i) = \mathbf{p}(u, v; \boldsymbol{\gamma}^{\boldsymbol{c}}) \circ \mathbf{p}^{-1}(x, y; \boldsymbol{\gamma}^{\boldsymbol{d}}) \circ \mathbf{p}(u_i, v_i; \boldsymbol{\gamma}^{\boldsymbol{d}} + \Delta\boldsymbol{\gamma}) \tag{12}$$

Under the first shape projection, the rightmost on the above equation, the vertex $i$ is mapped to the image frame, say the point $(x_i^+, y_i^+)$, under the parameters $\boldsymbol{\gamma}^{\boldsymbol{d}} + \Delta\boldsymbol{\gamma}$ using Equation (2). Then, under the inverse shape projection, this point $(x_i^+, y_i^+)$ is mapped to the reference frame, say the point $(u_i^+, v_i^+)$, using the procedure described in the third paragraph of Section 2.1. Finally, $(u_i^+, v_i^+)$ is mapped to $(x_i^*, y_i^*)$ using the shape projection with the parameters $\boldsymbol{\gamma}^{\boldsymbol{c}}$.

**Correspondences mapping to the shape model** The first step of the composition yields a set of correspondences between the model vertices $(u_i, v_i)$ and the points in the image frame $(x_i^*, y_i^*)$ (represented by the matrix $\mathbf{P}$ in Equation (2)). To recover the model parameters, $\boldsymbol{\alpha}, f, \mathbf{R}$ and $\mathbf{t}$ explaining these correspondences, the algorithm presented in the second paragraph of Section 2.2 is used.

### 3.2. Inverse Compositional Image Alignment Alg.

The main advantage of the ICIA algorithm over the former fitting algorithms is its efficiency: The Jacobian used in the computation of the update does not depend on the current estimate $\boldsymbol{\gamma}^{\boldsymbol{c}}$. Hence the Jacobian, $\mathbf{J}$, and the inverse of the Hessian, $\mathbf{H}^{-1}$, can be pre-computed. What is left to do during the iterative fitting is a matrix-vector product (Equation (11)) and a shape projection composition.

Due to the inverse shape projection introduced in this paper, our algorithm is more general than the original ICIA detailed in [2], for two reasons: (**i**) The extension of the ICIA to Flexible Correspondence based Models in [2] is a first order approximation (i.e. a simplification that yields generally reasonable results when the model is based on sparse correspondences). This is because the inverse shape projection proposed in [2] does not agree with the Axiom 1. The exposition of the ICIA algorithm applied to Morphable Models presented here is not based on a first order approximation. Hence, the shape projection composition of large displacement is more accurate for the algorithm presented in this paper. (**ii**) ICIA requires that the set of shape projection deformations is closed under composition. This means that two valid shape projections composed with one another must form a valid shape projection. A *valid* shape projection is a set of correspondences between vertices and image frame points for which there exist a set of $\boldsymbol{\alpha}$ and $\boldsymbol{\rho}$ parameters that can exactly satisfy Equation (2). However, the whole fitting process can *and should* be performed without this requirement. Indeed, the input to each iteration is a set of correspondences between the vertices $(u_i, v_i)$ and the image frame. However, it is not required that this set of correspondences is within the span of the shape model. Therefore the whole fitting algorithm can be performed by applying only the first stage of the composition and not the second. We call this scheme *Deferred Correspondences Mapping*. Under DCM, there are no current shape parameters $\boldsymbol{\gamma}^{\boldsymbol{c}}$, but a current set of correspondences, $\mathbf{p}^{\mathbf{c}}(u, v)$, hence Equation (12) is rewritten as:

$$\mathbf{p}^*(u_i, v_i) = \mathbf{p}^{\mathbf{c}}(u, v) \circ \mathbf{p}^{-1}(x, y; \boldsymbol{\gamma}^{\boldsymbol{d}}) \circ \mathbf{p}(u_i, v_i; \boldsymbol{\gamma}^{\boldsymbol{d}} + \Delta\boldsymbol{\gamma}) \tag{13}$$

DCM has two advantages: Firstly, it is more accurate. Indeed, due to our inverse shape projection, Equation (13) can be exactly satisfied. However, if $\mathbf{p}^*(u_i, v_i)$ was to be mapped to the shape model (thereby extracting $\boldsymbol{\gamma}^{\boldsymbol{c}}$), then the mapped correspondences could not satisfy this equation anymore: The mapping introduces an error in the composition because the set of correspondences after composition generally does not lie within the span of the shape model. When the result of the composition is not mapped to the shape model, then no error is introduced, i.e. the composition is exact. However it is shown in the experimental section that the error introduced by the projection to the shape model is very small. The second advantage is that it makes the algorithm more efficient as the most time consuming part of the composition is not applied during the iterative process. If, at the end of the fitting, the shape parameters are needed (for identification, for instance), then the correspondences are mapped and the shape parameters recovered. It must be noted that this introduces an error in the set of correspondences, which usually increases the value of the cost function $C$. This algorithm is, hence, a mixture between model constrained correspondence finding algorithms (a.k.a. fitting algorithms) and unconstrained correspondence finding algorithms (such as [4]). Indeed, the final set of correspondence is not in the span of the model, but each step of the iteration is constrained by the model. So, it combines advantages of both world: The correspondences obtained are indirectly constrained by the model, but if the model is not general enough, the correspondence estimates depart from the model in order to minimize the cost function. An advantage of this feature, is that the shape residual could be used to augment the object description.

The third modification of this version of the ICIA algorithm with respect to the one of Baker in [2] lies in the recovery of the texture parameters. The algorithm proposed in [2] is *invariant* to the texture parameters (this idea originates from [12]). This is performed by making the column of the shape Jacobian orthogonal to the ones of the texture Jacobian. We empirically verified, on images from the PIE face database which require the utilization of a robust cost function (see Section 4), that this introduces a non-negligible perturbation on the shape Jacobian which decreases the accuracy of the shape recovery. We therefore favor recovering the texture parameters as well, as explained in Section 3.

In an implementation of ICIA, the parameters at which the derivatives are computed $\boldsymbol{\gamma}^{\boldsymbol{d}} = [\boldsymbol{\alpha}^{\boldsymbol{d}^{\mathrm{T}}} \boldsymbol{\rho}^{\boldsymbol{d}^{\mathrm{T}}}]^{\mathrm{T}}$ must be selected. A natural choice for the shape parameters is $\boldsymbol{\alpha}^{\boldsymbol{d}} = 0$. The selection of $\boldsymbol{\rho}^{\boldsymbol{d}}$ is not as trivial, because the derivatives of the shape projections are computed in a particular image frame (see Equation (10)) set by $\mathbf{R}^{\mathbf{d}}$. Therefore, $\mathbf{R}^{\mathbf{d}}$ should be close to the optimal $\mathbf{R}$ (depending on the input image). Hence, a set of Jacobians is computed for a series of different $\mathbf{R}^{\mathbf{d}}$. During the iterative fitting, the derivatives used are the ones closest to the current estimation of $\mathbf{R}$. Note that, at first, this approach might seem very close to the View-based approach [18, 17, 7]. The difference is, however, fundamental. In this approach, the extraneous (rotation) parameters are clearly separated from the intrinsic (identity, i.e. $\boldsymbol{\alpha}, \boldsymbol{\beta}$) parameters. They are, however, convolved with one another in the View-based approach.

To summarize, the algorithm takes the following steps:

1. Start from the initial parameters $\boldsymbol{\alpha}^{\mathbf{c}} = 0$, $\boldsymbol{\beta}^{\mathbf{c}} = 0$ and $\boldsymbol{\rho}^{\mathbf{c}}$ sufficiently close to the optimum.
2. Compute the current correspondences: $\mathbf{p}^{\mathbf{c}}(u_i, v_i) = \mathbf{p}(u_i, v_i; \boldsymbol{\alpha}^{\mathbf{c}}, \boldsymbol{\rho}^{\mathbf{c}})$.
3. Compute the texture error $\mathbf{e}_i = \mathbf{t}(u_i, v_i; 0) - \mathbf{t}^{-1}(I \circ \mathbf{p}^{\mathbf{c}}(u_i, v_i); \boldsymbol{\beta}^{\mathbf{c}})$.
4. Compute the update $\Delta\boldsymbol{\alpha}, \Delta\boldsymbol{\beta}$ and $\Delta\boldsymbol{\rho}$ using Equation (11).
5. Set the new correspondences $\mathbf{p}^{\mathbf{c}}(u_i, v_i)$ to the composed shape projection obtained by Equation (13), and the new texture parameters $\boldsymbol{\beta}^{c}$ to $\boldsymbol{\beta}^{c} + \Delta\boldsymbol{\beta}$ (The composition of two textures is equivalent to the addition of their parameters).
6. Go back to step 3 until convergence, i.e. until the cost function $C$ does not decreases significantly.
7. If the optimal shape parameters are needed, then map $\mathbf{p}^{\mathbf{c}}(u_i, v_i)$ to the shape model.

## 4. Robustness and Efficiency

The ICIA algorithm is very efficient due to the fact that the Jacobian is constant and hence can be pre-computed. Then, the only iterative computation is a matrix-vector product and a shape composition (Equation (13)). The complexity of an iteration is of the order $O(N_{vv} \cdot (N_s + N_t))$. We have also shown that our version of the ICIA is accurate. As mentioned in the Introduction, robustness is also an important property of any fitting algorithm. The challenge is to attain a robust algorithm without sacrificing the efficiency and the accuracy. We propose to move toward this goal by two methods: (**i**) using a *Maximum A Posteriori* optimization rather than the Maximum Likelihood presented in Section 3, and (**ii**) alleviating the Gaussian noise assumption behind the least squares optimization of Section 3.

**Maximum A Posteriori** MAP optimization is performed when the prior of the model parameter is taken into account. As the shape and texture models are constructed by PCA, the model parameters have a Gaussian distribution with a diagonal covariance matrix $\Sigma$ and a null mean (We also assume such a distribution for the projection parameters for which their variance depends on their units). The shape prior is introduced in the cost function by augmenting it with the term $\lambda\boldsymbol{\gamma}^{*}\Sigma_{\gamma}^{-1}\boldsymbol{\gamma}^{*}$, where $\lambda$ is the variance of the noise. As the shape parameters after composition, $\boldsymbol{\gamma}^{*}$, cannot be easily derived from $\Delta\boldsymbol{\gamma}$ and from $\boldsymbol{\gamma}^{c}$, we approximate the prior term by $\lambda\Delta\boldsymbol{\gamma}\Sigma_{\gamma}^{-1}\Delta\boldsymbol{\gamma}$. Then, the MAP update is:

$$\begin{pmatrix}\Delta\boldsymbol{\gamma}\\ \Delta\boldsymbol{\beta}\end{pmatrix} = (\mathbf{H} + \lambda \cdot \Sigma^{-1})^{-1} \cdot \left(\mathbf{J}^{\mathrm{T}} \cdot \mathbf{e} + \lambda\Sigma_{\beta}^{-1} \cdot \boldsymbol{\beta}^{c}\right) \quad (14)$$

where $\Sigma$ is the covariance matrix of the shape and texture parameters. As $\lambda$ is rather difficult to estimate, it is tuned in a similar scheme to the Levenberg-Marquardt algorithm [11]. To keep the same efficiency, the new inverted Hessian is pre-computed for a series of characteristic $\lambda$.

**Robustness toward Outliers** The Morphable Model is not meant for modeling such things as glasses, specular highlights, facial hair, and the background surrounding the face. Their presence perturbs the fitting algorithm, as their residual dominates the least-squares cost function. *M-estimator* [15] is a standard technique for increasing the robustness of a fitting, by using a cost function which grows less that the square with respect to the residual. The cost function that we use is the *Talwar* [13] cost function which is a weighted least squares which gives a weight of 1 to inliers and 0 to outliers. A pixel is given the inlier or outlier status depending on its residual adjusted by the MAD, median of absolute deviations, and its leverage (i.e. down-weighting high leverage points) [10]. This optimization is efficiently implemented by *Iterative Reweighted Least-Squares* [14]. This method sacrifices little of the efficiency of the algorithm: The Jacobian is constant, hence the leverages can be also pre-computed. By use of the Talwar cost function, the Hessian is also constant as it does not depend on the image being fitted (and hence on the outlying pixels). So, Equation (11) can be iteratively solved by just discarding the rows of $\mathbf{J}$ corresponding to the outlying pixels. So, the complexity of this algorithm is $O((N_s + N_t) \cdot N_{vv} + N_{vv} \log N_{vv})$. The second term of this complexity is due to the computation of the MAD which requires the sorting of the residuals.

## 5. Experiments

We present, in Figure 3, a typical fitting result obtained on one of the CMU-PIE images [22]. The Morphable Model that we used was constructed on 100 example heads, the dimension of both the shape and texture model was chosen to 50. The model includes 18404 vertices. The example heads originate from individuals which are not present in the CMU-PIE database [22] used for testing. The selected photograph of the database is rather not 'easy' to fit due to the presence of glasses, which are not accounted for by the model. The average time used for fitting is 30s. on a C implementation on a P IV running a 2.8 GHz. The image (b) shows which pixels (in white and black) are sampled from to form the texture error $\mathbf{e}$. This image is taken at an early stage of the fitting, therefore the faces estimated with the current parameters overlaps some of the background. The white pixels are detected as outliers and are not taken into account in the fitting by the use of the Talwar cost function. Note that the glasses, the background pixels and the hair are properly detected as outliers. Therefore the fitting is very moderately perturbed by the presence of the glasses (see images (c) and (d)). It can be seen that the contour between the visible and invisible part of the face is not accurately matched (specially in the chin and ear areas). This is a common problem to the approaches based on a minimization of color differences (as [6]). Naturally the background color is not modeled (see image (d)) and hence the gradients of the input image and of the synthetic image are different on the contour, which explains the lack of fit. The contour of the synthetic image is the most probable one given the face interior. One way to improve the contour fitting, is to detect the visible contour in the image (either manually or automatically) and to add a cost function depending only on the shape to the current cost function. Further fitting as well as identification experiments are available at [1].

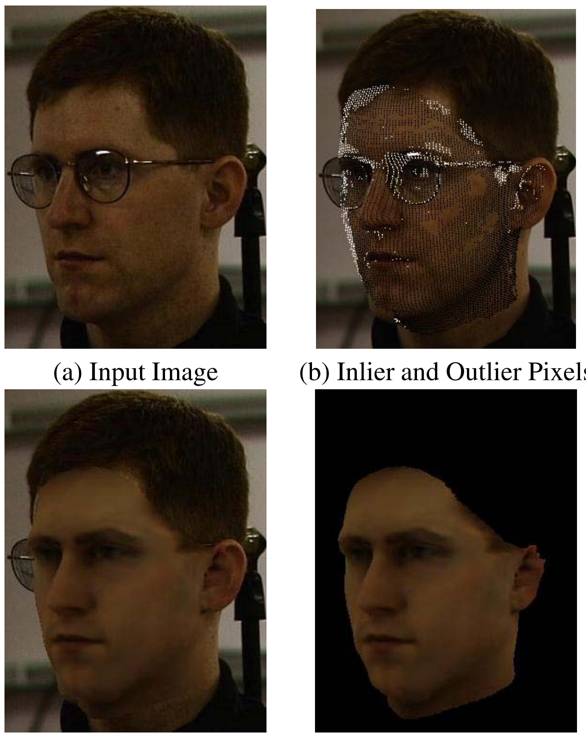

**Figure 3. Fitting result of an image of the CMU-PIE set. The image (b) shows, at an early stage of the fitting, in black and white, the pixels that the texture error $\mathbf{e}$ is sampled from and in white the outlier pixels thereof.**

**Deferred Correspondence Mapping** We conducted a set of experiments aiming at comparing the fitting with DCM (i.e. the correspondences *at each iteration* are not constrained to lie within the shape model) and without DCM. 68 images from the PIE face database [22] were fitted (front view at a neutral expression). Figure 4(a) shows the cost function per iteration averaged over the 68 fittings. The value of the cost function is averaged over the vertices used. The graphs shows that the results with or without DCM are essentially similar. Hence the fact that the shape model is not closed under composition has not a strong influence on the result. As a result, we favor DCM for its runtime performances.

**Influence of using a constant $\mathbf{R}^{\mathbf{d}}$** Ideally the azimuth and the elevation angles at which the derivatives are computed should be the same as the angles of the optimum. In this paragraph, we experiment on the error in correspondences obtained by using derivatives computed on a dif-

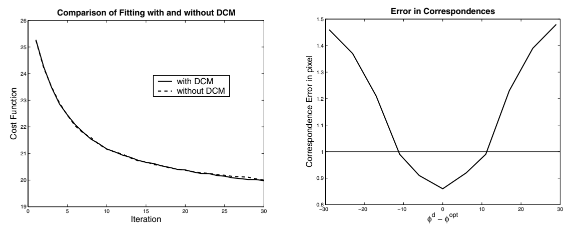

**Figure 4. (a) Left: Comparison of the cost function per iteration with and without DCM. (b) Right: Error in correspondence when the derivatives are computed in a different azimuth than the optimum.**

ferent azimuth angle than the optimum. We generated a synthetic face with one 3D Morphable Model and fitted it with another 3D Morphable Model. As the optimum face image is synthetic, we know exactly its pose and the 2D position of its vertices. Figure 4(b) shows a plot of the error of correspondences obtained after different fittings, each using derivatives computed at different azimuth angles. The figure shows that if a set of derivatives is pre-computed at 20° interval, the error in correspondences is less than a pixel.

## 6. Conclusions

After introducing a new mathematical formalism for describing Morphable Models, we used it to formulate the extension of the Baker's Inverse Compositional Image Alignment algorithm to 3D Morphable Models. We believe that this formalism is interesting in its own right, as it is rigorous and general to correspondence based models, and hence can be used to describe and compare different fitting strategies.

Our algorithm is more accurate than the original ICIA algorithm as it is not a first order approximation (no error is introduced in our inverse shape composition). We also presented a method which makes the algorithm robust without losing much of its efficiency and accuracy. A further advantage of our algorithm is that the correspondences are allowed to leave the span of the model in order to decrease the cost function.

One problem left to be addressed is the automation of the algorithm. This is difficult because making an algorithm both robust and fully automatic is contradictory in nature: An algorithm is robust because it down-weights outlier pixels detected by their high residual. However, far from the optimum, many pixels have a high-residual and yet they are inliers which should not be down-weighted. This is the challenge that we will address in the future.

**Acknowledgment** We thank Simon Baker for his enlightening views on the algorithm and his helpful comments. This material is based upon work supported by the European Research Office of the US Army under contract No. N68171-01-C-9000.

## References

[1] *http://informatik.unibas.ch/personen/romdhani_sami/icia*. 5

[2] S. Baker and I. Matthews. Equivalence and efficiency of image alignment algorithms. In *CVPR*, 2001. 1, 2, 2.2, 2, 3, 3.2, 3.2

[3] B. Bascle and A. Blake. Separability of pose and expression in facial tracking and animation. In *Sixth International Conference on Computer Vision*, 1998. 2.2

[4] J. Bergen and R. Hingorani. Hierarchical motion-based frame rate conversion. Technical report, David Sarnoff Research Center Princeton NJ 08540, 1990. 3.2

[5] V. Blanz, S. Romdhani, and T. Vetter. Face identification across different poses and illuminations with a 3d morphable model. In *Auto. Face and Gesture Recognition*, 2002. 1

[6] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D-faces. In *SIGGRAPH 99*, 1999. 1, 3, 5

[7] T. Cootes, K. Walker, and C. Taylor. View-based active appearance models. In *Automatic Face and Gesture Recognition*, 2000. 3.2

[8] I. Craw and P. Cameron. Parameterizing images for recognition and reconstruction. In *Proc. BMVC*, 1991. 2

[9] I. Craw and P. Cameron. Face recognition by computer. In *Proc. British Machine Vision Conference*, 1992. 2

[10] W. H. DuMouchel and F. L. O'Brien. Integrating a robust option into a multiple regression computing environment. In *21st Symposium on the Interface, American Statistical Association*, 1989. 4

[11] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, 1981. 4

[12] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *PAMI*, 1998. 3.2

[13] M. J. Hinich and P. P. Talwar. A simple method for robust regression. *J. Amer. Statist. Assoc.*, 1975. 4

[14] P. W. Holland and R. E. Welsch. Robust regression using iteratively reweighted least-squares. *Commun. Stat. (Theory & Methods)*, 1977. 4

[15] P. J. Huber. *Robust Statistics*. John Wiley & Sons, 1981. 4

[16] V. P. Kumar and T. Poggio. Learning-based approach to estimation of morphable model parameters. Technical report, MIT - A.I., 2000. 1

[17] H. Murase and S. Nayar. Visual learning and recognition of 3d objects from appearance. *IJCV*, 1995. 3.2

[18] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. In *CVPR*, 1994. 3.2

[19] T. D. Rikert and M. J. Jones. Gaze estimation using morphable models. In *Automatic Face and Gesture Recognition*, 1998. 1

[20] S. Romdhani, V. Blanz, and T. Vetter. Face identification by fitting a 3d morphable model using linear shape and texture error functions. In *Computer Vision – ECCV'02*, 2002. 1

[21] S. Romdhani, N. Canterakis, and T. Vetter. Selective vs. global recovery of rigid and non-rigid motion. Technical report, CS Dept, University of Basel, 2003. 2.2

[22] T. Sim, S. Baker, and M. Bsat. The cmu pose, illumination and expression (pie) database of human faces. Technical report, CMU, 2000. 5, 5

[23] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A*, 1987. 2

[24] T. Vetter and T. Poggio. Linear object classes and image synthesis from a single example image. *PAMI*, 1997. 2