3D Face Morphable Models "In-the-Wild"

James Booth Epameinondas Antonakos Stylianos Ploumpis George Trigeorgis
Yannis Panagakis Stefanos Zafeiriou
Imperial College London, UK
{james.booth,e.antonakos,s.ploumpis,g.trigeorgis,i.panagakis,s.zafeiriou}@imperial.ac.uk

Abstract

3D Morphable Models (3DMMs) are powerful statistical models of 3D facial shape and texture, and among the state-of-the-art methods for reconstructing facial shape from single images. With the advent of new 3D sensors, many 3D facial datasets have been collected containing both neutral as well as expressive faces. However, all datasets are captured under controlled conditions. Thus, even though powerful 3D facial shape models can be learnt from such data, it is difficult to build statistical texture models that are sufficient to reconstruct faces captured in unconstrained conditions ("in-the-wild"). In this paper, we propose the first, to the best of our knowledge, "in-the-wild" 3DMM by combining a powerful statistical model of facial shape, which describes both identity and expression, with an "in-the-wild" texture model. We show that the employment of such an "in-the-wild" texture model greatly simplifies the fitting procedure, because there is no need to optimize with regards to the illumination parameters. Furthermore, we propose a new fast algorithm for fitting the 3DMM in arbitrary images. Finally, we have captured the first 3D facial database with relatively unconstrained conditions and report quantitative evaluations with state-of-the-art performance. Complementary qualitative reconstruction results are demonstrated on standard "in-the-wild" facial databases. An open source implementation of our technique is released as part of the Menpo Project [1].

1. Introduction

During the past few years, we have witnessed significant improvements in various face analysis tasks such as face detection [20, 43] and 2D facial landmark localization on static images [41, 22, 7, 39, 44, 5, 6, 37]. This is primarily attributed to the fact that the community has made a considerable effort to collect and annotate facial images captured under unconstrained conditions [25, 46, 10, 33, 32] (commonly referred to as "in-the-wild") and to the discriminative methodologies that can capitalise on the availability of such large amount of data. Nevertheless, discriminative techniques cannot be applied for 3D facial shape estimation "in-the-wild", due to lack of ground-truth data.

Figure 1. Our "in-the-wild" Morphable Model is capable of recovering accurate 3D facial shape for a wide variety of images.

3D facial shape estimation from single images has attracted the attention of many researchers the past twenty years. The two main lines of research are *(i)* fitting a 3D Morphable Model (3DMM) [12, 13] and *(ii)* applying Shape from Shading (SfS) techniques [35, 36, 23]. The 3DMM fitting proposed in the work of Blanz and Vetter [12, 13] was among the first model-based 3D facial recovery approaches. The method requires the construction of a 3DMM which is a statistical model of facial texture and shape in a space where there are explicit correspondences. The first 3DMM was built using 200 faces captured in well-controlled conditions displaying only the neutral expression. That is the reason why the method was only shown to work on real-world, but not "in-the-wild", images. State-of-the-art SfS techniques capitalise on special multi-linear decompositions that find an approximate spherical harmonic decomposition of the illumination. Furthermore, in order to benefit from the large availability of "in-the-wild" images, these methods jointly reconstruct large collections of images. Nevertheless, even

thought the results of [35, 23] are quite interesting, given that there is no prior of the facial surface, the methods only recover 2.5D representations of the faces and particular smooth approximations of the facial normals.

3D facial shape recovery from a single image under "in-the-wild" conditions is still an open and challenging problem in computer vision mainly due to the fact that:

- The general problem of extracting the 3D facial shape from a single image is an ill-posed problem which is notoriously difficult to be solved without the use of any statistical priors for the shape and texture of faces. That is, without prior knowledge regarding the shape of the object at-hand there are inherent ambiguities present in the problem. The pixel intensity at a location in an image is the result of a complex combination of the underlying shape of the object, the surface albedo and normal characteristics, camera parameters and the arrangement of scene lighting and other objects in the scene. Hence, there are potentially infinite solutions to the problem.

- Learning statistical priors of the 3D facial shape and texture for "in-the-wild" images is currently very difficult by using modern acquisition devices. That is, even though there is a considerable improvement in 3D acquisition devices, they still cannot operate in arbitrary conditions. Hence, all the current 3D facial databases have been captured in controlled conditions.

With the available 3D facial data, it is feasible to learn a powerful statistical model of the facial shape that generalises well for both identity and expression [15, 31, 14]. However, it is not possible to construct a statistical model of the facial texture that generalises well for "in-the-wild" images and is, at the same time, in correspondence with the statistical shape model. That is the reason why current state-of-the-art 3D face reconstruction methodologies rely solely on fitting a statistical 3D facial shape prior on a sparse set of landmarks [3, 17].

In this paper, we make a number of contributions that enable the use of 3DMMs for "in-the-wild" face reconstruction (Fig. 1). In particular, our contributions are:

- We propose a methodology for learning a statistical texture model from "in-the-wild" facial images, which is in full correspondence with a statistical shape prior that exhibits both identity and expression variations. Motivated by the success of feature-based (e.g., HOG [16], SIFT [26]) Active Appearance Models (AAMs) [4, 5] we further show how to learn feature-based texture models for 3DMMs. We show that the advantage of using the "in-the-wild" feature-based texture model is that the fitting strategy gets simplified since there is not need to optimize with respect to the illumination parameters.

- By capitalising on the recent advancements in fitting statistical deformable models [30, 38, 5, 2], we propose a novel and fast algorithm for fitting "in-the-wild" 3DMMs. Furthermore, we make the implementation of our algorithm publicly available, which we believe can be of great benefit to the community, given the lack of robust open-source implementations for fitting 3DMMs.

- Due to lack of ground-truth data, the majority of the 3D face reconstruction papers report only qualitative results. In this paper, in order to provide quantitative evaluations, we collected a new dataset of 3D facial surfaces, using Kinect Fusion [19, 29], which has many "in-the-wild" characteristics, even though it is captured indoors.

- We release an open source implementation of our technique as part of the Menpo Project. [1]

The remainder of the paper is structured as follows. In Section 2 we elaborate on the construction of our "in-the-wild" 3DMM, whilst in Section 3 we outline the proposed optimization for fitting "in-the-wild" images with our model. Section 4 describes our new dataset, the first of its kind, to provide images with a ground-truth 3D facial shape that exhibit many "in-the-wild" characteristics. We outline a series of quantitative and qualitative experiments in Section 5, and end with conclusions in Section 6.

2. Model Training

A 3DMM consists of three parametric models: the *shape*, *camera* and *texture* models.

2.1. Shape Model

Let us denote the 3D mesh (shape) of an object with N vertexes as a $3N \times 1$ vector

$$\mathbf{s} = \left[\mathbf{x}_1^\mathsf{T}, \dots, \mathbf{x}_N^\mathsf{T}\right]^\mathsf{T} = [x_1, y_1, z_1, \dots, x_N, y_N, z_N]^\mathsf{T} \quad (1)$$

where $\mathbf{x}_i = [x_i, y_i, z_i]^\mathsf{T}$ are the object-centered Cartesian coordinates of the i-th vertex. A 3D shape model can be constructed by first bringing a set of 3D training meshes into dense correspondence so that each is described with the same number of vertexes and all samples have a shared semantic ordering. The corresponded meshes, $\{\mathbf{s}_i\}$, are then brought into a shape space by applying Generalized Procrustes Analysis and then Principal Component Analysis (PCA) is performed which results in $\{\bar{\mathbf{s}}, \mathbf{U}_s\}$, where $\bar{\mathbf{s}} \in \mathbb{R}^{3N}$ is the mean shape vector and $\mathbf{U}_s \in \mathbb{R}^{3N \times n_s}$ is the

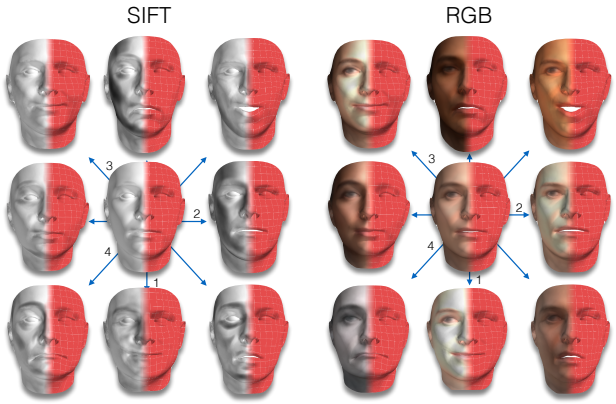

Figure 2. *Left:* The mean and first four shape and SIFT texture principal components of our "in-the-wild" SIFT texture model. *Right:* To aid in interpretation we also show the equivalent RGB basis.

orthonormal basis after keeping the first n_s principal components. This model can be used to generate novel 3D shape instances using the function $\mathcal{S} : \mathbb{R}^{n_s} \rightarrow \mathbb{R}^{3N}$ as

$$\mathcal{S}(\mathbf{p}) = \bar{\mathbf{s}} + \mathbf{U}_s \mathbf{p} \tag{2}$$

where $\mathbf{p} = [p_1, \ldots, p_{n_s}]^\mathsf{T}$ are the n_s *shape parameters*.

2.2. Camera Model

The purpose of the camera model is to map (project) the object-centered Cartesian coordinates of a 3D mesh instance $\mathbf{s}$ into 2D Cartesian coordinates on an image plane. In this work, we employ a pinhole camera model, which utilizes a perspective transformation. However, an orthographic projection model can also be used in the same way.

Perspective projection. The projection of a 3D point $\mathbf{x} = [x, y, z]^\mathsf{T}$ into its 2D location in the image plane $\mathbf{x}' = [x', y']^\mathsf{T}$ involves two steps. First, the 3D point is rotated and translated using a linear *view transformation*, under the assumption that the camera is still

$$[v_x, v_y, v_z]^\mathsf{T} = \mathbf{R}_v \mathbf{x} + \mathbf{t}_v \tag{3}$$

where $\mathbf{R}_v \in \mathbb{R}^{3\times3}$ and $\mathbf{t}_v = [t_x, t_y, t_z]^\mathsf{T}$ are the 3D rotation and translation components, respectively. Then, a non-linear *perspective transformation* is applied as

$$\mathbf{x}' = \frac{f}{v_z} \begin{bmatrix} v_x \\ v_y \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \tag{4}$$

where f is the focal length in pixel units (we assume that the x and y components of the focal length are equal) and $[c_x, c_y]^\mathsf{T}$ is the principal point that is set to the image center.

Quaternions. We parametrize the 3D rotation with quaternions [24, 40]. The quaternion uses four parameters $\mathbf{q} = [q_0, q_1, q_2, q_3]^\mathsf{T}$ in order to express a 3D rotation as

$$\mathbf{R}_v = 2 \begin{bmatrix} \frac{1}{2} - q_2^2 - q_3^2 & q_1 q_2 - q_0 q_3 & q_1 q_3 + q_0 q_2 \\ q_1 q_2 + q_0 q_3 & \frac{1}{2} - q_1^2 - q_3^2 & q_2 q_3 - q_0 q_1 \\ q_1 q_3 - q_0 q_2 & q_2 q_3 + q_0 q_1 & \frac{1}{2} - q_1^2 - q_2^2 \end{bmatrix} \tag{5}$$

Note that by enforcing a unit norm constraint on the quaternion vector, i.e. $\mathbf{q}^\mathsf{T}\mathbf{q} = 1$, the rotation matrix constraints of orthogonality with unit determinant are withheld. Given the unit norm property, the quaternion can be seen as a three-parameter vector $[q_1, q_2, q_3]^\mathsf{T}$ and a scalar $q_0 = \sqrt{1 - q_1^2 - q_2^2 - q_3^2}$. Most existing works on 3DMM parametrize the rotation matrix $\mathbf{R}_v$ using the three Euler angles that define the rotations around the horizontal, vertical and camera axes. Even thought Euler angles are more naturally interpretable, they have strong disadvantages when employed within an optimization procedure, most notably the solution ambiguity and the gimbal lock effect. Parametrization based on quaternions overcomes these disadvantages and further ensures computational efficiency, robustness and simpler differentiation.

Camera function. The projection operation performed by the camera model of the 3DMM can be expressed with the function $\mathcal{P}(\mathbf{s}, \mathbf{c}) : \mathbb{R}^{3N} \rightarrow \mathbb{R}^{2N}$, which applies the transformations of Eqs. 3 and 4 on the points of provided 3D mesh $\mathbf{s}$ with

$$\mathbf{c} = [f, q_1, q_2, q_3, t_x, t_y, t_z]^\mathsf{T} \tag{6}$$

being the vector of *camera parameters* with length $n_c = 7$. For abbreviation purposes, we represent the camera model of the 3DMM with the function $\mathcal{W} : \mathbb{R}^{n_s, n_c} \rightarrow \mathbb{R}^{2N}$ as

$$\mathcal{W}(\mathbf{p}, \mathbf{c}) \equiv \mathcal{P}\left(\mathcal{S}(\mathbf{p}), \mathbf{c}\right) \tag{7}$$

where $\mathcal{S}(\mathbf{p})$ is a 3D mesh instance using Eq. 2.

2.3. "In-the-Wild" Feature-Based Texture Model

The generation of an "in-the-wild" texture model is a key component of the proposed 3DMM. To this end, we take advantage of the existing large facial "in-the-wild" databases that are annotated in terms of sparse landmarks. Assume that for a set of M "in-the-wild" images $\{\mathbf{I}_i\}_1^M$, we have access to the associated camera and shape parameters $\{\mathbf{p_i}, \mathbf{c_i}\}$. Let us also define a *dense* feature extraction function

$$\mathcal{F} : \mathbb{R}^{H\times W} \rightarrow \mathbb{R}^{H\times W\times C} \tag{8}$$

where C is the number of channels of the feature-based image. For each image, we first compute its feature-based representation as $\mathbf{F}_i = \mathcal{F}(\mathbf{I}_i)$ and then use Eq. 7 to sample it at each vertex location to build back a vectorized texture sample $\mathbf{t}_i = \mathbf{F}_i\left(\mathcal{W}(\mathbf{p_i}, \mathbf{c_i})\right) \in \mathbb{R}^{CN}$. This texture sample

Figure 3. Building an ITW texture model

will be nonsensical for some regions mainly due to self-occlusions present in the mesh projected in the image space $\mathcal{W}(\mathbf{p_i}, \mathbf{c_i})$. To alleviate these issues, we cast a ray from the camera to each vertex and test for self-intersections with the triangulation of the mesh in order to learn a per-vertex occlusion mask $\mathbf{m}_i \in \mathbb{R}^N$ for the projected sample.

Let us create the matrix $\mathbf{X} = [\mathbf{t}_1, \ldots, \mathbf{t}_M] \in \mathbb{R}^{CN \times M}$ by concatenating the M grossly corrupted feature-based texture vectors with missing entries that are represented by the masks $\mathbf{m}_i$. To robustly build a texture model based on this heavily contaminated incomplete data, we need to recover a low-rank matrix $\mathbf{L} \in \mathbb{R}^{CN \times M}$ representing the clean facial texture and a sparse matrix $\mathbf{E} \in \mathbb{R}^{CN \times M}$ accounting for gross but sparse non-Gaussian noise such that $\mathbf{X} = \mathbf{L} + \mathbf{E}$. To simultaneously recover both $\mathbf{L}$ and $\mathbf{E}$ from incomplete and grossly corrupted observations, the Principal Component Pursuit with missing values [34] is solved

$$\begin{aligned} &\arg\min_{\mathbf{L},\mathbf{E}} \|\mathbf{L}\|_* + \lambda \|\mathbf{E}\|_1 \\ &\text{s.t. } \mathcal{P}_\Omega(\mathbf{X}) = \mathcal{P}_\Omega(\mathbf{L} + \mathbf{E}), \end{aligned} \tag{9}$$

where $\|\cdot\|_*$ denotes the nuclear norm, $\|\cdot\|_1$ is the matrix ℓ_1-norm and $\lambda > 0$ is a regularizer. Ω represents the set of locations corresponding to the observed entries of $\mathbf{X}$ (i.e., $(i,j) \in \Omega$ if $m_i = m_j = 1$). Then, $\mathcal{P}_\Omega(\mathbf{X})$ is defined as the projection of the matrix $\mathbf{X}$ on the observed entries Ω, namely $\mathcal{P}_\Omega(\mathbf{X})_{ij} = x_{ij}$ if $(i,j) \in \Omega$ and $\mathcal{P}_\Omega(\mathbf{X})_{ij} = 0$ otherwise. The unique solution of the convex optimization problem in Eq. 9 is found by employing an Alternating Direction Method of Multipliers-based algorithm [11].

The final texture model is created by applying PCA on the set of reconstructed feature-based textures acquired from the previous procedure. This results in $\{\bar{\mathbf{t}}, \mathbf{U}_t\}$, where $\bar{\mathbf{t}} \in \mathbb{R}^{CN}$ is the mean texture vector and $\mathbf{U}_t \in \mathbb{R}^{CN \times n_t}$ is the orthonormal basis after keeping the first n_t principal components. This model can be used to generate novel 3D feature-based texture instances with the function $\mathcal{T} : \mathbb{R}^{n_t} \rightarrow \mathbb{R}^{CN}$ as

$$\mathcal{T}(\boldsymbol{\lambda}) = \bar{\mathbf{t}} + \mathbf{U}_t \boldsymbol{\lambda} \tag{10}$$

where $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_{n_t}]^\mathsf{T}$ are the n_t *texture parameters*.

Finally, an iterative procedure is used in order to refine the texture. That is, we started with the 3D fits provided by using only the 2D landmarks [21]. Then, a texture model is learned using the above procedure. The texture model was used with the proposed 3DMM fitting algorithm on the same data and texture model was refined.

3. Model Fitting

We propose to fit the 3DMM on an input image using Gauss-Newton iterative optimization. To this end, herein, we first formulate the cost function and then present two optimization procedures.

3.1. Cost Function

The overall cost function of the proposed 3DMM formulation consists of a texture-based term, an optional error term based on sparse 2D landmarks and optional regularization terms on the parameters.

Texture reconstruction cost. The main term of the optimization problem is the one that aims to estimate the shape, texture and camera parameters that minimize the ${\ell_2}^2$ norm of the difference between the image feature-based texture that corresponds to the projected 2D locations of the 3D shape instance and the texture instance of the 3DMM. Let us denote by $\mathbf{F} = \mathcal{F}(\mathbf{I})$ the feature-based representation with C channels of an input image $\mathbf{I}$ using Eq. 8. Then, the texture reconstruction cost is expressed as

$$\arg\min_{\mathbf{p},\mathbf{c},\boldsymbol{\lambda}} \|\mathbf{F}\left(\mathcal{W}(\mathbf{p},\mathbf{c})\right) - \mathcal{T}(\boldsymbol{\lambda})\|^2 \tag{11}$$

Note that $\mathbf{F}\left(\mathcal{W}(\mathbf{p},\mathbf{c})\right) \in \mathbb{R}^{CN}$ denotes the operation of sampling the feature-based input image on the projected 2D locations of the 3D shape instance acquired by the camera model (Eq. 7).

Regularization. In order to avoid over-fitting effects, we augment the cost function with two optional regularization terms over the shape and texture parameters. Let us denote as $\boldsymbol{\Sigma}_s \in \mathbb{R}^{n_s \times n_s}$ and $\boldsymbol{\Sigma}_t \in \mathbb{R}^{n_t \times n_t}$ the diagonal matrices with the eigenvalues in their main diagonal for the shape and texture models, respectively. Based on the PCA nature of the shape and texture models, it is assumed that their parameters follow normal prior distributions, i.e. $\mathbf{p} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_s)$ and $\boldsymbol{\lambda} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_t)$. We formulate the regularization terms as the ${\ell_2}^2$ of the parameters' vectors weighted with the corresponding inverse eigenvalues, i.e.

$$\arg\min_{\mathbf{p},\boldsymbol{\lambda}} \; c_s \|\mathbf{p}\|^2_{\boldsymbol{\Sigma}_s^{-1}} + c_t \|\boldsymbol{\lambda}\|^2_{\boldsymbol{\Sigma}_t^{-1}} \tag{12}$$

where c_s and c_t are constants that weight the contribution of the regularization terms in the cost function.

2D landmarks cost. In order to rapidly adapt the camera parameters in the cost of Eq. 11, we further expand the

optimization problem with the term

$$\underset{\mathbf{p},\mathbf{c}}{\arg\min}\ c_l \left\| \mathcal{W}_l(\mathbf{p},\mathbf{c}) - \mathbf{s}_l \right\|^2 \tag{13}$$

where $\mathbf{s}_l = [x_1, y_1, \ldots, x_L, y_L]^\mathsf{T}$ denotes a set of L sparse 2D landmark points ($L \ll N$) defined on the image coordinate system and $\mathcal{W}_l(\mathbf{p},\mathbf{c})$ returns the $2L \times 1$ vector of 2D projected locations of these L sparse landmarks. Intuitively, this term aims to drive the optimization procedure using the selected sparse landmarks as anchors for which we have the optimal locations $\mathbf{s}_l$. This optional landmarks-based cost is weighted with the constant c_l.

Overall cost function. The overall 3DMM cost function is formulated as the sum of the terms in Eqs. 11, 12, 13, i.e.

$$\begin{aligned}\underset{\mathbf{p},\mathbf{c},\boldsymbol{\lambda}}{\arg\min}\ & \left\| \mathbf{F}\left(\mathcal{W}(\mathbf{p},\mathbf{c})\right) - \mathcal{T}(\boldsymbol{\lambda}) \right\|^2 + c_l \left\| \mathcal{W}_l(\mathbf{p},\mathbf{c}) - \mathbf{s}_l \right\|^2 + \\ & + c_s \left\| \mathbf{p} \right\|^2_{\boldsymbol{\Sigma}_s^{-1}} + c_t \left\| \boldsymbol{\lambda} \right\|^2_{\boldsymbol{\Sigma}_t^{-1}}\end{aligned} \tag{14}$$

The landmarks term as well as the regularization terms are optional and aim to facilitate the optimization procedure in order to converge faster and to a better minimum. Note that thanks to the proposed "in-the-wild" feature-based texture model, the cost function does not include any parametric illumination model similar to the ones in the relative literature [12, 13], which greatly simplifies the optimization.

3.2. Gauss-Newton Optimization

Inspired by the extensive literature in Lucas-Kanade 2D image alignment [8, 28, 30, 38, 5, 2], we formulate a Gauss-Newton optimization framework. Specifically, given that the camera projection model is applied on the image part of Eq. 14, the proposed optimization has a "forward" nature.

Parameters update. The shape, texture and camera parameters are updated in an additive manner, i.e.

$$\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p},\ \ \boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \Delta\boldsymbol{\lambda},\ \ \mathbf{c} \leftarrow \mathbf{c} + \Delta\mathbf{c} \tag{15}$$

where $\Delta\mathbf{p}$, $\Delta\boldsymbol{\lambda}$ and $\Delta\mathbf{c}$ are their increments estimated at each fitting iteration. Note that in the case of the quaternion used to parametrize the 3D rotation matrix, the update is performed as the multiplication

$$\begin{aligned}\mathbf{q} \leftarrow (\Delta\mathbf{q})\mathbf{q} &= \begin{bmatrix} \Delta q_0 \\ \Delta\mathbf{q}_{1:3} \end{bmatrix} \begin{bmatrix} q_0 \\ \mathbf{q}_{1:3} \end{bmatrix} = \\ &= \begin{bmatrix} \Delta q_0 q_0 - \Delta\mathbf{q}_{1:3}^\mathsf{T}\mathbf{q}_{1:3} \\ \Delta q_0 \mathbf{q}_{1:3} + q_0 \Delta\mathbf{q}_{1:3} + \Delta\mathbf{q}_{1:3} \times \mathbf{q}_{1:3} \end{bmatrix}\end{aligned} \tag{16}$$

However, we will still denote it as an addition for simplicity. Finally, we found that it is beneficial to keep the focal length constant in most cases, due to its ambiguity with t_z.

Linearization. By introducing the additive incremental updates on the parameters of Eq. 14, the cost function is expressed as

$$\begin{aligned}\underset{\Delta\mathbf{p},\Delta\mathbf{c},\Delta\boldsymbol{\lambda}}{\arg\min}\ & \left\| \mathbf{F}\left(\mathcal{W}(\mathbf{p}+\Delta\mathbf{p},\mathbf{c}+\Delta\mathbf{c})\right) - \mathcal{T}(\boldsymbol{\lambda}+\Delta\boldsymbol{\lambda}) \right\|^2 + \\ & + c_l \left\| \mathcal{W}_l(\mathbf{p}+\Delta\mathbf{p},\mathbf{c}+\Delta\mathbf{c}) - \mathbf{s}_l \right\|^2 + \\ & + c_s \left\| \mathbf{p}+\Delta\mathbf{p} \right\|^2_{\boldsymbol{\Sigma}_s^{-1}} + c_t \left\| \boldsymbol{\lambda}+\Delta\boldsymbol{\lambda} \right\|^2_{\boldsymbol{\Sigma}_t^{-1}}\end{aligned} \tag{17}$$

Note that the texture reconstruction and landmarks constraint terms of this cost function are non-linear due to the camera model operation. We need to linearize them around $(\mathbf{p},\mathbf{c})$ using first order Taylor series expansion at $(\mathbf{p}+\Delta\mathbf{p},\mathbf{c}+\Delta\mathbf{c}) = (\mathbf{p},\mathbf{c}) \Rightarrow (\Delta\mathbf{p},\Delta\mathbf{c}) = \mathbf{0}$. The linearization for the image term gives

$$\begin{aligned}\mathbf{F}\left(\mathcal{W}(\mathbf{p}+\Delta\mathbf{p},\mathbf{c}+\Delta\mathbf{c})\right) \approx & \mathbf{F}\left(\mathcal{W}(\mathbf{p},\mathbf{c})\right) + \\ & + \mathbf{J}_{\mathbf{F},\mathbf{p}}\Delta\mathbf{p} + \mathbf{J}_{\mathbf{F},\mathbf{c}}\Delta\mathbf{c}\end{aligned} \tag{18}$$

where $\mathbf{J}_{\mathbf{F},\mathbf{p}} = \nabla\mathbf{F}\ \frac{\partial\mathcal{W}}{\partial\mathbf{p}}\Big|_{\mathbf{p}=\mathbf{p}}$ and $\mathbf{J}_{\mathbf{F},\mathbf{c}} = \nabla\mathbf{F}\ \frac{\partial\mathcal{W}}{\partial\mathbf{c}}\Big|_{\mathbf{c}=\mathbf{c}}$ are the *image Jacobians* with respect to the shape and camera parameters, respectively. Note that most dense feature-extraction functions $\mathcal{F}(\cdot)$ are non-differentiable, thus we simply compute the gradient of the multi-channel feature image $\nabla\mathbf{F}$. Similarly, the linearization on the sparse landmarks projection term gives

$$\mathcal{W}_l(\mathbf{p}+\Delta\mathbf{p},\mathbf{c}+\Delta\mathbf{c}) \approx \mathcal{W}_l(\mathbf{p},\mathbf{c}) + \mathbf{J}_{\mathcal{W}_l,\mathbf{p}}\Delta\mathbf{p} + \mathbf{J}_{\mathcal{W}_l,\mathbf{c}}\Delta\mathbf{c} \tag{19}$$

where $\mathbf{J}_{\mathcal{W}_l,\mathbf{p}} = \frac{\partial\mathcal{W}_l}{\partial\mathbf{p}}\Big|_{\mathbf{p}=\mathbf{p}}$ and $\mathbf{J}_{\mathcal{W}_l,\mathbf{c}} = \frac{\partial\mathcal{W}_l}{\partial\mathbf{c}}\Big|_{\mathbf{c}=\mathbf{c}}$ are the *camera Jacobians*. Please refer to the supplementary material for more details on the computation of these derivatives.

3.2.1 Simultaneous

Herein, we aim to simultaneously solve for all parameters' increments. By substituting Eqs. 18 and 19 in Eq. 17 we get

$$\begin{aligned}&\underset{\Delta\mathbf{p},\Delta\mathbf{c},\Delta\boldsymbol{\lambda}}{\arg\min} \\ &\left\| \mathbf{F}\left(\mathcal{W}(\mathbf{p},\mathbf{c})\right) + \mathbf{J}_{\mathbf{F},\mathbf{p}}\Delta\mathbf{p} + \mathbf{J}_{\mathbf{F},\mathbf{c}}\Delta\mathbf{c} - \mathcal{T}(\boldsymbol{\lambda}+\Delta\boldsymbol{\lambda}) \right\|^2 + \\ &+ c_l \left\| \mathcal{W}_l(\mathbf{p},\mathbf{c}) + \mathbf{J}_{\mathcal{W}_l,\mathbf{p}}\Delta\mathbf{p} + \mathbf{J}_{\mathcal{W}_l,\mathbf{c}}\Delta\mathbf{c} - \mathbf{s}_l \right\|^2 + \\ &+ c_s \left\| \mathbf{p}+\Delta\mathbf{p} \right\|^2_{\boldsymbol{\Sigma}_s^{-1}} + c_t \left\| \boldsymbol{\lambda}+\Delta\boldsymbol{\lambda} \right\|^2_{\boldsymbol{\Sigma}_t^{-1}}\end{aligned} \tag{20}$$

Let us concatenate the parameters and their increments as $\mathbf{b} = [\mathbf{p}^\mathsf{T}, \mathbf{c}^\mathsf{T}, \boldsymbol{\lambda}^\mathsf{T}]^\mathsf{T}$ and $\Delta\mathbf{b} = [\Delta\mathbf{p}^\mathsf{T}, \Delta\mathbf{c}^\mathsf{T}, \Delta\boldsymbol{\lambda}^\mathsf{T}]^\mathsf{T}$. By taking the derivative of the final linearized cost function with respect to $\Delta\mathbf{b}$ and equalizing with zero, we get the solution

$$\mathbf{b} = -\mathbf{H}^{-1}\left(\mathbf{J}_\mathbf{F}^\mathsf{T}\mathbf{e}_\mathbf{F} + c_l\mathbf{J}_{\mathcal{W}_l}^\mathsf{T}\mathbf{e}_l + c_s\boldsymbol{\Sigma}_s^{-1}\mathbf{p} + c_t\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\lambda}\right) \tag{21}$$

where $\mathbf{H} = \mathbf{J}_\mathbf{F}{}^\mathsf{T}\mathbf{J}_\mathbf{F} + c_l\mathbf{J}_{\mathcal{W}_l}{}^\mathsf{T}\mathbf{J}_{\mathcal{W}_l} + c_s\boldsymbol{\Sigma}_s^{-1} + c_t\boldsymbol{\Sigma}_t^{-1}$ is the Hessian with

$$\begin{aligned}\mathbf{J}_\mathbf{F} &= \left[\mathbf{J}_{\mathbf{F},\mathbf{p}}^\mathsf{T}, \mathbf{J}_{\mathbf{F},\mathbf{c}}^\mathsf{T}, -\mathbf{U}_t^\mathsf{T}\right]^\mathsf{T} \\ \mathbf{J}_{\mathcal{W}_l} &= \left[\mathbf{J}_{\mathcal{W}_l,\mathbf{p}}^\mathsf{T}, \mathbf{J}_{\mathcal{W}_l,\mathbf{c}}^\mathsf{T}, \mathbf{0}_{n_t \times 2L}\right]^\mathsf{T}\end{aligned} \tag{22}$$

and

$$\begin{aligned} \mathbf{e_F} &= \mathbf{F}\left(\mathcal{W}(\mathbf{p},\mathbf{c})\right) - \mathcal{T}(\boldsymbol{\lambda}) \\ \mathbf{e}_l &= \mathcal{W}_l(\mathbf{p},\mathbf{c}) - \mathbf{s}_l \end{aligned} \tag{23}$$

are the residual terms. The computational complexity of the Simultaneous algorithm per iteration is dominated by the texture reconstruction term as $\mathcal{O}((n_s + n_c + n_t)^3 + CN(n_s + n_c + n_t)^2)$, which in practice is too slow.

3.2.2 Project-Out

We propose to use a Project-Out optimization approach that is much faster than the Simultaneous. The main idea is to optimize on the orthogonal complement of the texture subspace which will eliminate the need to solve for the texture parameters increment at each iteration. By substituting Eqs. 18 and 19 into Eq. 17 and removing the incremental update on the texture parameters as well as the texture parameters regularization term, we end up with the problem

$$\begin{aligned} \underset{\Delta\mathbf{p},\Delta\mathbf{c},\boldsymbol{\lambda}}{\arg\min}\ & \left\|\mathbf{F}\left(\mathcal{W}(\mathbf{p},\mathbf{c})\right) + \mathbf{J}_{\mathbf{F},\mathbf{p}}\Delta\mathbf{p} + \mathbf{J}_{\mathbf{F},\mathbf{c}}\Delta\mathbf{c} - \mathcal{T}(\boldsymbol{\lambda})\right\|^2 + \\ & + c_l \left\|\mathcal{W}_l(\mathbf{p},\mathbf{c}) + \mathbf{J}_{\mathcal{W}_l,\mathbf{p}}\Delta\mathbf{p} + \mathbf{J}_{\mathcal{W}_l,\mathbf{c}}\Delta\mathbf{c} - \mathbf{s}_l\right\|^2 + \\ & + c_s \left\|\mathbf{p} + \Delta\mathbf{p}\right\|^2_{\boldsymbol{\Sigma}_s^{-1}} \end{aligned} \tag{24}$$

The solution of Eq. 24 with respect to $\boldsymbol{\lambda}$ is readily given by

$$\boldsymbol{\lambda} = \mathbf{U}_t{}^{\mathsf{T}}\left(\mathbf{F}(\mathcal{W}(\mathbf{p},\mathbf{c})) + \mathbf{J}_{\mathbf{F},\mathbf{p}}\Delta\mathbf{p} + \mathbf{J}_{\mathbf{F},\mathbf{c}}\Delta\mathbf{c} - \bar{\mathbf{t}}\right) \tag{25}$$

By plugging Eq. 25 into Eq. 24, we get

$$\begin{aligned} \underset{\Delta\mathbf{p},\Delta\mathbf{c}}{\arg\min}\ & \left\|\mathbf{F}\left(\mathcal{W}(\mathbf{p},\mathbf{c})\right) + \mathbf{J}_{\mathbf{F},\mathbf{p}}\Delta\mathbf{p} + \mathbf{J}_{\mathbf{F},\mathbf{c}}\Delta\mathbf{c} - \bar{\mathbf{t}}\right\|^2_{\mathbf{P}} + \\ & + c_l \left\|\mathcal{W}_l(\mathbf{p},\mathbf{c}) + \mathbf{J}_{\mathcal{W}_l,\mathbf{p}}\Delta\mathbf{p} + \mathbf{J}_{\mathcal{W}_l,\mathbf{c}}\Delta\mathbf{c} - \mathbf{s}_l\right\|^2 + \\ & + c_s \left\|\mathbf{p} + \Delta\mathbf{p}\right\|^2_{\boldsymbol{\Sigma}_s^{-1}} \end{aligned} \tag{26}$$

where $\mathbf{P} = \mathbf{E} - \mathbf{U}_t\mathbf{U}_t{}^{\mathsf{T}}$ is the orthogonal complement of the texture subspace that functions as the "project-out" operator with $\mathbf{E}$ denoting the $CN \times CN$ unitary matrix. Note that in order to derive Eq. 26, we use the properties $\mathbf{P}^{\mathsf{T}} = \mathbf{P}$ and $\mathbf{P}^{\mathsf{T}}\mathbf{P} = \mathbf{P}$. By differentiating Eq. 26 and equalizing to zero, we get the solution

$$\begin{aligned} \Delta\mathbf{p} &= \mathbf{H}_{\mathbf{p}}{}^{-1}\left(\mathbf{J}^{\mathsf{T}}_{\mathbf{F},\mathbf{p}}\mathbf{P}\mathbf{e}_{\mathbf{F}} + c_l\mathbf{J}^{\mathsf{T}}_{\mathcal{W}_l,\mathbf{p}}\mathbf{e}_l + c_s\boldsymbol{\Sigma}_s^{-1}\mathbf{p}\right) \\ \Delta\mathbf{c} &= \mathbf{H}_{\mathbf{c}}{}^{-1}\left(\mathbf{J}^{\mathsf{T}}_{\mathbf{F},\mathbf{c}}\mathbf{P}\mathbf{e}_{\mathbf{F}} + c_l\mathbf{J}^{\mathsf{T}}_{\mathcal{W}_l,\mathbf{c}}\mathbf{e}_l\right) \end{aligned} \tag{27}$$

where

$$\begin{aligned} \mathbf{H}_{\mathbf{p}} &= \mathbf{J}^{\mathsf{T}}_{\mathbf{F},\mathbf{p}}\mathbf{P}\mathbf{J}_{\mathbf{F},\mathbf{p}} + c_l\mathbf{J}^{\mathsf{T}}_{\mathcal{W}_l,\mathbf{p}}\mathbf{J}_{\mathcal{W}_l,\mathbf{p}} + c_s\boldsymbol{\Sigma}^{-1} \\ \mathbf{H}_{\mathbf{c}} &= \mathbf{J}^{\mathsf{T}}_{\mathbf{F},\mathbf{c}}\mathbf{P}\mathbf{J}_{\mathbf{F},\mathbf{c}} + c_l\mathbf{J}^{\mathsf{T}}_{\mathcal{W}_l,\mathbf{c}}\mathbf{J}_{\mathcal{W}_l,\mathbf{c}} \end{aligned} \tag{28}$$

are the Hessian matrices and

$$\begin{aligned} \mathbf{e_F} &= \mathbf{F}\left(\mathcal{W}(\mathbf{p},\mathbf{c})\right) - \bar{\mathbf{t}} \\ \mathbf{e}_l &= \mathcal{W}_l(\mathbf{p},\mathbf{c}) - \mathbf{s}_l \end{aligned} \tag{29}$$

are the residual terms. The texture parameters can be estimated at the end of the iterative procedure using Eq. 25.

Note that the most expensive operation is $\mathbf{J}^{\mathsf{T}}_{\mathbf{F},\mathbf{p}}\mathbf{P}$. However, if we first do $\mathbf{J}^{\mathsf{T}}_{\mathbf{F},\mathbf{p}}\mathbf{U}_t$ and then multiply this result with $\mathbf{U}_t^{\mathsf{T}}$, the total cost becomes $\mathcal{O}(CNn_tn_s)$. The same stands for $\mathbf{J}^{\mathsf{T}}_{\mathbf{F},\mathbf{c}}\mathbf{P}$. Consequently, the cost per iteration is $\mathcal{O}((n_s+n_c)^3 + CNn_t(n_s+n_c) + CN(n_s+n_c)^2)$ which is much faster than the Simultaneous algorithm.

Residual masking. In practice, we apply a mask on the texture reconstruction residual of the Gauss-Newton optimization, in order to speed-up the 3DMM fitting. This mask is constructed by first acquiring the set of visible vertexes using z-buffering and then randomly selecting K of them. By keeping the number of vertexes small ($K \approx 5000 \ll N$), we manage to greatly speed-up the fitting process without any accuracy penalty.

4. KF-ITW Dataset

For the evaluation of the 3DMM, we have constructed KF-ITW, the first dataset of 3D faces captured under relatively unconstrained conditions. The dataset consists of 17 different subjects recorded under various illumination conditions performing a range of expressions (*neutral*, *happy*, *surprise*). We employed the KinectFusion [19, 29] framework to acquire a 3D representation of the subjects with a Kinect v1 sensor.

The fused mesh for each subject serves as a 3D face ground-truth in which we can evaluate our algorithm and compare it to other methods. A voxel grid of size 608^3 was utilized to get the detailed 3D scans of the faces. In order to accurately reconstruct the entire surface of the faces, a circular motion scanning pattern was carried out. Each subject was instructed to stay still in a fixed pose during the entire scanning process. The frame rate for every subject was constant to 8 frames per second. After getting the 3D scans from the KinectFusion framework we fit our shape model in a non-rigid manner to get a clear mesh with a distinct number of vertexes for the evaluation process. Finally, each mesh was manually annotated with the iBUG 49 sparse landmark set.

5. Experiments

To train our model, which we label as *ITW*, we use a variant of the Basel Face Model (BFM) [31] that we trained to contain both identities drawn from the original BFM model along with expressions provided by [15]. We trained the "in-the-wild" texture model on the images of iBUG, LFPW & AFW datasets [32] as described in Sec. 2.3 using the 3D shape fits provided by [45]. Additionally, we elect to use the project-out formulation for the throughout our experiments due its superior run-time performance and equivalent fitting performance to the simultaneous one.

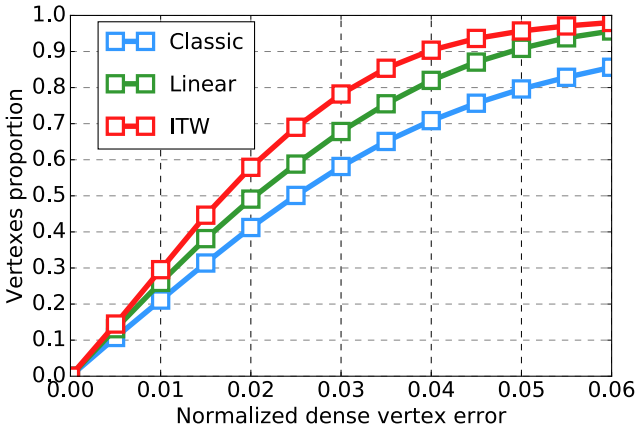

Figure 4. Accuracy results for facial shape estimation on KF-ITW database. The results are presented as Cumulative Error Distributions of the normalized dense vertex error. Table 1 reports additional measures.

Method	*AUC*	*Failure Rate (%)*
ITW	**0.678**	**1.79**
Linear	0.615	4.02
Classic	0.531	13.9

Table 1. Accuracy results for facial shape estimation on KF-ITW database. The table reports the Area Under the Curve (AUC) and Failure Rate of the Cumulative Error Distributions of Fig. 4.

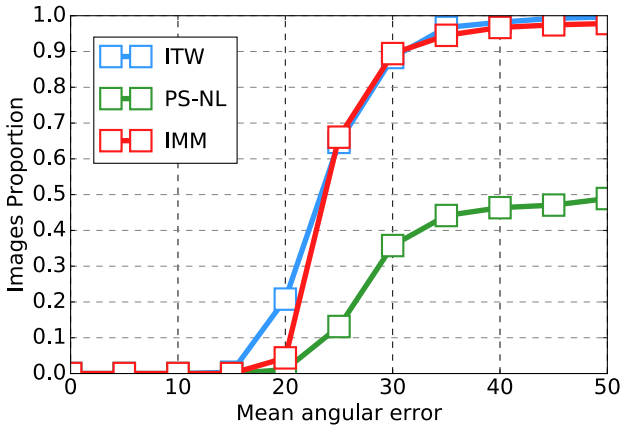

Figure 5. Results on facial surface normal estimation in the form of Cumulative Error Distribution of mean angular error.

5.1. 3D Shape Recovery

Herein, we evaluate our "in-the-wild" 3DMM (*ITW*) in terms of 3D shape estimation accuracy against two popular state-of-the-art alternative 3DMM formulations. The first one is a classic 3DMM with the original Basel laboratory texture model and full lighting equation which we term *Classic*. The second is the texture-less linear model proposed in [17, 18] which we refer to as *Linear*. For *Linear* code we use the Surrey Model with related blendshapes along with the implementation given in [18].

We use the ground-truth annotations provided in the KF-ITW dataset to initialize and fit all three techniques to the "in-the-wild" style images in the dataset. The mean mesh of each model under test is landmarked with the same 49-point markup used in the dataset, and is registered against the ground truth mesh by performing a Procrustes alignment using the sparse annotations followed by Non-Rigid Iterative Closest Point (N-ICP) to iteratively deform the two surfaces until they are brought into correspondence. This provides a per-model 'ground-truth' for the 3D shape recovery problem for each image under test. Our error metric is the per-vertex dense error between the recovered shape and the model-specific corresponded ground-truth fit, normalized by the inter-ocular distance for the test mesh. Fig. 4 shows the cumulative error distribution for this experiment for the three models under test. Table 1 reports the corresponding Area Under the Curve (AUC) and failure rates. The *Classic* model struggles to fit to the "in-the-wild" conditions present in the test set, and performs the worst. The texture-free *Linear* model does better, but the *ITW* model is most able to recover the facial shapes due to its ideal feature basis for the "in-the-wild" conditions.

Figure 6 demonstrates qualitative results on a wide range of fits of "in-the-wild" images drawn from the Helen and 300W datasets [32, 33] that qualitatively highlight the effectiveness of the proposed technique. We note that in a wide variety of expression, identity, lighting and occlusion conditions our model is able to robustly reconstruct a realistic 3D facial shape that stands up to scrutiny.

5.2. Quantitative Normal Recovery

As a second evaluation, we use our technique to find per-pixel normals and compare against two well established Shape-from-Shading (SfS) techniques: *PS-NL* [9] and *IMM* [23]. For experimental evaluation we employ images of 100 subjects from the Photoface database [42]. As a set of four illumination conditions are provided for each subject then we can generate ground-truth facial surface normals using calibrated 4-source Photometric Stereo [27]. In Fig. 5 we show the cumulative error distribution in terms of the mean angular error. *ITW* slightly outperforms *IMM* even though both *IMM* and *PS-NL* use all four available images of each subject.

6. Conclusion

We have presented a novel formulation of 3DMMs reimagined for use in "in-the-wild" conditions. We capitalise on the annotated "in-the-wild" facial databases to propose a methodology for learning an "in-the-wild" feature-based texture model suitable for 3DMM fitting without having to optimise for illumination parameters. Furthermore, we propose a novel optimisation procedure for 3DMM fitting. We

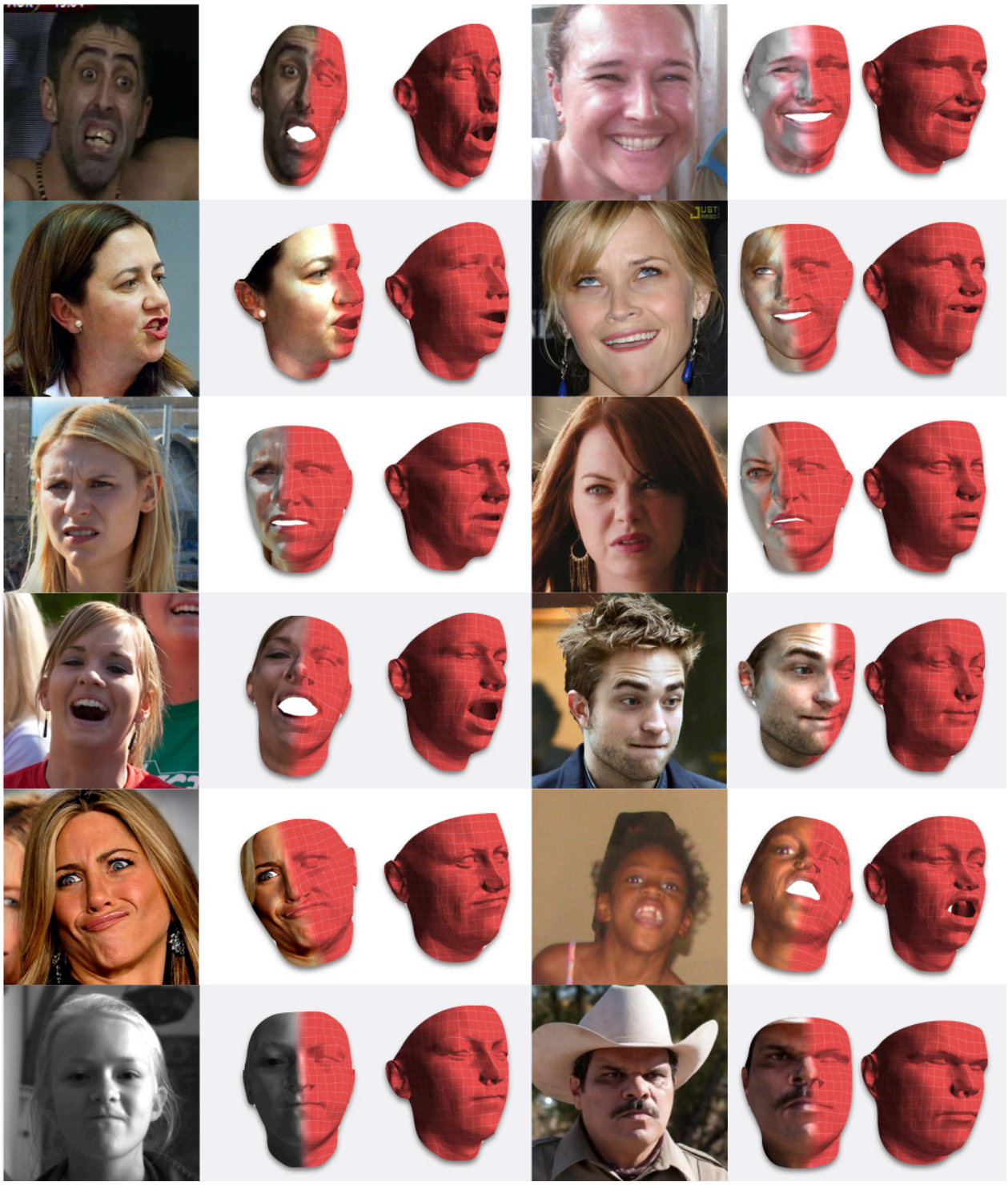

Figure 6. Examples of in the wild fits of our *ITW* 3DMM taken from 300W [32].

show that we are able to recover shapes with more detail than is possible using purely landmark-driven approaches. Our newly introduced "in-the-wild" KinectFusion dataset allows for the first time a quantitative evaluation of 3D facial reconstruction techniques in the wild, and on these evaluations we demonstrate that our in the wild formulation is state of the art, outperforming classical 3DMM approaches by a considerable margin.

References

[1] J. Alabort-i Medina, E. Antonakos, J. Booth, P. Snape, and S. Zafeiriou. Menpo: A comprehensive platform for parametric image alignment and visual deformable models. In *Proceedings of the ACM International Conference on Multimedia*, MM '14, pages 679–682, New York, NY, USA, 2014. ACM. 1, 2

[2] J. Alabort-i Medina and S. Zafeiriou. A unified framework for compositional fitting of active appearance models. *International Journal of Computer Vision*, pages 1–39, 2016. 2, 5

[3] O. Aldrian and W. A. Smith. Inverse rendering of faces with a 3d morphable model. *IEEE transactions on pattern analysis and machine intelligence*, 35(5):1080–1093, 2013. 2

[4] E. Antonakos, J. Alabort-i-Medina, G. Tzimiropoulos, and S. Zafeiriou. Hog active appearance models. In *Proceedings of IEEE International Conference on Image Processing*, pages 224–228. IEEE, 2014. 2

[5] E. Antonakos, J. Alabort-i-Medina, G. Tzimiropoulos, and S. Zafeiriou. Feature-based lucas-kanade and active appearance models. *IEEE Transactions on Image Processing*, 24(9):2617–2632, September 2015. 1, 2, 5

[6] E. Antonakos, J. Alabort-i-Medina, and S. Zafeiriou. Active pictorial structures. In *Proceedings of IEEE International Conference on Computer Vision & Pattern Recognition*, pages 5435–5444, Boston, MA, USA, June 2015. IEEE. 1

[7] A. Asthana, S. Zafeiriou, S. Cheng, and M. Pantic. Incremental face alignment in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1859–1866, 2014. 1

[8] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004. 5

[9] R. Basri, D. Jacobs, and I. Kemelmacher. Photometric stereo with general, unknown lighting. *IJCV*, 72(3):239–257, 2007. 7

[10] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar. Localizing parts of faces using a consensus of exemplars. *IEEE transactions on pattern analysis and machine intelligence*, 35(12):2930–2940, 2013. 1

[11] D. P. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014. 4

[12] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194. ACM Press/Addison-Wesley Publishing Co., 1999. 1, 5

[13] V. Blanz and T. Vetter. Face recognition based on fitting a 3d morphable model. *IEEE Transactions on pattern analysis and machine intelligence*, 25(9):1063–1074, 2003. 1, 5

[14] J. Booth, A. Roussos, S. Zafeiriou, A. Ponniah, and D. Dunaway. A 3d morphable model learnt from 10,000 faces. In *CVPR*, 2016. 2

[15] C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou. Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):413–425, 2014. 2, 6

[16] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005. 2

[17] P. Huber, Z.-H. Feng, W. Christmas, J. Kittler, and M. Rätsch. Fitting 3d morphable face models using local features. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 1195–1199. IEEE, 2015. 2, 7

[18] P. Huber, G. Hu, R. Tena, P. Mortazavian, W. P. Koppen, W. Christmas, M. Rätsch, and J. Kittler. A multiresolution 3d morphable face model and fitting framework. In *Proceedings of the 11th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2016. 7

[19] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011. 2, 6

[20] V. Jain and E. Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010. 1

[21] A. Jourabloo and X. Liu. Large-pose face alignment via cnn-based dense 3d model fitting. In *CVPR*, 2016. 4

[22] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1867–1874, 2014. 1

[23] I. Kemelmacher-Shlizerman. Internet based morphable model. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3256–3263, 2013. 1, 2, 7

[24] J. B. Kuipers et al. *Quaternions and rotation sequences*, volume 66. Princeton university press Princeton, 1999. 3

[25] V. Le, J. Brandt, Z. Lin, L. Bourdev, and T. S. Huang. Interactive facial feature localization. In *European Conference on Computer Vision*, pages 679–692. Springer, 2012. 1

[26] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999. 2

[27] D. Marr and H. K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Royal Society of London B: Biological Sciences*, 200(1140):269–294, 1978. 7

[28] I. Matthews and S. Baker. Active appearance models revisited. *International Journal of Computer Vision*, 60(2):135–164, 2004. 5

[29] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011. 2, 6

[30] G. Papandreou and P. Maragos. Adaptive and constrained algorithms for inverse compositional active appearance model fitting. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 2, 5

[31] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter. A 3d face model for pose and illumination invariant face recognition. In *Advanced video and signal based surveillance, 2009. AVSS'09. Sixth IEEE International Conference on*, pages 296–301. IEEE, 2009. 2, 6

[32] C. Sagonas, E. Antonakos, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. 300 faces in-the-wild challenge: Database and results. *Image and Vision Computing, Special Issue on Facial Landmark Localisation "In-The-Wild"*, 47:3–18, 2016. 1, 6, 7, 8

[33] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. 300 faces in-the-wild challenge: The first facial landmark localization challenge. In *Proceedings of IEEE Intl Conf. on Computer Vision (ICCV-W 2013), 300 Faces in-the-Wild Challenge (300-W)*, Sydney, Australia, December 2013. 1, 7

[34] F. Shang, Y. Liu, J. Cheng, and H. Cheng. Robust principal component analysis with missing data. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 1149–1158, New York, NY, USA, 2014. ACM. 4

[35] P. Snape, Y. Panagakis, and S. Zafeiriou. Automatic construction of robust spherical harmonic subspaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 91–100, 2015. 1, 2

[36] P. Snape and S. Zafeiriou. Kernel-pca analysis of surface normals for shape-from-shading. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1059–1066, 2014. 1

[37] G. Trigeorgis, P. Snape, M. Nicolaou, E. Antonakos, and S. Zafeiriou. Mnemonic descent method: A recurrent process applied for end-to-end face alignment. In *Proceedings of IEEE International Conference on Computer Vision & Pattern Recognition*, Las Vegas, NV, USA, June 2016. IEEE. 1

[38] G. Tzimiropoulos and M. Pantic. Optimization problems for fast aam fitting in-the-wild. In *Proceedings of the IEEE international conference on computer vision*, pages 593–600, 2013. 2, 5

[39] G. Tzimiropoulos and M. Pantic. Gauss-newton deformable part models for face alignment in-the-wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1851–1858, 2014. 1

[40] M. Wheeler and K. Ikeuchi. Iterative estimation of rotation and translation using the quaternion: School of computer science, 1995. 3

[41] X. Xiong and F. De la Torre. Supervised descent method and its applications to face alignment. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 532–539, 2013. 1

[42] S. Zafeiriou, M. Hansen, G. Atkinson, V. Argyriou, M. Petrou, M. Smith, and L. Smith. The photoface database. In *CVPR*, pages 132–139, June 2011. 7

[43] S. Zafeiriou, C. Zhang, and Z. Zhang. A survey on face detection in the wild: past, present and future. *Computer Vision and Image Understanding*, 138:1–24, 2015. 1

[44] S. Zhu, C. Li, C. Change Loy, and X. Tang. Face alignment by coarse-to-fine shape searching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4998–5006, 2015. 1

[45] X. Zhu, Z. Lei, X. Liu, H. Shi, and S. Z. Li. Face alignment across large poses: A 3d solution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 6

[46] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2879–2886. IEEE, 2012. 1

Supplementary Material for: 3D Face Morphable Models "In-the-Wild"

1. Gauss-Newton Jacobians

Herein, we derive the Jacobians used in Sec. 3.2 (Eqs. 18, 19) of the main paper. Let us first define some basic derivatives which are shared among all Jacobians.

Shape generation function. As explained in Sec. 2.1 (Eq. 2) of the main paper, the shape generation function is defined as

$$\begin{aligned}[x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_N, y_N, z_N]^\mathsf{T} \equiv \mathcal{S}(\mathbf{p}) = \\ = \bar{\mathbf{s}} + \mathbf{U}_s \mathbf{p}\end{aligned} \tag{1}$$

Perspective camera function. As mentioned in Sec. 2.2 of the main paper, we define the perspective projection function as $\mathcal{P}(\mathbf{s}, \mathbf{c}) : \mathbb{R}^{3N} \rightarrow \mathbb{R}^{2N}$ which, given a 3D shape instance (Eq. 1) it applies the following transformations

$$\begin{aligned}[x'_1, y'_1, x'_2, y'_2, \ldots, x'_N, y'_N]^\mathsf{T} \equiv \mathcal{P}(\mathcal{S}(\mathbf{p}), \mathbf{c}) = \\ = \left[f\frac{v_1^x}{v_1^z} + c_x, f\frac{v_1^y}{v_1^z} + c_y, f\frac{v_2^x}{v_2^z} + c_x, f\frac{v_2^y}{v_2^z} + c_y, \ldots, f\frac{v_N^x}{v_N^z} + c_x, f\frac{v_N^y}{v_N^z} + c_y\right]^\mathsf{T}\end{aligned} \tag{2}$$

with

$$\begin{bmatrix} v_1^x & v_2^x & \cdots & v_N^x \\ v_1^y & v_2^y & \cdots & v_N^y \\ v_1^z & v_2^z & \cdots & v_N^z \end{bmatrix} = 2 \underbrace{\begin{bmatrix} \frac{1}{2} - q_2^2 - q_3^2 & q_1 q_2 - q_0 q_3 & q_1 q_3 + q_0 q_2 \\ q_1 q_2 + q_0 q_3 & \frac{1}{2} - q_1^2 - q_3^2 & q_2 q_3 - q_0 q_1 \\ q_1 q_3 - q_0 q_2 & q_2 q_3 + q_0 q_1 & \frac{1}{2} - q_1^2 - q_2^2 \end{bmatrix}}_{\mathbf{R}_v} \begin{bmatrix} x_1 & x_2 & \cdots & x_N \\ y_1 & y_2 & \cdots & y_N \\ z_1 & z_2 & \cdots & z_N \end{bmatrix} + \underbrace{\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}}_{\mathbf{t}_v} \tag{3}$$

Note that Eqs. 2 and 3 are redefining Eqs. 3 and 4 of the main paper for a mesh with N points.

Spatial derivative of feature-based image on projected image coordinates. Using the notation of Eq. 2, this derivative is expressed as

$$\left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathcal{P}(\mathcal{S}(\mathbf{p}),\mathbf{c})} \equiv \nabla \mathbf{F} = \left[\frac{\partial \mathbf{F}}{\partial x'_1}, \frac{\partial \mathbf{F}}{\partial y'_1}, \frac{\partial \mathbf{F}}{\partial x'_2}, \frac{\partial \mathbf{F}}{\partial y'_2}, \ldots, \frac{\partial \mathbf{F}}{\partial x'_N}, \frac{\partial \mathbf{F}}{\partial y'_N}\right] \tag{4}$$

where each $\left[\frac{\partial \mathbf{F}}{\partial x'_i}, \frac{\partial \mathbf{F}}{\partial y'_i}\right], \forall i = 1, \ldots, N$ has size $CN \times 2$. This derivative denotes the spatial gradient of the feature-based image $\mathbf{F}$ with respect to the x and y coordinates of the projected image space and it has size $CN \times 2N$.

Spatial derivative of perspective projection function on 3D coordinates. By denoting the coordinates of a 3D shape (Eq. 1) as $\mathbf{x}_i = [x_i, y_i, z_i]^\mathsf{T}, \forall i = 1, \ldots, N$ and the coordinates of its 2D projection (Eq. 2) as $\mathbf{x}'_i = [x'_i, y'_i]^\mathsf{T}, \forall i = 1, \ldots, N$,

then this derivative is expressed as the block diagonal matrix

$$\left.\frac{\partial \mathcal{P}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathcal{S}(\mathbf{p})} = \left[\frac{\partial \mathcal{P}}{\partial \mathbf{x}_1}, \frac{\partial \mathcal{P}}{\partial \mathbf{x}_2}, \ldots, \frac{\partial \mathcal{P}}{\partial \mathbf{x}_N}\right] = \begin{bmatrix} \frac{\partial \mathbf{x}'_1}{\partial \mathbf{x}_1} & \mathbf{0}_{2\times 3} & \cdots & \mathbf{0}_{2\times 3} \\ \mathbf{0}_{2\times 3} & \frac{\partial \mathbf{x}'_2}{\partial \mathbf{x}_2} & \cdots & \mathbf{0}_{2\times 3} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{2\times 3} & \mathbf{0}_{2\times 3} & \cdots & \frac{\partial \mathbf{x}'_N}{\partial \mathbf{x}_N} \end{bmatrix} \tag{5}$$

with size $2N \times 3N$. Each 2×3 derivative $\frac{\partial \mathbf{x}'_i}{\partial \mathbf{x}_i}$, $\forall i = 1, \ldots, N$ is defined as

$$\frac{\partial \mathbf{x}'_i}{\partial \mathbf{x}_i} = \begin{bmatrix} f\dfrac{\frac{\partial v_i^x}{\partial \mathbf{x}_i} v_i^z - v_i^x \frac{\partial v_i^z}{\partial \mathbf{x}_i}}{{v_i^z}^2} \\ f\dfrac{\frac{\partial v_i^y}{\partial \mathbf{x}_i} v_i^z - v_i^y \frac{\partial v_i^z}{\partial \mathbf{x}_i}}{{v_i^z}^2} \end{bmatrix} \tag{6}$$

where

$$\begin{aligned} \frac{\partial v_i^x}{\partial \mathbf{x}_i} &= \left[1 - 2q_2^2 - 2q_3^2,\ 2q_1q_2 - 2q_0q_3,\ 2q_1q_3 + 2q_0q_2\right] \\ \frac{\partial v_i^y}{\partial \mathbf{x}_i} &= \left[2q_1q_2 + 2q_0q_3,\ 1 - 2q_1^2 - 2q_3^2,\ 2q_2q_3 - 2q_0q_1\right] \\ \frac{\partial v_i^z}{\partial \mathbf{x}_i} &= \left[2q_1q_3 - 2q_0q_2,\ 2q_2q_3 + 2q_0q_1,\ 1 - 2q_1^2 - 2q_2^2\right] \end{aligned} \tag{7}$$

Derivative of perspective projection function with respect to camera parameters. From the definition of the perspective camera function in Eq. 2, this derivative has the form

$$\left.\frac{\partial \mathcal{P}}{\partial \mathbf{c}}\right|_{\mathbf{c}=\mathbf{c}} = \left[\frac{\partial \mathcal{P}}{\partial f}, \frac{\partial \mathcal{P}}{\partial q_1}, \frac{\partial \mathcal{P}}{\partial q_2}, \frac{\partial \mathcal{P}}{\partial q_3}, \frac{\partial \mathcal{P}}{\partial t_x}, \frac{\partial \mathcal{P}}{\partial t_y}, \frac{\partial \mathcal{P}}{\partial t_z}\right] \tag{8}$$

with size $2N \times n_c$.

The derivative with respect to the focal length is

$$\frac{\partial \mathcal{P}}{\partial f} = \left[\frac{v_1^x}{v_1^z}, \frac{v_1^y}{v_1^z}, \frac{v_2^x}{v_2^z}, \frac{v_2^y}{v_2^z}, \ldots, \frac{v_N^x}{v_N^z}, \frac{v_N^y}{v_N^z}\right]^\mathsf{T} \tag{9}$$

with size $2N \times 1$.

The derivatives with respect to the quaternion parameters q_1, q_2, q_3 are

$$\frac{\partial \mathcal{P}}{\partial q_1} = f\left[\frac{\frac{\partial v_1^x}{\partial q_1} v_1^z - v_1^x \frac{\partial v_1^z}{\partial q_1}}{{v_1^z}^2}, \frac{\frac{\partial v_1^y}{\partial q_1} v_1^z - v_1^y \frac{\partial v_1^z}{\partial q_1}}{{v_1^z}^2}, \ldots, \frac{\frac{\partial v_N^x}{\partial q_1} v_N^z - v_N^x \frac{\partial v_N^z}{\partial q_1}}{{v_N^z}^2}, \frac{\frac{\partial v_N^y}{\partial q_1} v_N^z - v_N^y \frac{\partial v_N^z}{\partial q_1}}{{v_N^z}^2}\right]^\mathsf{T} \tag{10}$$

$$\frac{\partial \mathcal{P}}{\partial q_2} = f\left[\frac{\frac{\partial v_1^x}{\partial q_2} v_1^z - v_1^x \frac{\partial v_1^z}{\partial q_2}}{{v_1^z}^2}, \frac{\frac{\partial v_1^y}{\partial q_2} v_1^z - v_1^y \frac{\partial v_1^z}{\partial q_2}}{{v_1^z}^2}, \ldots, \frac{\frac{\partial v_N^x}{\partial q_2} v_N^z - v_N^x \frac{\partial v_N^z}{\partial q_2}}{{v_N^z}^2}, \frac{\frac{\partial v_N^y}{\partial q_2} v_N^z - v_N^y \frac{\partial v_N^z}{\partial q_2}}{{v_N^z}^2}\right]^\mathsf{T} \tag{11}$$

$$\frac{\partial \mathcal{P}}{\partial q_3} = f \left[\frac{\frac{\partial v_1^x}{\partial q_3} v_1^z - v_1^x \frac{\partial v_1^z}{\partial q_3}}{{v_1^z}^2}, \frac{\frac{\partial v_1^y}{\partial q_3} v_1^z - v_1^y \frac{\partial v_1^z}{\partial q_3}}{{v_1^z}^2}, \ldots, \frac{\frac{\partial v_N^x}{\partial q_3} v_N^z - v_N^x \frac{\partial v_N^z}{\partial q_3}}{{v_N^z}^2}, \frac{\frac{\partial v_N^y}{\partial q_3} v_N^z - v_N^y \frac{\partial v_N^z}{\partial q_3}}{{v_N^z}^2} \right]^\mathsf{T} \tag{12}$$

with sizes $2N \times 1$, where

$$\begin{aligned}
\left[\frac{\partial v_i^x}{\partial q_1}, \frac{\partial v_i^x}{\partial q_2}, \frac{\partial v_i^x}{\partial q_3}\right] &= 2\left[q_2 y_i + q_3 z_i, \ -2q_2 x_i + q_1 y_i + q_0 z_i, \ -2q_3 x_i - q_0 y_i + q_1 z_i\right] \\
\left[\frac{\partial v_i^y}{\partial q_1}, \frac{\partial v_i^y}{\partial q_2}, \frac{\partial v_i^y}{\partial q_3}\right] &= 2\left[q_2 x_i - 2q_1 y_i - q_0 z_i, \ q_1 x_i + q_3 z_i, \ q_0 x_i - 2q_3 y_i + q_2 z_i\right] \\
\left[\frac{\partial v_i^z}{\partial q_1}, \frac{\partial v_i^z}{\partial q_2}, \frac{\partial v_i^z}{\partial q_3}\right] &= 2\left[q_3 x_i + q_0 y_i - 2q_1 z_i, \ -q_0 x_i + q_3 y_i - 2q_2 z_i, \ q_1 x_i + q_2 y_i\right] \\
\forall i &= 1, \ldots, N
\end{aligned} \tag{13}$$

The derivatives with respect to the translation vector $\mathbf{t}_v = [t_x, t_y, t_z]^\mathsf{T}$ are

$$\begin{aligned}
\frac{\partial \mathcal{P}}{\partial t_x} &= f\left[\frac{1}{v_1^z}, 0, \frac{1}{v_2^z}, 0, \ldots, \frac{1}{v_N^z}, 0\right]^\mathsf{T} \\
\frac{\partial \mathcal{P}}{\partial t_y} &= f\left[0, \frac{1}{v_1^z}, 0, \frac{1}{v_2^z}, \ldots, 0, \frac{1}{v_N^z}\right]^\mathsf{T} \\
\frac{\partial \mathcal{P}}{\partial t_z} &= -f\left[\frac{v_1^x}{{v_1^z}^3}, \frac{v_1^y}{{v_1^z}^3}, \frac{v_2^x}{{v_2^z}^3}, \frac{v_2^y}{{v_2^z}^3}, \ldots, \frac{v_N^x}{{v_N^z}^3}, \frac{v_N^y}{{v_N^z}^3}\right]^\mathsf{T}
\end{aligned} \tag{14}$$

with size $2N \times 1$ each.

Derivative of 3D shape generation function with respect to shape parameters. This is the derivative of Eq. 1 with respect to the shape parameters, which is simply

$$\left.\frac{\partial \mathcal{S}}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}} = \mathbf{U}_s \tag{15}$$

with size $3N \times n_s$.

1.1. Image Jacobians

Herein, we define the image Jacobians that are used in Sec. 3.2 (Eq. 18) of the main paper. Specifically, using the chain rule, the image Jacobian with respect to the shape parameters takes the form

$$\mathbf{J}_{\mathbf{F},\mathbf{p}} = \left.\frac{\partial \mathbf{F}(\mathcal{W}(\mathbf{p},\mathbf{c}))}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}} = \left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathcal{P}(\mathcal{S}(\mathbf{p}),\mathbf{c})} \left.\frac{\partial \mathcal{P}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathcal{S}(\mathbf{p})} \left.\frac{\partial \mathcal{S}}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}} \tag{16}$$

where the three components are given from Eqs. 4, 5 and 15, respectively. The Jacobian has size $CN \times n_s$.

Similarly, using the chain rule, the image Jacobian with respect to the camera parameters takes the form

$$\mathbf{J}_{\mathbf{F},\mathbf{c}} = \left.\frac{\partial \mathbf{F}(\mathcal{W}(\mathbf{p},\mathbf{c}))}{\partial \mathbf{c}}\right|_{\mathbf{c}=\mathbf{c}} = \left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathcal{P}(\mathcal{S}(\mathbf{p}),\mathbf{c})} \left.\frac{\partial \mathcal{P}}{\partial \mathbf{c}}\right|_{\mathbf{c}=\mathbf{c}} \tag{17}$$

where the two components are given from Eqs. 4, and 8, repsectively. The Jacobian has size $CN \times n_c$.

1.2. Landmarks Jacobians

Herein we define the Jacobians of the landmarks term that are used in Sec. 3.2 (Eq. 19) of the main paper. First, let us define the 3D landmarks generation function $\mathcal{S}_l(\mathbf{p}) : \mathbb{R}^{n_s} \rightarrow \mathbb{R}^{3L}$ which generates a 3D instance of L sparse landmarks by slicing the linear combination defined in Eq. 1. Then the perspective camera function is defined as $\mathcal{W}_l(\mathbf{p},\mathbf{c}) = \mathcal{P}(\mathcal{S}_l(\mathbf{p}),\mathbf{c})$ and it has the same formulation as in Eq. 2.

Specifically, using the chain rule, the landmarks Jacobian with respect to the shape parameters takes the form

$$\mathbf{J}_{\mathcal{W}_l,\mathbf{p}} = \left.\frac{\partial \mathcal{W}_l(\mathbf{p},\mathbf{c})}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}} = \left.\frac{\partial \mathcal{P}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathcal{S}_l(\mathbf{p})} \left.\frac{\partial \mathcal{S}_l}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}} \tag{18}$$

where the two components are given from Eqs. 5 and 15 for the L sparse landmarks. The Jacobian has size $2L \times n_s$.

Similarly, using the chain rule, the landmarks Jacobian with respect to the camera parameters takes the form

$$\mathbf{J}_{\mathcal{W}_l,\mathbf{c}} = \left.\frac{\partial \mathcal{W}_l(\mathbf{p},\mathbf{c})}{\partial \mathbf{c}}\right|_{\mathbf{c}=\mathbf{c}} = \left.\frac{\partial \mathcal{P}}{\partial \mathbf{c}}\right|_{\mathbf{c}=\mathbf{c}} \tag{19}$$

which is given by Eq. 8 for the L sparse landmarks. It has size $2L \times n_c$.

2. Qualitative 3DMM fit

In the main paper, Fig. 4 shows a qualitative evaluation of three 3D Morphable Models on our new KF-ITW dataset. In Fig. 1 in this document, we present an example image from this experiment, along with the ground truth scan from Kinect Fusion, and the resulting fits from the three models under test.

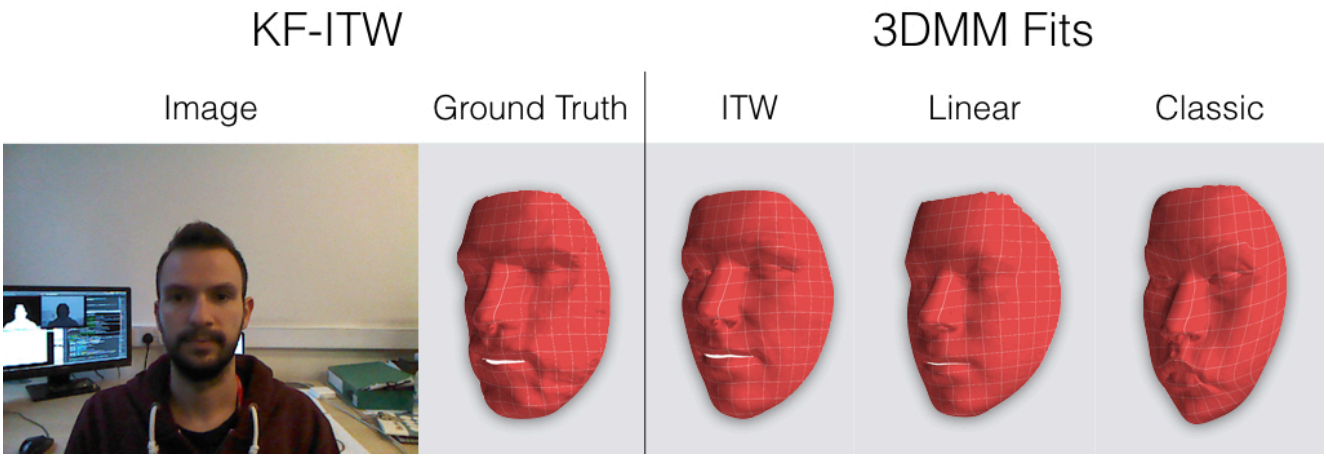

Figure 1. *Left:* An example RGB image and the complementary ground-truth 3D Kinect Fusion from our *KF-ITW* dataset. *Right:* The 3D shape recovered using the three methods under test.

3. Additional In-The-Wild texture extraction examples

In the main paper, Fig. 3 shows one example of how we extracted textures from "in-the-wild" images in order to form our robust texture basis. In Fig. 2 in this document, we present five more example texture extractions. For each, we show:

- The source "in-the-wild" image (left)
- The sampled texture from the image with mask (center)
- The final reconstructed texture used after completion with Robust PCA with Missing Values (right)

For clarification, we note the following:

- Our texture model is per-vertex — the "unwrapped" nature of the textures in these figures is purely a way to visualise these per-vertex texture models.
- The central column shows the colour per-vertex as extracted from the image when self occlusion is not taken into account. The transparent red overly shows the masked region as calculated by ray casting between each vertex and the camera.
- We actually used SIFT features for our texture model as explained in the main paper, but these RGB visualizations are much easier to interpret.

Figure 2. Five examples of "in-the-wild" images taken from AFW along with the textures and masks and final complete reconstructed textures extracted from them as outlined in Sec. 2.3 of the original paper.