International Journal of Computer Vision 75(1), 93–113, 2007

DOI: 10.1007/s11263-007-0043-2

2D vs. 3D Deformable Face Models: Representational Power, Construction, and Real-Time Fitting

IAIN MATTHEWS
The Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213
iainm@cs.cmu.edu

JING XIAO
Epson Palo Alto Laboratory, Epson Research & Development, 2580 Orchard Parkway, Suite 225, San Jose, CA 95131
jing.xiaoj@gmail.com

SIMON BAKER*
Microsoft Research, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6339
sbaker@microsoft.com

Received March 09, 2006; Accepted January 26, 2007

First online version published in February, 2007

Abstract. Model-based face analysis is a general paradigm with applications that include face recognition, expression recognition, lip-reading, head pose estimation, and gaze estimation. A face model is first constructed from a collection of training data, either 2D images or 3D range scans. The face model is then fit to the input image(s) and the model parameters used in whatever the application is. Most existing face models can be classified as either 2D (e.g. Active Appearance Models) or 3D (e.g. Morphable Models). In this paper we compare 2D and 3D face models along three axes: (1) representational power, (2) construction, and (3) real-time fitting. For each axis in turn, we outline the differences that result from using a 2D or a 3D face model.

Keywords: Model-based face analysis, 2D Active Appearance Models, 3D Morphable Models, representational power, model construction, non-rigid structure-from-motion, factorization, real-time fitting, the inverse compositional algorithm, constrained fitting

1. Introduction

Model-based face analysis is a general paradigm with numerous applications. Perhaps the most well known face models are 2D Active Appearance Models (AAMs) (Cootes et al., 2001) and 3D Morphable Models (3DMMs) (Blanz and Vetter, 1999). A face model is first constructed from either a set of 2D training images (Cootes et al., 2001) or a set of 3D range scans (Blanz and Vetter, 1999). The face model is then fit to the input image(s) and the model parameters are used in whatever the application is. In Lanitis et al. (1997), the same face model was used for face recognition, pose estimation, and expression recognition. Other applications include lip-reading (Matthews et al., 2002) and gaze estimation (Ishikawa et al., 2004).

AAMs and 3DMMs are similar in many ways. Both consist of a linear shape model and a linear appearance (texture) model. The main difference between them is that the shape component of an AAM is 2D, whereas the shape component of a 3DMM is 3D. A natural question, then, is "what are the relative advantages and disadvantages of 2D and 3D face models?" In this paper, we attempt to answer this question by comparing 2D and 3D face models along three different axes: (1) representational power, (2) construction, and (3) real-time fitting. With

*Corresponding author.

the exception of the background material in Section 2, our discussion is on the level of general 2D and 3D linear face models, rather than comparing specific models such as AAMs and 3DMMS.

Note that there are other differences between 2D and 3D models, beyond those covered in this paper. One benefit of a 3D model is the ability to model surface normals, and hence use physically-based models of illumination and reflectance. The combination of a 3D shape model and a physically-based reflectance model is one of the main reasons 3D Morphable Models are so effective (Blanz and Vetter, 1999). The comparison in this paper does not include such illumination effects, but it largely limited to geometric differences. Another difference not considered in this paper is the ease of handling occlusions. The reader is referred to a related technical report by the authors (Matthews et al., 2006) for an empirical comparison of occlusion handing in 2D and 3D.

In Section 3 we compare the representational power of 2D and 3D linear face models. We prove 3 main results. (1) Under the scaled orthographic image formation model, 2D and 3D face models have the same power to represent 3D objects; i.e. any shape model that can be generated by a 3D model can also be generated by a 2D model. (2) Up to 6 times as many parameters may be required by the 2D model to represent the same phenomenon as the 3D model. (3) In general, a 2D model can generate model instances that are not possible to generate with the corresponding 3D model; i.e. some (actually most) 2D model instances are "physically unrealizable."

In Section 4 we compare the construction of 2D and 3D models. We first investigate how a 2D face model can be converted into a 3D model. We explain how the fact that a 2D model can generate model instances that are not possible to generate from the corresponding 3D model means that a 2D model cannot be upgraded to 3D in isolation. Instead we show how a non-rigid structure-from-motion algorithm can be used to construct a 3D face model indirectly from a 2D model through 2D tracking results. We also present an algorithm for the construction of a 2D model from a 3D model and show how it can be used to build 2D face models that better model rotated faces.

In Section 5 we study the real-time fitting of face models. We begin by reviewing the efficient "inverse compositional" 2D fitting algorithm (Matthews and Baker, 2004). We then show that the naive 3D generalization of this algorithm violates one of the key assumptions made by the 2D algorithm. The 3D generalization is therefore not a correct algorithm. We describe a "work around" to this problem, the simultaneous fitting of both a 2D and a 3D shape model. We show how this formulation of the 3D fitting problem does lead to a real-time implementation. We empirically compare the 2D fitting algorithm with its 3D extension. The results show that fitting a 3D model is: (1) inherently more robust than fitting a 2D model, and (2) takes fewer iterations to converge.

In Section 6 we summarize the main conclusions of the paper and in Section 7 we briefly describe how the research described in this paper was influenced by the work of Takeo Kanade.

2. Background

We begin with a brief review of 2D Active Appearance Models (AAMs) (Cootes et al., 2001) and 3D Morphable Models (3DMMs) (Blanz and Vetter, 1999). We have taken the liberty to simplify the presentation and change the notation from Cootes et al. (2001) and Blanz and Vetter (1999) to highlight the similarities and differences between the two types of models. Note that our definition of an AAM corresponds to that of an "Independent AAM" in Matthews and Baker (2004).

2.1. 2D Active Appearance Models

The *2D shape* of an Active Appearance Model (AAM) is defined by a 2D triangulated mesh and in particular the vertex locations of the mesh. Mathematically, the shape $\mathbf{s}$ of an AAM is defined as the 2D coordinates of the n vertices that make up the mesh:

$$\mathbf{s} = \begin{pmatrix} u_1 & u_2 & \dots & u_n \\ v_1 & v_2 & \dots & v_n \end{pmatrix}. \quad (1)$$

AAMs allow linear shape variation. This means that the shape matrix $\mathbf{s}$ can be expressed as a base shape $\mathbf{s}_0$ plus a linear combination of m shape matrices $\mathbf{s}_i$:

$$\mathbf{s} = \mathbf{s}_0 + \sum_{i=1}^{m} p_i \, \mathbf{s}_i \quad (2)$$

where the coefficients $\mathbf{p} = (p_1, \dots, p_m)^{\mathrm{T}}$ are the shape parameters.

AAMs are normally computed from training data consisting of a set of images with the shape mesh (usually hand) marked on them Cootes et al. (2001). The Iterative Procrustes Algorithm and Principal Component Analysis (PCA) are then applied to compute the base shape $\mathbf{s}_0$ and the shape variation $\mathbf{s}_i$. The base shape $\mathbf{s}_0$ is the mean shape computed by the Procrustes algorithm and the matrices $\mathbf{s}_i$ are the (reshaped) PCA eigenvectors corresponding to the m largest eigenvalues.

The *appearance* of the AAM is defined within the base mesh $\mathbf{s}_0$. Let $\mathbf{s}_0$ also denote the set of pixels $\mathbf{u} = (u, v)^{\mathrm{T}}$ that lie inside the base mesh $\mathbf{s}_0$, a convenient notational shortcut. The appearance of the AAM is then an image $A(\mathbf{u})$ defined over the pixels $\mathbf{u} \in \mathbf{s}_0$. AAMs allow linear appearance variation. This means that the appearance

$A(\mathbf{u})$ can be expressed as a base appearance $A_0(\mathbf{u})$ plus a linear combination of l appearance images $A_i(\mathbf{u})$:

$$A(\mathbf{u}) \;=\; A_0(\mathbf{u}) + \sum_{i=1}^{l} \lambda_i A_i(\mathbf{u}) \tag{3}$$

where the coefficients $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_l)^T$ are the appearance parameters. The base appearance A_0 and appearance modes A_i are usually computed by applying PCA to the shape normalized training images (Cootes et al., 2001). The base appearance A_0 is the mean appearance and the appearance images A_i are the PCA eigenvectors corresponding to the l largest eigenvalues.

Although Eqs. (2) and (3) describe the AAM shape and appearance variation, they do not describe how to generate an AAM *model instance.* AAMs use a simple 2D image formation model (sometimes called a normalization), a 2D similarity transformation $\mathbf{N}(\mathbf{u}; \mathbf{q})$, where $\mathbf{q} = (q_1, \ldots, q_4)^T$ contains the rotation, translation, and scale parameters (Matthews and Baker, 2004). Given the AAM shape parameters $\mathbf{p} = (p_1, \ldots, p_m)^T$, Eq. (2) is used to generate the shape of the AAM $\mathbf{s}$. The shape $\mathbf{s}$ is then mapped into the image with the similarity transformation to give $\mathbf{N}(\mathbf{s}; \mathbf{q})$. Equation (3) is used to generate the AAM appearance $A(\mathbf{u})$ from the AAM appearance parameters $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_l)^T$. The AAM model instance with shape parameters $\mathbf{p}$, image formation parameters $\mathbf{q}$, and appearance parameters $\boldsymbol{\lambda}$ is then created by warping the appearance $A(\mathbf{u})$ from the base mesh $\mathbf{s}_0$ to the model shape mesh in the image $\mathbf{N}(\mathbf{s}; \mathbf{q})$. In particular, the pair of meshes $\mathbf{s}_0$ and $\mathbf{N}(\mathbf{s}; \mathbf{q})$ define a piecewise affine warp from $\mathbf{s}_0$ to $\mathbf{N}(\mathbf{s}; \mathbf{q})$ which we denote $\mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p}); \mathbf{q})$. For each triangle in $\mathbf{s}_0$ there is a corresponding triangle in $\mathbf{N}(\mathbf{s}; \mathbf{q})$ and each pair of triangles defines a unique affine warp. See Matthews and Baker (2004) for more details.

2.2. *3D Morphable Models*

We now describe the 3D model that we use in this paper. We describe it as a 3D Morphable Model to give full credit to the seminal work in Blanz and Vetter (1999). But note that the model here is a subset of the model in Blanz and Vetter (1999). For example, we do not include a reflectance model, like the Phong model used in Blanz and Vetter (1999). Our goal is to define a 3D model that uses the geometric aspects of 3D Morphable Models (Blanz and Vetter, 1999) but is also the direct 3D equivalent of the 2D Active Appearance Model in Section 2.1; i.e. the models should be as close as possible, except in the dimension of the shape model.

The *3D shape* of a 3D Morphable Model (3DMM) is defined by a 3D triangulated mesh. Mathematically, we define the shape $\overline{\mathbf{s}}$ ("overline" or "bar" is used throughout this paper to denote the 3D model equivalent of a 2D model quantity) of a 3DMM as the 3D coordinates:

$$\overline{\mathbf{s}} \;=\; \begin{pmatrix} x_1 & x_2 & \ldots & x_{\overline{n}} \\ y_1 & y_2 & \ldots & y_{\overline{n}} \\ z_1 & z_2 & \ldots & z_{\overline{n}} \end{pmatrix} \tag{4}$$

of the $\overline{n}$ vertices that make up the mesh. 3DMMs allow linear shape variation. The shape matrix $\overline{\mathbf{s}}$ can be expressed as a base shape $\overline{\mathbf{s}}_0$ plus a linear combination of $\overline{m}$ shape matrices $\overline{\mathbf{s}}_i$:

$$\overline{\mathbf{s}} \;=\; \overline{\mathbf{s}}_0 + \sum_{i=1}^{\overline{m}} \overline{p}_i\, \overline{\mathbf{s}}_i. \tag{5}$$

where the coefficients $\overline{\mathbf{p}} = (\overline{p}_1, \ldots, \overline{p}_{\overline{m}})^T$ are the 3D shape parameters.

3DMMs are normally computed from training data consisting of a number of 3D *range* scans with the mesh vertices located in them Blanz and Vetter (1999). PCA is then applied to the 3D coordinates of the training meshes. The base shape $\overline{\mathbf{s}}_0$ is the mean shape and the matrices $\overline{\mathbf{s}}_i$ are the (reshaped) eigenvectors corresponding to the $\overline{m}$ largest eigenvalues.

The *appearance* of a 3DMM is defined within a 2D texture-map image. In this texture-map, there is a 2D triangulated mesh that has the same topology (vertex connectivity) as the base 3D mesh $\overline{\mathbf{s}}_0$. Each 2D mesh point in the texture map is then associated with its corresponding 3D mesh vertex to define a 3D triangulated appearance model. Let $\overline{\mathbf{s}}_0^*$ denote the set of pixels $\mathbf{u} = (u, v)^T$ that lie inside this 2D mesh. The appearance of the 3DMM is then an image $\overline{A}(\mathbf{u})$ defined over the pixels $\mathbf{u} \in \overline{\mathbf{s}}_0^*$. 3DMMs also allow linear appearance variation. The appearance $\overline{A}(\mathbf{u})$ can be expressed as a base appearance $\overline{A}_0(\mathbf{u})$ plus a linear combination of $\overline{l}$ appearance images $\overline{A}_i(\mathbf{u})$:

$$\overline{A}(\mathbf{u}) \;=\; \overline{A}_0(\mathbf{u}) + \sum_{i=1}^{\overline{l}} \overline{\lambda}_i\, \overline{A}_i(\mathbf{u}) \tag{6}$$

where the coefficients $\overline{\boldsymbol{\lambda}} = (\overline{\lambda}_1, \ldots, \overline{\lambda}_{\overline{l}})^T$ are the appearance parameters. The base appearance $\overline{A}_0$ and the appearance images $\overline{A}_i$ are computed by applying PCA to the texture components of the range scans, appropriated warped onto the 2D triangulated mesh $\overline{\mathbf{s}}_0^*$ (Blanz and Vetter, 1999).

To generate a 3DMM *model instance* an image formation model is needed to convert the 3D shape $\overline{\mathbf{s}}$ into a 2D mesh, onto which the appearance is warped. As in Romdhani and Vetter (2003), we use the scaled orthographic model defined by the projection matrix:

$$\mathbf{P} \;=\; \begin{pmatrix} i_x & i_y & i_z \\ j_x & j_y & j_z \end{pmatrix} \tag{7}$$

and the offset of the origin $(o_u, o_v)^T$. The two vectors $\mathbf{i} = (i_x, i_y, i_z)$ and $\mathbf{j} = (j_x, j_y, j_z)$ are the projection axes. We require that the projection axes are equal length and orthogonal; i.e. we require that $\mathbf{i} \cdot \mathbf{i} = i_x i_x + i_y i_y + i_z i_z = j_x j_x + j_y j_y + j_z j_z = \mathbf{j} \cdot \mathbf{j}$ and $\mathbf{i} \cdot \mathbf{j} = i_x j_x + i_y j_y + i_z j_z = 0$. The result of imaging the 3D point $\mathbf{x} = (x, y, z)^T$ with this scaled orthographic model is the 2D point:

$$\mathbf{u} = \mathbf{P}\mathbf{x} + \begin{pmatrix} o_u \\ o_v \end{pmatrix}. \tag{8}$$

Note that the scaled orthographic projection has 6 degrees of freedom which can be mapped onto a 3D pose (yaw, pitch, roll), a 2D translation, and a scale. The 3DMM model instance is then computed as follows. Given the shape parameters $\overline{p}_i$, the 3D shape $\overline{\mathbf{s}}$ is computed using Eq. (5). Each 3D vertex $(x_i, y_i, z_i)^T$ is then mapped to a 2D vertex using Eq. (8). (Note that during this process the visibility of the triangles in the mesh should be computed and respected.) The appearance is then computed using Eq. (6) and warped onto the 2D mesh using the piecewise affine warp defined by the mapping from the 2D vertices in $\overline{\mathbf{s}}_0^*$ to the corresponding 2D vertices computed by the applying Eq. (8) to the 3D shape $\overline{\mathbf{s}}$.

2.3. *Similarities and Differences*

AAMs and 3DMMs are similar in many ways. They both consist of a linear shape model and a linear appearance model. In particular, Eqs. (2) and (5) are almost identical. Equations (3) and (6) are also almost identical. The main difference between them is that the shape component of the AAM is 2D (see Eq. (1)) whereas that of the 3DMM is 3D (see Eq. (4)).

Note that there are other differences between 2D AAMs (Cootes et al., 2001) and 3DMMs (Blanz and Vetter, 1999). For example, (1) 3DMMs are usually constructed to be denser; i.e. consist of more triangles, and (2) because of their 3D shape and density, 3DMMs can also use the surface normal in their appearance model. In this paper, we address the differences between 2D and 3D face models in general, rather than specifically comparing AAMs and 3DMMS.

3. Representational Power

It is a common preconception that 3D models are "more powerful" than 2D models. In this section, we investigate to what extent this is really the case (for the scaled orthographic image formation model in Eq. (8).) The shape of either a 2D or 3D model instance is a vector of 2D points in an image. In Section 3.1 we show that the set of all such 2D point vectors generated by a 3D model can also be generated by a 2D model.[1] On the other hand, we also show that 3D models are more compact than 2D models. In particular, we show that up to approximately 6 times as many shape parameters may be needed by a 2D model to represent the same phenomenon represented by a 3D model. Finally, in Section 3.2 we show that, in general, when a 2D model is used to model a 3D phenomenon the resulting 2D model can generate model instances that are not physically realizable (in the sense that there are valid 2D model instances that are not valid 3D model instances.) Note that the analysis in this section ignores occlusion which is covered in Matthews et al. (2006).

3.1. *Equivalence and Number of Parameters*

The shape variation of a 2D model is described by Eq. (2) and $\mathbf{N}(\mathbf{u}; \mathbf{q})$. That of an 3D model is described by Eqs. (5) and (8). We can ignore the offset of the origin $(o_u, o_v)^T$ in the scaled orthographic model because this offset corresponds to a translation which can be modeled by the 2D similarity transformation $\mathbf{N}(\mathbf{u}; \mathbf{q})$. The 2D shape variation of the 3D model is then:

$$\begin{pmatrix} i_x & i_y & i_z \\ j_x & j_y & j_z \end{pmatrix} \cdot \left(\overline{\mathbf{s}}_0 + \sum_{i=1}^{\overline{m}} \overline{p}_i \, \overline{\mathbf{s}}_i \right) \tag{9}$$

where (i_x, i_y, i_z), (j_x, j_y, j_z), and the 3D shape parameters $\overline{p}_i$ vary over their allowed values. The projection matrix can be expressed as the sum of 6 matrices:

$$\begin{pmatrix} i_x & i_y & i_z \\ j_x & j_y & j_z \end{pmatrix} = i_x \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + i_y \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} + i_z \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} + j_x \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} + j_y \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} + j_z \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{10}$$

Equation (9) is therefore a linear combination of:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \overline{\mathbf{s}}_i, \quad \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \overline{\mathbf{s}}_i, \quad \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \cdot \overline{\mathbf{s}}_i, \quad \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \cdot \overline{\mathbf{s}}_i, \quad \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \cdot \overline{\mathbf{s}}_i, \quad \text{and} \quad \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \overline{\mathbf{s}}_i \tag{11}$$

for $i = 0, 1, \ldots, \overline{m}$. The shape variation of the 3D model can therefore be represented by the following 2D shape vectors:

$$\mathbf{s}_{6*i+1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \overline{\mathbf{s}}_i, \quad \mathbf{s}_{6*i+2} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \overline{\mathbf{s}}_i,$$

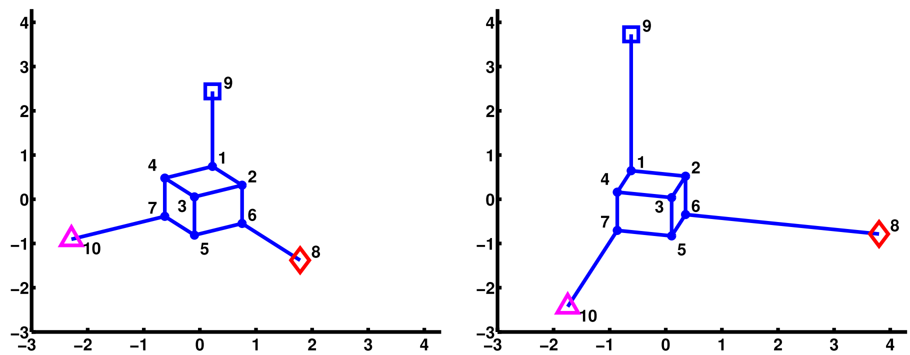

Figure 1. A scene consisting of a static cube and 3 points moving along fixed directions. (a) The base configuration. (b) The cube viewed from a different direction with the 3 points (8, 9, 10) moved.

$$\mathbf{s}_{6*i+3} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \bar{\mathbf{s}}_i, \quad \mathbf{s}_{6*i+4} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \cdot \bar{\mathbf{s}}_i,$$
$$\mathbf{s}_{6*i+5} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \cdot \bar{\mathbf{s}}_i, \quad \mathbf{s}_{6*i+6} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \bar{\mathbf{s}}_i \tag{12}$$

for $i = 0, 1, \ldots, \overline{m}$. In total as many as $m = 6 \times (\overline{m} + 1)$ 2D shape vectors may be needed to model the same shape variation as the 3D model with only $\overline{m}$ 3D shape vectors. Although many more shape vectors may be needed, the main point is that 2D models can represent any geometric phenomena that 3D models can. (Note that although theoretically more than 6 times as many 2D shape vectors may be needed, in practice not that many are required. Typically 2-3 times as many are needed.)

$$\begin{pmatrix} -0.194 & -0.141 & -0.093 & -0.146 & -0.119 & -0.167 & -0.172 & -0.027 & 0.520 & 0.539 \\ 0.280 & 0.139 & 0.036 & 0.177 & -0.056 & 0.048 & 0.085 & -1.013 & 0.795 & -0.491 \end{pmatrix}. \tag{13}$$

3.2. *3D Realizability of 2D Model Instances*

If it takes 6 times as many parameters to represent a certain phenomenon with an 2D model than it does with the corresponding 3D model, the 2D model must be able to generate a large number of model instances that are impossible to generate with the 3D model. In effect, the 2D model has too much representational power. It describes the phenomenon in question, plus a variety of other shapes that are "physically unrealizable." If the parameters of the 2D model are chosen so that the orthogonality constraints on the corresponding 3D projection axes $\mathbf{i}$ and $\mathbf{j}$ do not hold, the 2D model instance is not realizable with the 3D model. An example is presented in Fig. 1.

The scene in Fig. 1 consists of a static cube (7 visible vertices) and 3 moving points (diamond, triangle, and square.) The 3 points can move along the three axes at the same, non-constant speed. The 3D shape of the scene $\bar{\mathbf{s}}$ is composed of a base shape $\bar{\mathbf{s}}_0$ and a single 3D shape vector $\bar{\mathbf{s}}_1$. The base shape $\bar{\mathbf{s}}_0$ correspond to the static cube and the initial locations of the three moving points. The 3D shape vector $\bar{\mathbf{s}}_1$ corresponds to the motion of the three moving points (8, 9, 10).

We randomly generated 60 sets of shape parameters $\overline{p}_1$ and camera projection matrices $\mathbf{P}$. We then used these to synthesize the 2D shapes of 60 3D model instances. We then computed the 2D shape model by performing PCA on the 60 2D shapes. The result consists of 12 shape vectors, confirming the result above that as many as $6 \times (\overline{m} + 1)$ 2D shape vectors might be required.

The resulting 2D shape model can generate shapes that are impossible to generate with the 3D model. One concrete example is the base 2D shape $\mathbf{s}_0$. In our experiment, $\mathbf{s}_0$ turns out to be:

To show that s_0 is not a 3D model instance, we first note that $\mathbf{s}_0$ can be *uniquely* decomposed into:

$$\mathbf{s}_0 = \begin{pmatrix} 0.053 & 0.026 & 0.048 \\ -0.141 & 0.091 & -0.106 \end{pmatrix} \bar{\mathbf{s}}_0 + \begin{pmatrix} 0.087 & 0.688 & 0.663 \\ -0.919 & 0.424 & -0.473 \end{pmatrix} \bar{\mathbf{s}}_1. \tag{14}$$

The projection matrices in this expression are not legitimate scaled orthographic matrices (i.e. composed of two equal length, orthogonal vectors). Therefore, $\mathbf{s}_0$ is not a valid 3D model instance.

One question that might be asked at this point is whether the redundancy in the 2D model is a problem in practice. In Section 5.5 we present quantitative fitting results that can be interpreted as showing that the redundancy in the 2D model leads to significantly more local minima (lower fitting robustness) when fitting 2D models, compared to the corresponding 3D model.

3.3. Summary

Although we have shown that a 2D model can be used to model a 3D face just as well as a 3D model, the parameterization is less compact and the 2D model will be able to generate "physically unrealizable" face instances; i.e. not all 2D face model instances are valid projections of 3D faces. As a result, the fitting of 2D face models is more prone to local minima and so is less robust. See Section 5.5 for empirical results that validate this claim. One additional benefit of 3D models is that the parameterization is more natural. The head pose is contained in the scaled orthographic camera matrix $\mathbf{P}$ rather than being confused with the non-rigid face motion in the 2D shape model.

4. Construction

Usually 2D face models are constructed from 2D image data and 3D face models from 3D range scans. In Section 4.1 we investigate how a 2D face model can be converted into a 3D model. In Section 4.2 we show how a 3D model can be projected into 2D and describe how such a procedure is useful for building 2D models that better model rotated faces.

4.1. Constructing a 3D Model from a 2D Model

Suppose we have a 2D AAM. A natural question to ask is whether it can be upgraded to a 3D model (in isolation). One approach might be to sample the model, generate 2D point sets, and apply a non-rigid structure-from-motion algorithm such as Xiao et al. (2004). The argument in Section 3.2, however, shows that doing this may well lead to sets of 2D points with no consistent 3D interpretation. Instead, we need to use 2D point sets from real images of the face as input because we know that these sets have a consistent 3D interpretation. In particular, suppose that we have a collection of images of one or more faces $I^t(\mathbf{u})$ for $t = 0, \ldots, N$, have fit the 2D AAM to each image, and that the 2D AAM shape parameters in image $I^t(\mathbf{u})$ are $\mathbf{p}^t = (p_1^t, \ldots, p_m^t)^T$. Using Eq. (2) we can then compute the 2D AAM shape vector $\mathbf{s}^t$:

$$\mathbf{s}^t = \begin{pmatrix} u_1^t & u_2^t & \ldots & u_n^t \\ v_1^t & v_2^t & \ldots & v_n^t \end{pmatrix} \tag{15}$$

for $t = 0, \ldots, N$. The images $I^t(\mathbf{u})$ could be the training images used to build the AAM, but could also be a larger set with a denser sampling of pose and expression variation.

An alternative approach would be to apply the same non-rigid structure-from-motion algorithm (Xiao et al., 2004) to the original data used to build the AAM. Besides the ability to include more data, the main advantage of first building the AAM and then fitting to a set of images is that this process removes noise. Non-rigid structure-from-motion can be a fairly noise-sensitive process. Because of the large number of parameters being estimated, it can also be prone to overfitting. In our experience, the noise removal in the AAM construction and fitting process, and the ability to use far more data, dramatically improve performance.

A variety of non-rigid structure-from-motion algorithms have been proposed to convert the feature points in Eq. (15) into a 3D shape model. Bregler et al. were perhaps the first to study the problem (Bregler et al., 2000). Later, Torresani et al. proposed an algorithm to simultaneously reconstruct the non-rigid 3D shape and camera projection matrices (Torresani et al., 2001). This trilinear optimization involves three types of unknowns, 3D shape vectors, shape parameters, and projection matrices. At each step, two of the unknowns are fixed and the third refined. Brand proposed a similar non-linear optimization method (Brand, 2001). Both of these methods only use the usual orthonormality constraints on the projection matrices (Tomasi and Kanade, 1992). In Xiao et al. (2004) we proved that only enforcing the orthonormality constraints in a linear algorithm is ambiguous and demonstrated that it can lead to an incorrect solution. We now outline how our linear algorithm (Xiao et al., 2004) can be used to compute 3D shape modes from the 2D AAM fitting data in Eq. (15). Any of the other algorithms mentioned above could be used instead, although they do not have the advantage of a linear solution. Since they are large non-linear optimizations, they require good initial estimates to converge.

We begin by stacking the 2D AAM fitting data from Eq. (15) into a measurement matrix:

$$W = \begin{pmatrix} u_1^0 & u_2^0 & \ldots & u_n^0 \\ v_1^0 & v_2^0 & \ldots & v_n^0 \\ \vdots & \vdots & \vdots & \vdots \\ u_1^N & u_2^N & \ldots & u_n^N \\ v_1^N & v_2^N & \ldots & v_n^N \end{pmatrix}. \tag{16}$$

If this data can be explained by $\overline{m}$ 3D shape modes, the matrix W can be factored into:

$$W = MB = \begin{pmatrix} \mathbf{P}^0 & p_1^0\mathbf{P}^0 & \ldots & p_{\overline{m}}^0\mathbf{P}^0 \\ \mathbf{P}^1 & p_1^1\mathbf{P}^1 & \ldots & p_{\overline{m}}^1\mathbf{P}^1 \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{P}^N & p_1^N\mathbf{P}^N & \ldots & p_{\overline{m}}^N\mathbf{P}^N \end{pmatrix} \begin{pmatrix} \bar{\mathbf{s}}_0 \\ \vdots \\ \bar{\mathbf{s}}_{\overline{m}} \end{pmatrix} \tag{17}$$

where each $\mathbf{P}^i$, $i = 0, \ldots, N$ is a 2×3 projection matrix (see Eq. (7) for a definition), M is a $2(N+1) \times 3(\overline{m}+1)$ scaled projection matrix, and B is a $3(\overline{m}+1) \times n$ shape matrix (setting the number of 3D shape model vertices $\overline{n}$ equal to the number of AAM vertices n.) Since $\overline{m}$ is the number of 3D shape vectors, it is usually small and the rank of W is at most $3(\overline{m}+1)$.

We perform a Singular Value Decomposition (SVD) on W and factorize it into the product of a $2(N+1) \times 3(\overline{m}+1)$ matrix $\tilde{M}$ and a $3(\overline{m}+1) \times n$ matrix $\tilde{B}$. This decomposition is not unique. It is only determined up to a linear transformation. Any non-singular $3(\overline{m}+1) \times 3(\overline{m}+1)$ matrix G and its inverse G^{-1} could be inserted between $\tilde{M}$ and $\tilde{B}$ and their product would still equal W. The scaled projection matrix M and the shape vector matrix B are then given by:

$$M = \tilde{M} \cdot G, \qquad B = G^{-1} \cdot \tilde{B} \tag{18}$$

where G is a corrective matrix. The first step in solving for G is to impose the usual orthonormality constraints matrices (Tomasi and Kanade, 1992). If we represent G as the columns $G = (g_0 \dots g_{\overline{m}})$ where g_k, $k = 0, \dots, \overline{m}$ are $3(\overline{m}+1) \times 3$ matrices, the left half of Eq. (18) takes the form:

$$\tilde{M} \cdot g_k = \begin{pmatrix} p_k^0 \mathbf{P}^0 \\ p_k^1 \mathbf{P}^1 \\ \vdots \\ p_k^N \mathbf{P}^N \end{pmatrix}, \quad k = 0, \dots, \overline{m}. \tag{19}$$

Denote $g_k g_k^T$ by Q_k. Using the orthogonality properties of $\mathbf{P}^i$ outlined in Section 2.2 we have:

$$\tilde{M}_{2i+1} Q_k \tilde{M}_{2i+1}^T = \tilde{M}_{2i+2} Q_k \tilde{M}_{2i+2}^T \tag{20}$$

$$\tilde{M}_{2i+1} Q_k \tilde{M}_{2i+2}^T = 0 \tag{21}$$

for $i = 0, \dots, N$, and where $\tilde{M}_i$ is the ith row of $\tilde{M}$. Equations (20) and (21) provide $2(N+1)$ linear constraints on the symmetric matrix Q_k, which consists of $(9(\overline{m}+1)^2 + 3(\overline{m}+1))/2$ unknowns. In Xiao et al. (2004), however, we proved that even if $2(N+1) \geq (9(\overline{m}+1)^2 + 3(\overline{m}+1))/2$, these orthonormality constraints are not sufficient to determine Q_k.

Although the 3D shape modes span a unique shape space, the basis of the space is not unique. Any non-singular linear transformation of the shape vectors yields a new set of shape vectors. We therefore need an additional set of constraints. Without loss of generality, we can assume that the 3D shapes in the first $(\overline{m}+1)$ frames are independent[2] and treat them as the 3D shape basis. Equivalently, we can specify the shape parameters in the first $(\overline{m}+1)$ frames to be:

$$\overline{p}_i^i = \frac{1}{|\mathbf{P}^i|} \quad \text{for } i = 0, \dots, \overline{m}$$
$$\overline{p}_j^i = 0 \quad \text{for } i, j = 0, \dots, \overline{m},\ i \neq j \tag{22}$$

where $|\mathbf{P}^i| = i_x^i i_x^i + i_y^i i_y^i + i_z^i i_z^i$. See Eq. (7) in Section 2.2 for the definition of $\mathbf{P}$. Combining these constraints with Eq. (19) leads to another set of linear constraints on Q_k:

$$\tilde{M}_{2i+1} Q_k \tilde{M}_{2j+1}^T = \begin{cases} 1 & \text{if } i = j = k \\ 0 & \text{if } i \neq k \end{cases} \tag{23}$$

$$\tilde{M}_{2i+2} Q_k \tilde{M}_{2j+2}^T = \begin{cases} 1 & \text{if } i = j = k \\ 0 & \text{if } i \neq k \end{cases} \tag{24}$$

$$\tilde{M}_{2i+1} Q_k \tilde{M}_{2j+2}^T = 0 \quad \text{if } i \neq k \text{ or } i = j = k \tag{25}$$

$$\tilde{M}_{2i+2} Q_k \tilde{M}_{2j+1}^T = 0 \quad \text{if } i \neq k \text{ or } i = j = k \tag{26}$$

where $i = 0, \dots, \overline{m}$ and $j = 0, \dots, N$. Enforcing the (roughly) $4(\overline{m}+1)(N+1)$ linear constraints in Eqs. (23)–(26) and the $2(N+1)$ linear constraints in Eqs. (20) and (21) leads to a closed-form solution for Q_k. Since $Q_k = g_k g_k^T$ we can solve for g_k using SVD (Tomasi and Kanade, 1992) and then recover M and B using Eq. (18). However, because each g_k is estimated independently, the orthogonal Procrustes method must be used to place all of the estimates in a common coordinate frame (Xiao et al., 2006).

We illustrate the computation of a 3D shape model from a 2D shape model in Fig. 2. We first constructed a 2D AAM for 5 people using approximately 20 training images of each person. In Fig. 2(A) we include the AAM mean 2D shape $\mathbf{s}_0$ and the first 3 (of 8) 2D shape variation modes $\mathbf{s}_1$, $\mathbf{s}_2$, and $\mathbf{s}_3$. In Fig. 2(B) we include the AAM mean appearance A_0 and an illustration of the appearance variation. Instead of displaying the appearance modes themselves, we display the result of adding and subtracting the first 3 (of 40) modes to the mean appearance. For example, $+A_1$ denotes the addition of the first appearance mode to the mean appearance; i.e. $A_0 + A_1$. Next we use the 2D AAM to track short 180 frame videos of each of the 5 subjects to generate the input to the non-rigid structure-from-motion algorithm. One example frame for 4 of the 5 subjects is included in Fig. 2(C). The results of our 3D model construction algorithm are shown in Fig. 2(D). In particular, we display the mean 3D shape $\overline{\mathbf{s}}_0$ and the first 3 (of 4) 3D shape modes $\overline{\mathbf{s}}_1$, $\overline{\mathbf{s}}_2$, and $\overline{\mathbf{s}}_3$, with the 3D shape modes illustrated as arrows on the mean shape.

Similar results for a different model and dataset are included in Fig. 3. In this case, the 2D AAM is constructed from 20–30 training images of each of 9 subjects, has 9 2D shape modes, and 59 2D appearance modes. We use the 2D AAM to track 150 frame videos for 9

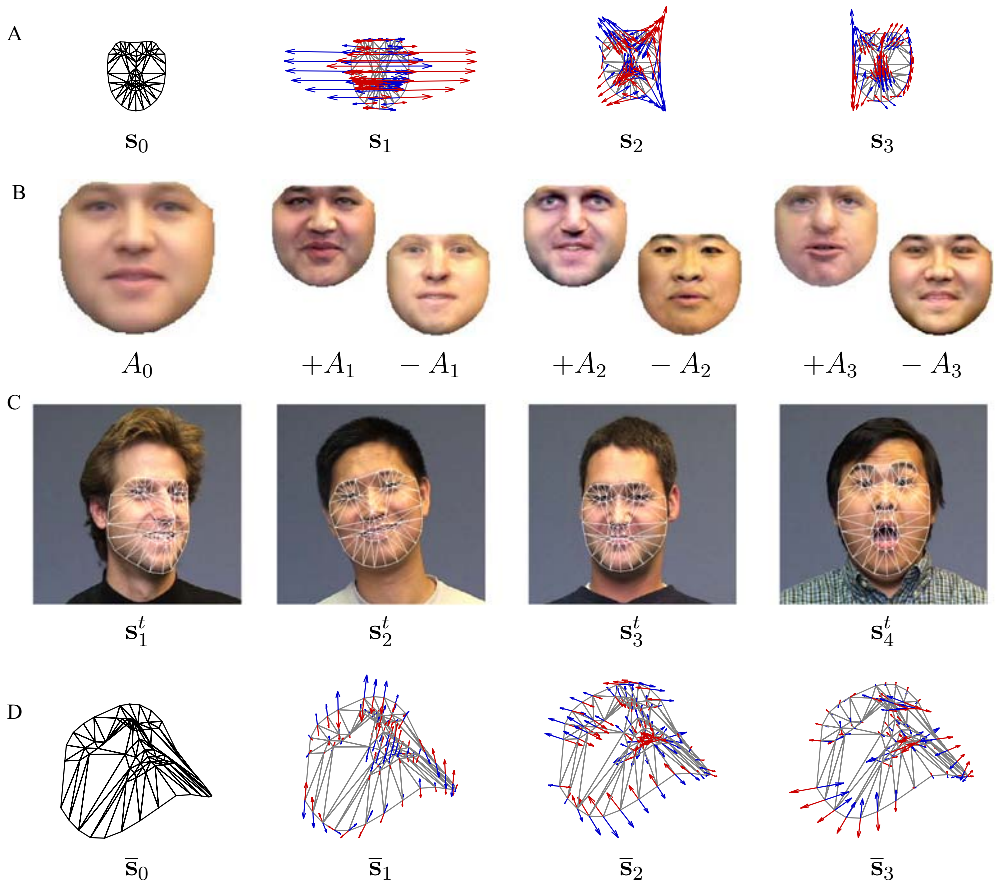

Figure 2. Model and Dataset 1: Constructing a 3D model from 2D data for a 5 person model. (A) The 2D AAM shape variation. (B) The 2D AAM appearance variation. (C) Example frames of the AAM being used to track 4 of the 5 subjects, the input to our algorithm. (D) The output of our algorithm, the mean 3D shape $\bar{\mathbf{s}}_0$ and the first 3 (of 4) 3D shape modes $\bar{\mathbf{s}}_1$, $\bar{\mathbf{s}}_2$, and $\bar{\mathbf{s}}_3$.

subjects. The output of our algorithm consists of a 3D model with 4 3D shape modes, of which we show the mean 3D shape $\bar{\mathbf{s}}_0$ and the first 3 modes $\bar{\mathbf{s}}_1$, $\bar{\mathbf{s}}_2$, and $\bar{\mathbf{s}}_3$ in Fig. 3(D).

4.2. *Constructing a 2D Model from a 3D Model*

Constructing a 2D model from a 3D model is easier than the other way around. We actually already did this in Section 3.2 when we were searching for a "physically unrealizable" 2D model instance. We simply take the 3D model, generate a large number of model instances (which are 2D shape vectors), and then construct the 2D model in the usual manner. The 3D model parameters and scaled orthographic projection matrix parameters can either be sampled systematically or randomly. In practice, we use a systematic sampling of the 2D (yaw/pitch) pose space and non-rigid shape space. (Scale, roll, and translation do not need to be sampled as they are modeled by the 2D similarity transformation.) In Fig. 4 we include 15 (of 465) example 3D model instances generated from the 4-dimensional 3D shape model in Fig. 2(D). These 465 training examples were then used to re-generate the 8-dimensional 2D shape model in Fig. 2(A). The result is a new 2D shape model with 9 modes.

A natural question, however, is: "why would you ever want to convert a 3D model into a 2D model?" One possibility is when working with either a 2D or a 2D + 3D model (Xiao et al., 2004) (to obtain real-time fitting speed) and trying to build models that can model the face across large pose variation. Although algorithms have been proposed to build 2D AAMs in the presence of occlusion caused by large pose variation in the training set (Gross et al., 2004), another way to build a 2D AAM than can model large pose variation is to: (1) build a normal 2D model using training data with no occlusion and so limited pose variation, (2) convert the 2D model into a 3D model using the approach in Section 4.1, and finally (3) convert the resulting 3D model back into a 2D model using an extending sampling of the camera

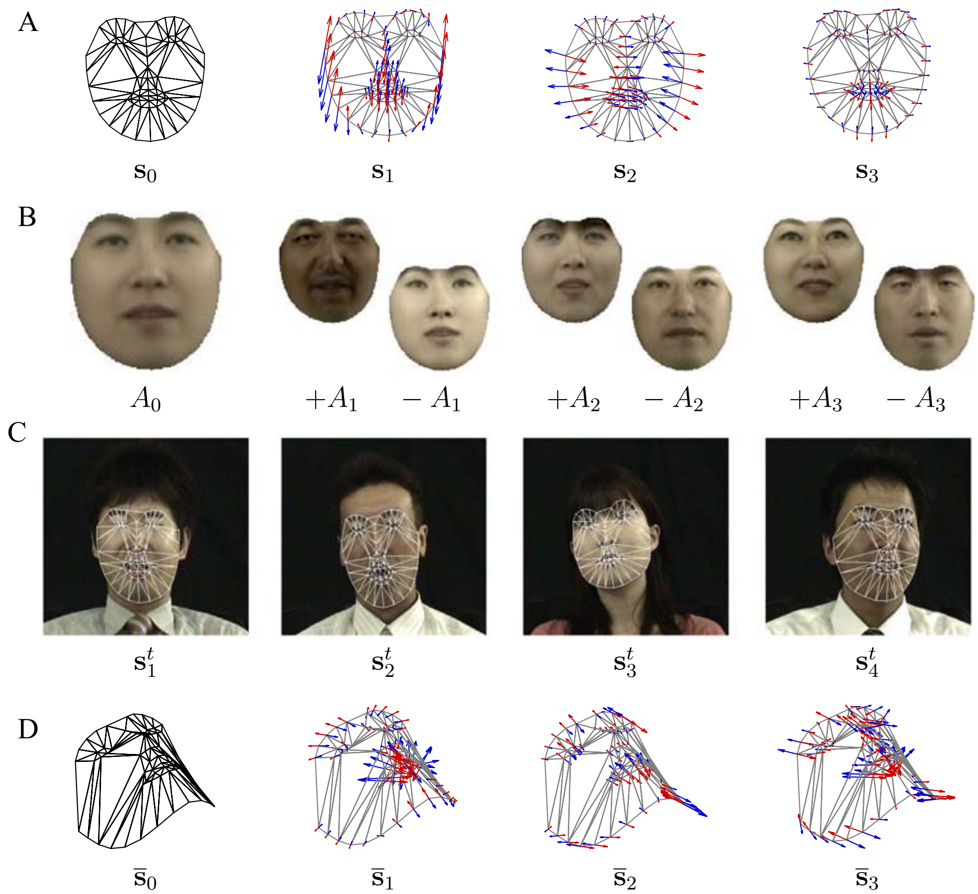

Figure 3. Model and Dataset 2: Constructing a 3D model from 2D data. (A) The 2D AAM shape variation. (B) The 2D AAM appearance variation. (C) Example frames of the AAM being used to track 4 of the subjects, the input to our algorithm. (D) The output of our algorithm, the mean 3D shape $\bar{\mathbf{s}}_0$ and the first 3 (of 4) 3D shape modes $\bar{\mathbf{s}}_1$, $\bar{\mathbf{s}}_2$, and $\bar{\mathbf{s}}_3$.

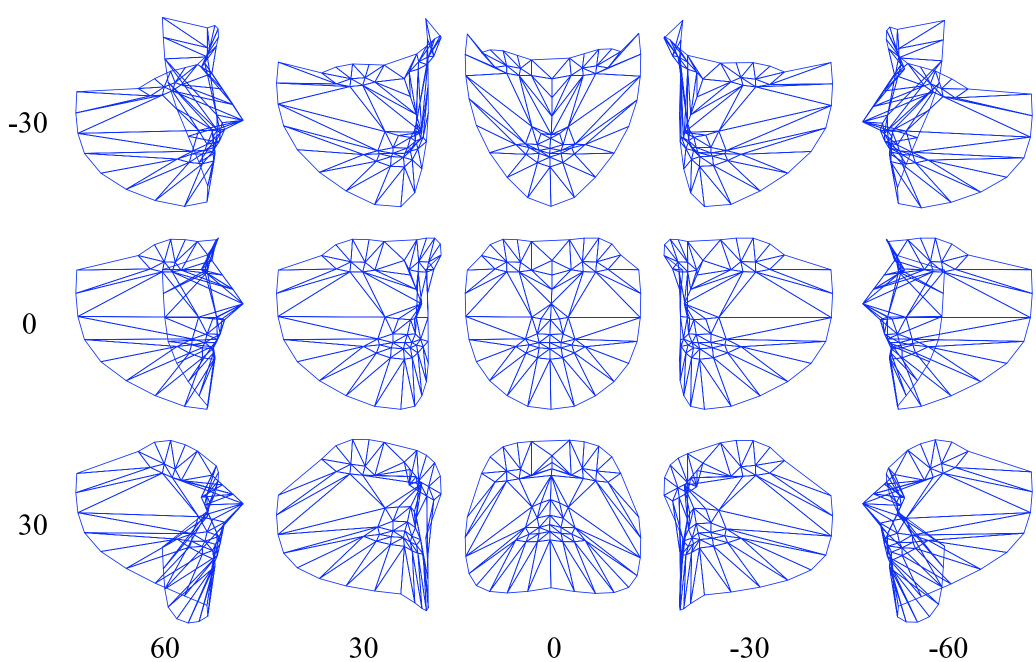

Figure 4. 3D model instances generated using the 3D model in Fig. 2(D). The pose is sampled systematically and the non-rigid shape randomly. These model instances are a subset of a set of 465 used to re-generate the 2D shape model in Fig. 2(A), and which is evaluated in Fig. 5.

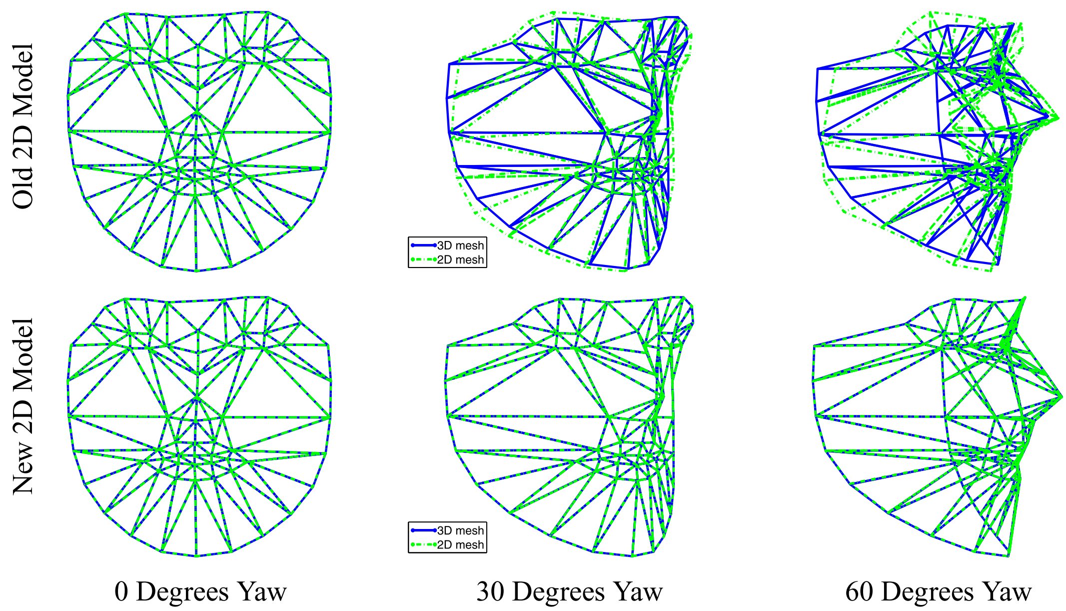

Figure 5. An example of 2D model construction from a 3D model. We display 3D model instances (solid blue line) and the closest 2D model instance (dashed green line) for 3 different head poses. We display the closest 2D model instances for the original 2D model (top row) and the new 2D model computed from the 3D model (bottom row). The results show that the new 2D model computed from the 3D model is better able to model faces at high degrees of pose variation.

matrix to model increased pose variation and self occlusion. Some results of performing this three-step procedure are included in Fig. 5. In particular, for 3 different head poses (0 degrees yaw, 30 degrees yaw, and 60 degrees yaw) we display the 3D model instance (solid blue line) and the closest 2D model instance (dashed green line) for both the original 2D model and the new 2D model computed from the 3D model. The results show that the new 2D model computed from the 3D model is better able to model faces at high degrees of pose variation than the original 2D model was able to.

4.3. Summary

We have shown how 3D models can be constructed from 2D models, and vica versa. In particular, 3D models such as 3D Morphable Models (Blanz and Vetter, 1999; Blanz et al., 2003) and 2D + 3D Active Appearance Models (Xiao et al., 2004) can be constructed from images collected with the same cameras that will be used in the final application. The training data is therefore more closely matched to the application data, a property that generally results in better (fitting and application) performance. On the other hand, one residual advantage of building 3D models from 3D range scans is that it is currently easier to build dense 3D models from range scans (although the non-rigid structure-from-motion algorithms described in this section could obviously operate on dense sets of features.) Techniques have been proposed for computing dense 3D models from 2D data, however, currently the models constructed in this manner do not match the quality of 3D models constructed from 3D range scans.

5. Fitting

In Baker and Matthews (2001) and Matthews and Baker (2004) we proposed a real-time algorithm for fitting 2D Active Appearance Models. On the other hand, Romdhani and Vetter recently presented an algorithm for fitting 3D Morphable Models that operates at around 30 seconds per frame (Romdhani and Vetter, 2003); i.e. thousands of times slower. In this section, we investigate the relative fitting speed of 2D and 3D models. We begin in Section 5.1 with a brief review of our real-time algorithm for fitting 2D AAMs. This algorithm is based on the inverse compositional image alignment algorithm (Baker and Matthews, 2004). In Section 5.2 we present a brief argument of why the inverse compositional algorithm does not generalize naturally to 3D shape models (Baker et al., 2004). In Section 5.3, we propose a work around to this problem, a real-time algorithm for fitting a 3D model that operates by fitting the 3D model indirectly through a 2D model (Xiao et al., 2004). After a qualitative evaluation of the 3D fitting algorithm in Section 5.4, we present a quantitative comparison of the robustness, rate of convergence, and computational cost of the 2D and 3D fitting algorithms in Sections 5.5–5.6.

5.1. *Efficient 2D Model Fitting*

The goal of fitting a 2D AAM (Matthews and Baker, 2004) is to minimize:

$$\sum_{\mathbf{u}\in\mathbf{s}_0}\left[A_0(\mathbf{u})+\sum_{i=1}^{l}\lambda_i A_i(\mathbf{u})-I(\mathbf{N}(\mathbf{W}(\mathbf{u};\mathbf{p});\mathbf{q}))\right]^2 \quad (27)$$

simultaneously with respect to the shape $\mathbf{p}$, appearance $\boldsymbol{\lambda}$, and normalization $\mathbf{q}$ parameters. The algorithm in Matthews and Baker (2004) uses the "Project Out" algorithm (Hager and Belhumeur, 1998) (described in more detail with an empirical evaluation that highlights some limitations of the algorithm in Baker et al. (2003)) to break the optimization into two steps. We first optimize:

$$\|A_0(\mathbf{u})-I(\mathbf{N}(\mathbf{W}(\mathbf{u};\mathbf{p});\mathbf{q}))\|^2_{\mathrm{span}(A_i)^\perp} \quad (28)$$

with respect to $\mathbf{p}$ and $\mathbf{q}$, where $\|\cdot\|^2_{\mathrm{span}(A_i)^\perp}$ denotes the square of the L2 norm of the vector projected into orthogonal complement of the linear subspace spanned by the vectors $A_1,\ldots,A_l$. Afterwards, we solve for the appearance parameters using the linear closed-form solution:

$$\lambda_i = -\sum_{\mathbf{u}\in\mathbf{s}_0} A_i(\mathbf{u})\cdot[A_0(\mathbf{u})-I(\mathbf{N}(\mathbf{W}(\mathbf{u};\mathbf{p});\mathbf{q}))] \quad (29)$$

where the shape $\mathbf{p}$ and normalization $\mathbf{q}$ parameters are the result of the previous optimization. Equation (29) assumes that the appearance vectors $A_i(\mathbf{u})$ have been orthonormalized.

In Matthews and Baker (2004) we showed how to use the inverse compositional algorithm (Baker and Matthews, 2004) to optimize the expression in Eq. (28) with respect to the shape parameters $\mathbf{p}$ and the normalization parameters $\mathbf{q}$. The main technical contribution allowing the use of the inverse compositional algorithm is a proof that approximately minimizing:

$$\|A_0(\mathbf{u})-I(\mathbf{N}(\mathbf{W}(\mathbf{u};\mathbf{p}+\Delta\mathbf{p});\mathbf{q}+\Delta\mathbf{q}))\|^2_{\mathrm{span}(A_i)^\perp} \quad (30)$$

with respect to $\Delta\mathbf{p}$ and $\Delta\mathbf{q}$, and then updating $\mathbf{p}\leftarrow\mathbf{p}+\Delta\mathbf{p}$ and $\mathbf{q}\leftarrow\mathbf{q}+\Delta\mathbf{q}$, is the same to a first order approximation as approximately minimizing:

$$\|A_0(\mathbf{N}(\mathbf{W}(\mathbf{u};\Delta\mathbf{p});\Delta\mathbf{q}))-I(\mathbf{N}(\mathbf{W}(\mathbf{u};\mathbf{p};\mathbf{q}))\|^2_{\mathrm{span}(A_i)^\perp} \quad (31)$$

with respect to $\Delta\mathbf{p}$ and $\Delta\mathbf{q}$, and then updating the warp using the inverse compositional update:

$$\mathbf{N}(\mathbf{W}(\mathbf{u};\mathbf{p});\mathbf{q})\leftarrow\mathbf{N}(\mathbf{W}(\mathbf{u};\mathbf{p});\mathbf{q})\circ\mathbf{N}(\mathbf{W}(\mathbf{u};\Delta\mathbf{p});\Delta\mathbf{q})^{-1}. \quad (32)$$

The algorithm itself then consists of iterating two main steps. The first step computes:

$$\begin{pmatrix}\Delta\mathbf{p}\\ \Delta\mathbf{q}\end{pmatrix} = -H_{2\mathrm{D}}^{-1}\begin{pmatrix}\Delta\mathbf{p}_{\mathrm{SD}}\\ \Delta\mathbf{q}_{\mathrm{SD}}\end{pmatrix} \quad (33)$$

where the steepest descent parameter updates $\binom{\Delta\mathbf{p}_{\mathrm{SD}}}{\Delta\mathbf{q}_{\mathrm{SD}}}$ are:

$$\begin{pmatrix}\Delta\mathbf{p}_{\mathrm{SD}}\\ \Delta\mathbf{q}_{\mathrm{SD}}\end{pmatrix}=\sum_{\mathbf{u}\in\mathbf{s}_0}[\mathbf{SD}_{2\mathrm{D}}(\mathbf{u})]^T[A_0(\mathbf{u})-I(\mathbf{N}(\mathbf{W}(\mathbf{u};\mathbf{p});\mathbf{q}))]. \quad (34)$$

In this expression, the steepest descent images $\mathbf{SD}_{2\mathrm{D}}(\mathbf{u})$ are the concatenation of:

$$\mathbf{SD}_{2\mathrm{D}}(\mathbf{u}) = \left(\left[\nabla A_0\frac{\partial\mathbf{N}\circ\mathbf{W}}{\partial\mathbf{p}}\right]_{\mathrm{span}(A_i)^\perp}\left[\nabla A_0\frac{\partial\mathbf{N}\circ\mathbf{W}}{\partial\mathbf{q}}\right]_{\mathrm{span}(A_i)^\perp}\right) \quad (35)$$

one for the shape, and one for the normalization parameters. The Hessian matrix $H_{2\mathrm{D}}$ is:

$$H_{2\mathrm{D}} = \sum_{\mathbf{u}\in\mathbf{s}_0}[\mathbf{SD}_{2\mathrm{D}}(\mathbf{u})]^T\,\mathbf{SD}_{2\mathrm{D}}(\mathbf{u}). \quad (36)$$

The second step consists of updating the warp with the inverse compositional update in Eq. (32). The inverse compositional algorithm obtains its efficiency from the fact that Eqs. (35) and (36) can be pre-computed for any given model. The Hessian can even be pre-multiplied by the steepest descent images. (This pre-computation is omitted in the presentation above to allow easier explanation of the extension of the algorithm in Section 5.3 below.) All that is left is: (1) a piecewise affine warp to generate $I(\mathbf{N}(\mathbf{W}(\mathbf{u};\mathbf{p});\mathbf{q}))$, (2) an image subtraction to generate $A_0(\mathbf{u})-I(\mathbf{N}(\mathbf{W}(\mathbf{u};\mathbf{p});\mathbf{q}))$, (3) $m+4$ image dot products where there are m shape and 4 similarity normalization parameters, and (4) the inverse compositional update to the warp. Although computing the inverse compositional warp update is involved (Matthews and Baker, 2004), it is not computationally demanding because it only requires operations on the mesh, rather than on the images. All four of these steps can be implemented in real-time (Matthews and Baker, 2004).

5.2. *Invalidity of the 3D Inverse Compositional Algorithm*

At this point it is natural to ask whether the inverse compositional algorithm generalizes to 3D models. Unfortunately, the *direct generalization* violates one of the key assumptions made in the proof of correctness (Baker et al.,

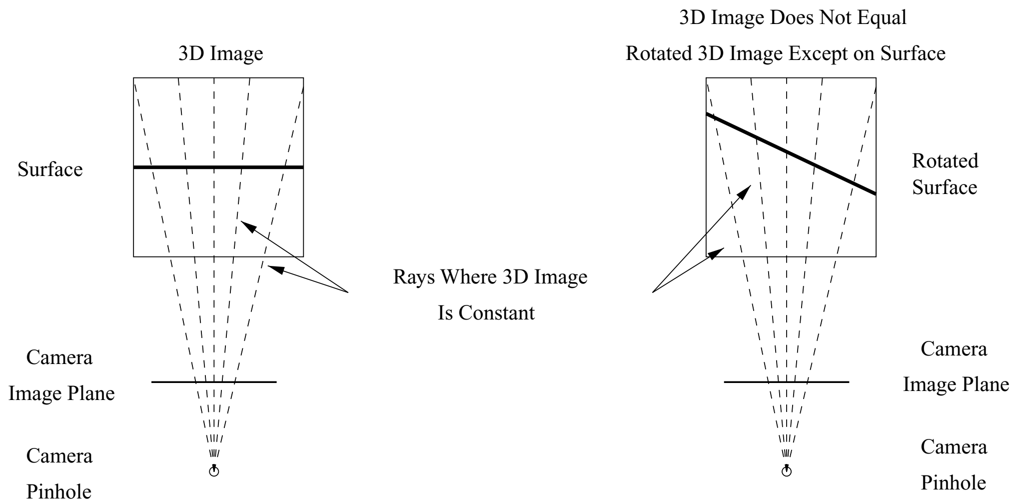

Figure 6. An illustrative example to help explain the invalidity of the inverse compositional algorithm when directly applied to a 3D model. Left: A 2D surface imaged by a camera generates an equivalent 3D image where the ray through each pixel has a constant intensity. Right: If the 2D surface is rotated (by a warp $\overline{\mathbf{W}}(\mathbf{x}; \overline{\mathbf{p}})$, say) the corresponding 3D images agree on the surface after compensating for the rotation, but they disagree away from the surface. Their 3D gradients at the surface are therefore not interchangeable.

2004). In particular, when the 3D-to-2D projection matrix $\mathbf{P}$ is added to the image part of the model fitting goal, the inverse compositional algorithm is not applicable. Note, however, that at least two "work-arounds" have been proposed to avoid this problem: (1) the 2D + 3D constrained fitting algorithm described in Section 5.3 and (2) the implicit 2D parameterization of Romdhani and Vetter (2003) described in Section 5.8.

In the proof of equivalence of the forwards and inverse compositional algorithms in Baker and Matthews (2004), an assumption is made that:

$$\|A_0(\mathbf{N}(\mathbf{W}(\mathbf{u}; \Delta\mathbf{p}); \Delta\mathbf{q})) - I(\mathbf{N}(\mathbf{W}(\mathbf{u}; \mathbf{p}; \mathbf{q}))\|^2_{\text{span}(A_i)^\perp} = O(\Delta\mathbf{p}, \Delta\mathbf{q}). \tag{37}$$

Intuitively, in order to replace gradients of I in the forwards compositional algorithm with gradients of A_0 in the inverse compositional algorithm, I and A_0 must be approximately the same, after compensating for the geometric distortion modeled by the warp $\mathbf{N} \circ \mathbf{W}$, and working in the subspace $\text{span}(A_i)^\perp$ to compensate for the appearance variation. The assumption in Eq. (37) is reasonable in the 2D case. Informally, it says that the model matches the image reasonably well at the correct solution, the implicit assumption made by the model fitting process anyway. If we generalize Eq. (37) to 3D we end up with:

$$\left\|A_0(\mathbf{P}(\overline{\mathbf{W}}(\mathbf{x}; \Delta\overline{\mathbf{p}}))) - I(\mathbf{P}(\overline{\mathbf{W}}(\mathbf{x}; \overline{\mathbf{p}})))\right\|^2_{\text{span}(A_i)^\perp} = O(\Delta\overline{\mathbf{p}}) \tag{38}$$

where $\overline{\mathbf{W}}(\mathbf{x}; \overline{\mathbf{p}})$ is a 3D warp with 3D shape parameters $\overline{\mathbf{p}}$ using 3D coordinates $\mathbf{x}$ rather than 2D coordinates $\mathbf{u}$. Note that because the 3D versions of the algorithms use the 3D gradients of $A_0(\mathbf{P}(\mathbf{x}))$ and $I(\mathbf{P}(\mathbf{x}))$, Eq. (38) must hold *both on the surface of the object and in an infinitesimal 3D neighborhood around the surface.* However, while it is reasonable to assume that Eq. (38) holds for $\mathbf{x}$ on the surface of the object, Eq. (38) does not hold (in general; e.g. for non-zero rotations) for points off the surface of the object.

The functions $I(\mathbf{P}(\mathbf{x}))$ and $A_0(\mathbf{P}(\mathbf{x}))$ can be thought of as implicitly defined 3D images; for each 3D point $\mathbf{x}$ a unique intensity $I(\mathbf{P}(\mathbf{x}))$ or $A_0(\mathbf{P}(\mathbf{x}))$ is defined. Essentially the image is projected out into 3D space. A simple illustrative example is given in Fig. 6. If the surface is rotated in 3D (say), without moving the camera, a different 3D image is created that is not simply a rotated version of the 3D image created by the unrotated object. The rotated 3D images will agree on the surface of the object, where they both correspond to the light irradiate by the object, but the directions of constant intensity depend on the direction to the camera center of projection. The gradients of the two 3D images therefore do not agree on the surface, even after compensating for the rotation. In the case of the inverse compositional algorithm, the first 3D image is $I(\mathbf{P}(\mathbf{x}))$ and the second rotated surface $A_0(\mathbf{P}(\mathbf{x}))$. The above argument says that the 3D gradient of $I(\mathbf{P}(\mathbf{x}))$ cannot be replaced by the 3D gradient of $A_0(\mathbf{P}(\mathbf{x}))$ and so the direct generalization of the inverse compositional algorithm is not correct. It is of course possible to compute the appropriate correction to the gradient, but this would need to be done online, depending on the current estimate of the 3D warp and so the Hessian (etc.) would need to be updated. This would eliminate the computational speedup obtained by the inverse computational algorithm.

5.3. Constrained 3D Fitting

One way to avoid the problem just described is to use both a 2D model and a 3D model. The 2D model is fit to the image subject to the constraints that the 2D model equals the projection of the 3D model. In essence, the 3D model is fit to the image[3] through the 2D model. The image gradients are 2D, so the inverse compositional algorithm can be used. Imposing the constraints between the 2D and 3D models can be performed without significantly reducing the speed of the algorithm, as we will now show. The result is a real-time 3D model fitting algorithm.

Suppose we have a 2D shape model defined by Eq. (2) and a 3D shape model defined by Eq. (5). Denote the result of mapping the 2D model into an image with the normalization $\mathbf{N}(\mathbf{u};\mathbf{q})$ by $\mathbf{N}(\mathbf{s};\mathbf{q}) = \mathbf{N}(\mathbf{s}_0 + \sum_{i=1}^{m} p_i\, \mathbf{s}_i;\mathbf{q})$. Similarly, denote the projection of the 3D model into the image by $\mathbf{P}(\overline{\mathbf{s}}_0 + \sum_{i=1}^{\overline{m}} \overline{p}_i\, \overline{\mathbf{s}}_i) + \left(\begin{smallmatrix} o_u & \dots & o_u \\ o_v & \dots & o_v \end{smallmatrix}\right)$. See Eq. (8) in Section 2.2 for a definition of the scaled orthographic image formation model being used in this paper. We can impose the constraints that the projection of the 3D shape model equals the normalized 2D shape model as soft constraints with a large weight K by modifying the fitting goal in Eq. (27) to:

$$\sum_{\mathbf{u}\in\mathbf{s}_0}\left[A_0(\mathbf{u}) + \sum_{i=1}^{l}\lambda_i A_i(\mathbf{u}) - I(\mathbf{N}(\mathbf{W}(\mathbf{u};\mathbf{p});\mathbf{q}))\right]^2 + K\cdot\left\|\mathbf{P}\left(\overline{\mathbf{s}}_0 + \sum_{i=1}^{\overline{m}}\overline{p}_i\,\overline{\mathbf{s}}_i\right) + \begin{pmatrix} o_u & \dots & o_u \\ o_v & \dots & o_v\end{pmatrix} - \mathbf{N}\left(\mathbf{s}_0 + \sum_{i=1}^{m} p_i\,\mathbf{s}_i;\mathbf{q}\right)\right\|^2. \quad (39)$$

The first term in Eq. (39) is the 2D fitting goal of Eq. (27). The second term imposes the constraint that the projection of the 3D model equals the normalized 2D shape. The optimization is performed simultaneously with respect to the 2D shape parameters $\mathbf{p}$, the 2D normalization parameters $\mathbf{q}$, the appearance parameters $\boldsymbol{\lambda}$, the 3D shape parameters $\overline{\mathbf{p}}$, the parameters of the scaled orthographic projection matrix $\mathbf{P}$ (see below for more details), and the offset to the origin $(o_u, o_v)^{\mathrm{T}}$. In the limit $K \to \infty$ the constraints become hard. In practice, a suitably large value for K results in the system being solved approximately as though the constraints are hard. Finally note that for the constraints in the second term of Eq. (39) to make sense, we need the argument in Section 3.1 to show that the projection of any 3D shape can also be generated by a 2D shape model. In a sense, Section 3.1 is a "proof of correctness" of this formulation of 3D model fitting.

As above, we use the "Project Out" algorithm (Hager and Belhumeur, 1998) and first optimize:

$$\|A_0(\mathbf{u}) - I(\mathbf{N}(\mathbf{W}(\mathbf{u};\mathbf{p});\mathbf{q}))\|^2_{\mathrm{span}(A_i)^{\perp}} + K\cdot\left\|\mathbf{P}\left(\overline{\mathbf{s}}_0 + \sum_{i=1}^{\overline{m}}\overline{p}_i\,\overline{\mathbf{s}}_i\right) + \begin{pmatrix} o_u & \dots & o_u \\ o_v & \dots & o_v\end{pmatrix} - \mathbf{N}\left(\mathbf{s}_0 + \sum_{i=1}^{m} p_i\,\mathbf{s}_i;\mathbf{q}\right)\right\|^2. \quad (40)$$

with respect to $\mathbf{p}$, $\mathbf{q}$, $\overline{\mathbf{p}}$, $\mathbf{P}$, and $(o_u, o_v)^{\mathrm{T}}$. Afterwards, we solve for $\boldsymbol{\lambda}$ using Eq. (29).

In Baker et al. (2004) we showed how to extend the inverse compositional algorithm to allow constraints on the warp parameters. Equation (40) is of the form:

$$G(\mathbf{p};\mathbf{q}) + K\cdot\sum_{t=u,v}\sum_{i=1}^{n} F_{t_i}^2(\mathbf{p};\mathbf{q};\overline{\mathbf{p}};\mathbf{P};o_u;o_v) \quad (41)$$

where $G(\mathbf{p};\mathbf{q}) = \|A_0(\mathbf{u}) - I(\mathbf{N}(\mathbf{W}(\mathbf{u};\mathbf{p});\mathbf{q}))\|^2_{\mathrm{span}(A_i)^{\perp}}$, $F_{u_i}^2(\mathbf{p};\mathbf{q};\overline{\mathbf{p}};\mathbf{P};o_u;o_v)$ is the component of the L2 norm in the second term corresponding to vertex u_i, and $F_{v_i}^2(\mathbf{p};\mathbf{q};\overline{\mathbf{p}}\mathbf{P};o_u;o_v)$ is the component corresponding to vertex v_i. In the usual forwards additive (Lucas-Kanade) algorithm (Baker and Matthews, 2004), we solve for updates the the parameters $\Delta\mathbf{p}$ and $\Delta\mathbf{q}$ that approximately minimize $G(\mathbf{p}+\Delta\mathbf{p};\mathbf{q}+\Delta\mathbf{q})$. With the inverse compositional algorithm, however, (see Baker et al. 2004) we effectively solve for updates to the parameters that approximately minimize:

$$G(\mathbf{p} + \mathbf{J_p}\Delta\mathbf{p};\mathbf{q} + \mathbf{J_q}\Delta\mathbf{q}) \quad (42)$$

where to a first order approximation:

$$\mathbf{N}(\mathbf{W}(\mathbf{u};\mathbf{p}+\mathbf{J_p}\Delta\mathbf{p});\mathbf{q}+\mathbf{J_q}\Delta\mathbf{q}) \approx \mathbf{N}(\mathbf{W}(\mathbf{u};\mathbf{p});\mathbf{q})\circ\mathbf{N}(\mathbf{W}(\mathbf{u};\Delta\mathbf{p});\Delta\mathbf{q})^{-1} \quad (43)$$

In this expression $\mathbf{J_p}$ is an $m\times m$ matrix and $\mathbf{J_q}$ is an 4×4 matrix. The meaning of $\mathbf{J_p}$ is, for example, given a small change $\Delta\mathbf{p}$ to the warp parameters, how do the parameters of the right hand side of Eq. (43) change. Both $\mathbf{J_p}$ and $\mathbf{J_q}$ depend on $\mathbf{p}$ and $\mathbf{q}$. We describe how to compute them (in combination with $\frac{\partial F_{t_i}}{\partial \mathbf{p}}$ and $\frac{\partial F_{t_i}}{\partial \mathbf{q}}$) in Appendix A. We experimentally verified that failure to take into account $\mathbf{J_p}$ and $\mathbf{J_q}$ in the inverse compositional update can cause the algorithm to fail, particularly for large pose variation. The 2D shape and the projection of the 3D shape diverge.

The Gauss-Newton inverse compositional algorithm to minimize Eq. (41) then consists of iteratively computing:

$$\begin{pmatrix} \Delta\mathbf{p} \\ \Delta\mathbf{q} \\ \Delta\overline{\mathbf{p}} \\ \Delta\mathbf{P} \\ \Delta o_u \\ \Delta o_v \end{pmatrix} = -H_{3\mathrm{D}}^{-1} \left(\begin{pmatrix} \Delta\mathbf{p}_{\mathrm{SD}} \\ \Delta\mathbf{q}_{\mathrm{SD}} \\ \mathbf{0} \\ \mathbf{0} \\ 0 \\ 0 \end{pmatrix} + K \cdot \sum_{t=u,v} \sum_{i=1}^{n} [\mathbf{SD}_{F_{t_i}}]^{\mathrm{T}} F_{t_i}(\mathbf{p};\mathbf{q};\overline{\mathbf{p}};\mathbf{P};o_u;o_v) \right) \tag{44}$$

where:

$$\mathbf{SD}_{F_{t_i}} = \left(\frac{\partial F_{t_i}}{\partial \mathbf{p}} \mathbf{J}_{\mathbf{p}} \; \frac{\partial F_{t_i}}{\partial \mathbf{q}} \mathbf{J}_{\mathbf{q}} \; \frac{\partial F_{t_i}}{\partial \overline{\mathbf{p}}} \; \frac{\partial F_{t_i}}{\partial \mathbf{P}} \; \frac{\partial F_{t_i}}{\partial o_u} \; \frac{\partial F_{t_i}}{\partial o_v} \right) \tag{45}$$

and:

$$H_{3\mathrm{D}} = \begin{pmatrix} H_{2\mathrm{D}} & \mathbf{0} & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} + K \cdot \sum_{t=u,v} \sum_{i=1}^{n} [\mathbf{SD}_{F_{t_i}}]^T [\mathbf{SD}_{F_{t_i}}]. \tag{46}$$

After the parameter updates have been computed using Eq. (44), the warp is updated using Eq. (32), the 3D shape parameters are updated $\overline{\mathbf{p}} \leftarrow \overline{\mathbf{p}} + \Delta\overline{\mathbf{p}}$, and the origin offsets are updated $o_u \leftarrow o_u + \Delta o_u$ and $o_v \leftarrow o_v + \Delta o_v$. The update to the camera matrix $\mathbf{P}$ is slightly more involved because there are only really 4 degrees of freedom. See Appendix A for the details.

Although Eqs. (44–46) may appear fairly complex, the problem is now $m + 4 + \overline{m} + 6$ dimensional rather than $m + 4$ dimensional, and the only things that can be pre-computed are the 2D steepest descent images $\mathbf{SD}_{2\mathrm{D}}(\mathbf{u})$ and the 2D Hessian $H_{2\mathrm{D}}$, implementing these equations does not take significantly more computation than implementing Eqs. (33) and (34). The reasons are: (1) the summations in Eqs. (44) and (46) are over $2n$ vertex u and v coordinates rather than over the thousands of pixels in the model $\mathbf{u} \in \mathbf{s}_0$, (2) as shown in Section 3, usually $\overline{m} \ll m$ and so the difference in dimensionality is not so great, and (3) this optimization is more constrained and so requires less iterations to converge (see experimental results below for more details.)

5.4. *Qualitative Evaluation*

In Fig. 7 we include 4 examples of the 3D algorithm fitting to a single input image, 2 for Dataset 1 Fig. 2), and 2 for Dataset 2 (Fig. 3). In the left column we display the initial configuration, in the middle column the results after 5 iterations, and in the right column the results after the algorithm has converged. In each case, we display the input image with the current estimate of the 3D shape mesh overlayed in white. The 3D shape is projected onto the image with the current camera matrix. In the top right of each image, we also include renderings of the 3D shape from two different viewpoints. In the top left of the image, we display estimates of the 3D pose extracted from the current estimate of the camera matrix **P**. We also display the current estimate of the 2D shape as blue dots. In the accompanying movie `fit.mpg` we include the full fitting movie for a few more challenging examples from Dataset 1. In this movie, we start the algorithm quite far from the correct solution and so a relatively large number of iterations are required.

In Fig. 8 we include 12 frames of the 3D algorithm being used to track faces. The results for Dataset 1 in the top 2 rows are examples tracking a subset of the 5 people used to build the model in Fig. 2. The results for Dataset 2 in the bottom 2 rows are examples tracking people who were not in the training data used to build the model in Fig. 3. A movie concatenating tracking results for all 5 people in Dataset 1 and the 6 people not in Dataset 2 is included in `track.mpg`.

The range of convergence of the 3D fitting algorithm is sufficient to allow initialization and re-initialization from a face detector. We run the 3D tracking algorithm in a foreground thread. In a background thread we first run a real-time face detector and then apply the 3D tracking algorithm to the result. The foreground tracking thread is re-initialized whenever the background thread gives a better fit than the foreground thread gave on the same frame. The accompanying movie `fd_initialize.mpg` includes a screen capture of the system, a few frames of which are included in Fig. 9. Overall, the tracking thread runs at 30 Hz on a dual 3.0 GHz PC, limited by the camera. See Section 5.6 for more details of the computational cost of the fitting algorithm.

5.5. *Quantitative Comparison: 2D Fitting vs. 3D Fitting*

Directly comparing the fitting performance of 2D AAMs (Cootes et al., 2001) and 3DMMs (Blanz and Vetter, 1999) is difficult because of the numerous small differences in the models and algorithms. See Section 2.3. On the other hand, comparing the 2D fitting algorithm (Matthews and Baker, 2004) with the 3D fitting algorithm in Section 5.3 is a more direct comparison of the inherent properties of 2D and 3D face models. Almost all of the code in the two algorithms is the same. The 3D algorithm just has a little extra code to deal with the second term in Eq. (39).

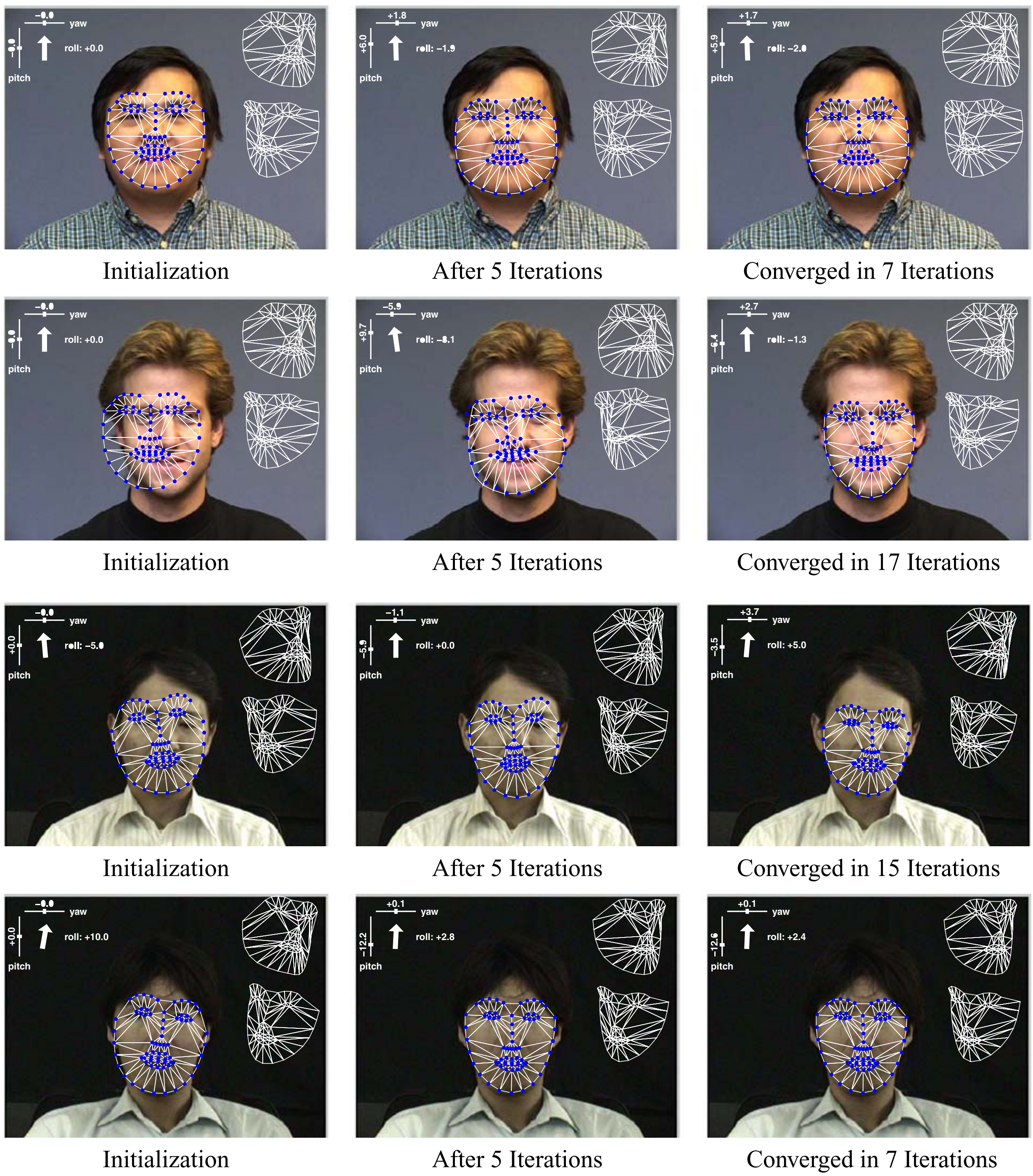

Figure 7. Examples of the 3D algorithm fitting to a single image, 2 for Dataset 1 in Fig. 2, and 2 for Dataset 2 in Fig. 3. We display the 3D shape estimate (white) projected onto the original image and also from a couple of other viewpoints in the top right of each image. In the top left of each image we display estimates of the 3D pose extracted from the current estimate of the camera matrix **P**. We also display the 2D shape estimate (blue dots.) See `fit.mpg` for a few more challenging examples from Dataset 1.

In Fig. 10 we present results comparing the fitting robustness and rate of convergence of the 2D fitting algorithm (Matthews and Baker, 2004) with the 3D fitting algorithm, following the protocol in Matthews and Baker (2004). In the top row we present results for Dataset 1 (Fig. 2) and in the bottom row for Dataset 2 (Fig. 3.) In the left column we present fitting robustness results. These results are generated by first generating a "convergence point" for each algorithm by tracking a set of videos. For Dataset 1 we use 828 frames from 5 videos

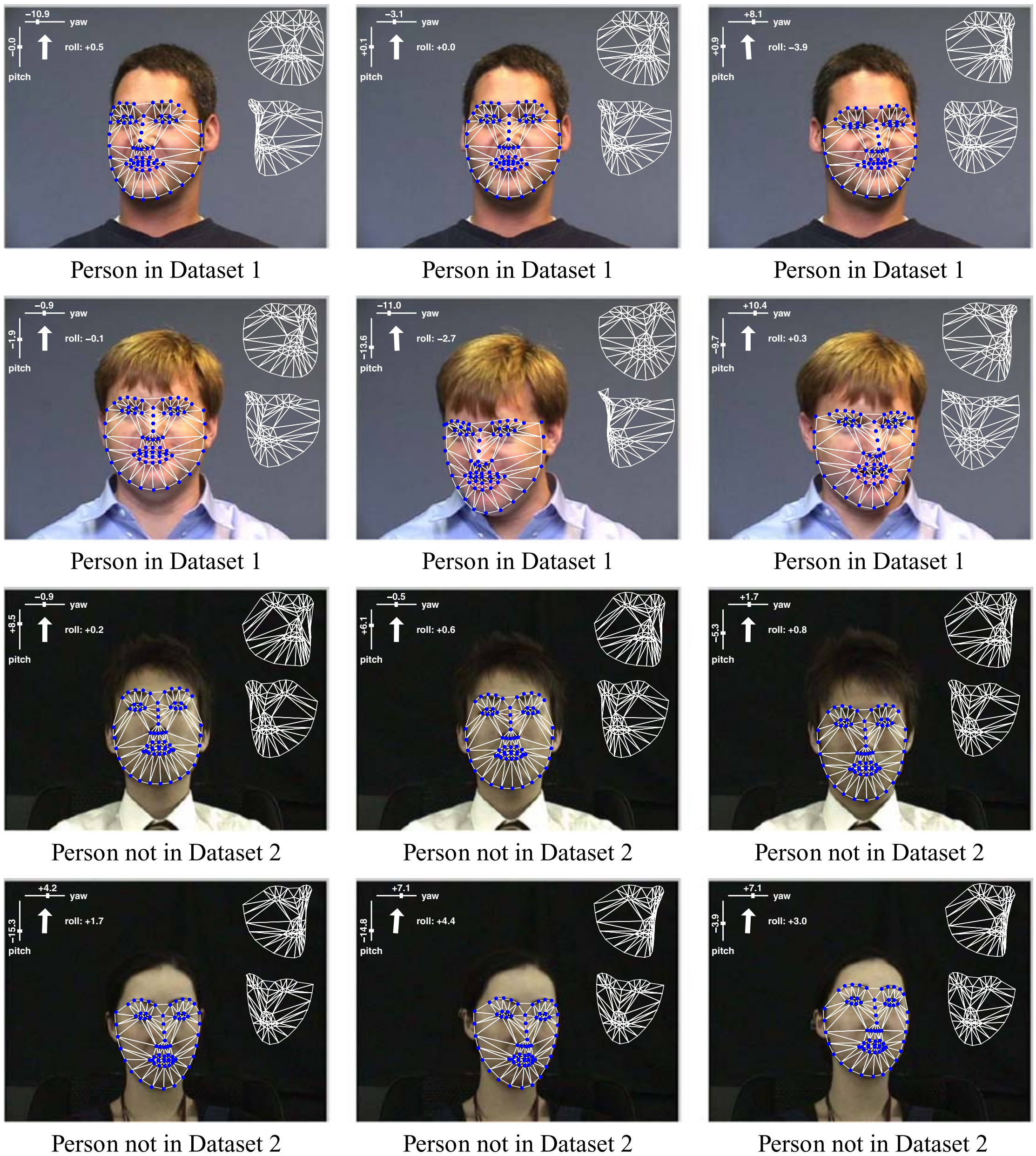

Figure 8. Example frames from the accompanying movie `track.mpg`. This movie concatenates tracking results for the 5 people in Dataset 1 and 6 people not used to build the model in Dataset 2.

of the 5 subjects in the training set. For Dataset 2 we use 900 frames from 6 videos of 6 subjects *not* in the training set. For each frame we then generate a number of trials by perturbing the 2D shape parameters from the convergence point. We follow the procedure in Matthews and Baker (2004) and weight the similarity and 2D shape parameters so that the algorithms would be approximately equally likely to convergence if perturbed independently in similarity and shape. We generate 20 trials for each of a set of larger and larger perturbations. We then run both fitting algorithms from the perturbed convergence point and determine whether they converge by comparing the final RMS 2D shape error with the convergence point. In particular, we used a threshold of 1.0 pixels to define convergence.

The rate of convergence results in the right column are generated by perturbing away from the convergence point in a similar manner. In this case we consider 2 different perturbation magnitudes, which result in the 2 curves for each algorithm. For each set of trials, we compute the average RMS pixel error after each iteration for all of the trials that end up converging to within 1.0 pixels. We then

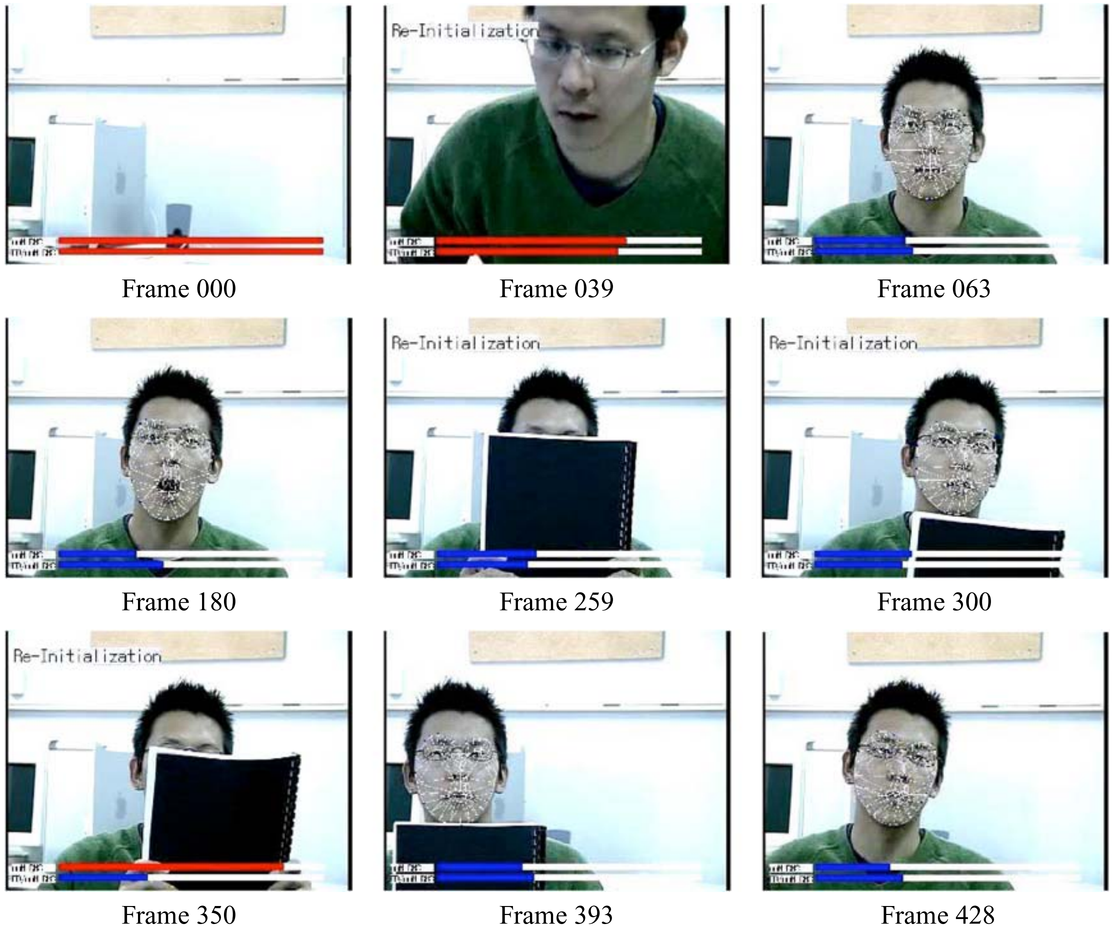

Figure 9. A few frames from the accompanying movie `fdinitialize.mpg` illustrating our integrated real-time 3D tracking system, with automatic initialization and re-initialization using a face detector. The subject enters the field of view of the camera. The face detector finds their face and initializes the tracker. When the person occludes their face temporally, the tracker is re-initialized after the face becomes visible again. Overall, the tracking thread in the system runs at 30 Hz on a dual 3.0 GHz PC, limited by the camera.

plot this value against the iteration number, resulting in the plots in Fig. 10(B).

The results in Fig. 10 show that the 3D fitting algorithm is both more robust than the 2D algorithm (Matthews and Baker, 2004) (A) and converges faster (B) in terms of the number of iterations. The results hold for both datasets. The improvement in performance is slightly more for the harder Dataset 2 (where the fitting is to subjects not in the training data) than for Dataset 1 (where the fitting it to subjects in the training data). The fact that the 3D algorithm is more robust than the 2D algorithm may at first glance be surprising; the 3D optimization is higher dimensional and so it might be expected to suffer from more local minima. This reasoning is incorrect. Instead, we argue that because the 2D model can generate "physically unrealizable" model instances (see Section 3.2), it is actually more prone to local minima than a 2D model constrained to only move as though it were 3D. Because of the tight coupling of parameters, the effective number of degrees of freedom of the 2D model is more than that of the 3D model.

5.6. Computational Cost

In Table 1 we compare the fitting speed of the 3D fitting algorithm with that of the 2D fitting algorithm (Matthews and Baker, 2004). We include results for both the model in Dataset 1 (8 2D shape modes, 4 similarity parameters, 40 appearance modes, 4 3D shape modes, 6 camera parameters, and 29,976 color pixels) and the model in Dataset 2 (9 2D shape modes, 4 similarity parameters, 59 appearance modes, 4 3D shape modes, 6 camera parameters, and 30,000 color pixels). The 2D fitting algorithm (Matthews and Baker, 2004) operates at 70–80 frames per second and the 3D fitting algorithm operates at 65–70 frames per second, both results computed on a dual,

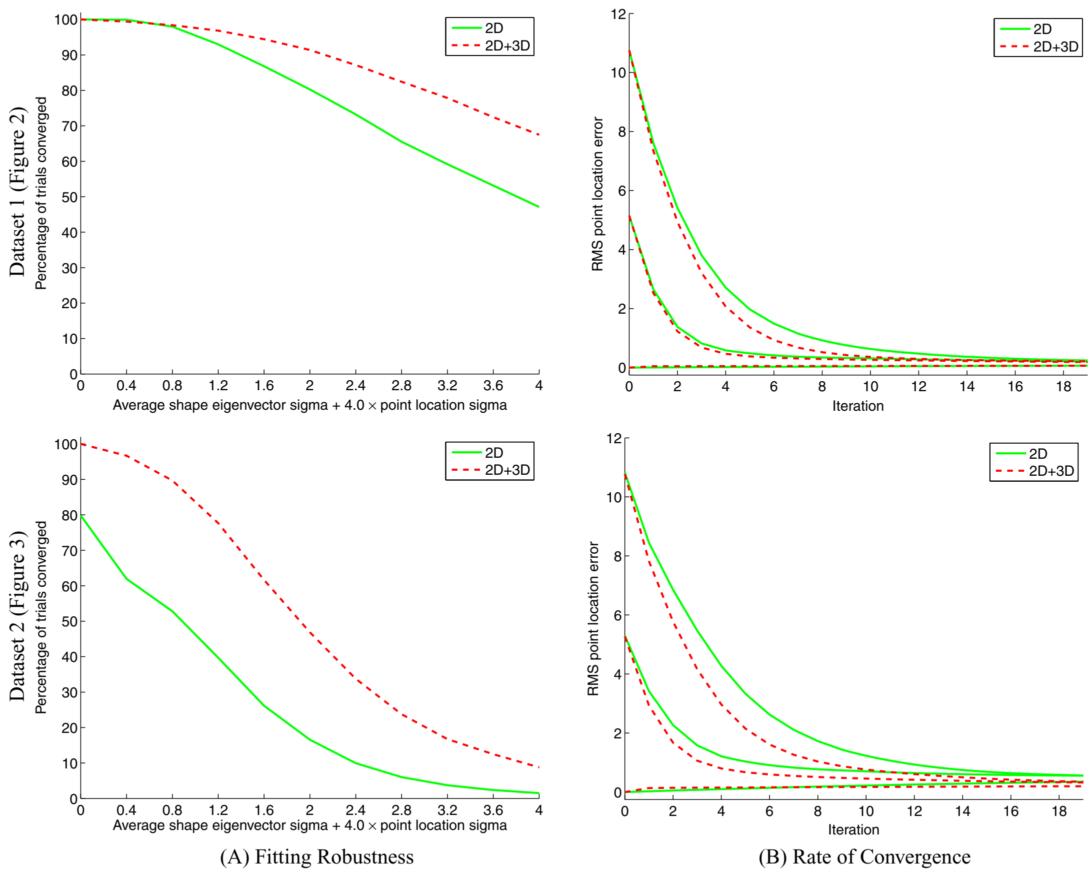

Figure 10. A comparison of the fitting robustness and rate of convergence of the 2D fitting algorithm (Matthews and Baker, 2004) with the 3D fitting algorithm following the same evaluation protocol as Matthews and Baker (2004). The results show that the 3D algorithm is both more robust (A) than the 2D algorithm and converges faster (B) in terms of number of iterations.

dual-core 2.5 GHz PowerPC Macintosh G5 with 4 GB of RAM. Note that the 3D algorithm requires slightly fewer iterations to converge on average than the 2D algorithm, particularly for Dataset 2, further validating the results in Fig. 10(B).

Table 1. Fitting speed comparison on a dual, dual-core 2.5 GHz PowerPC Macintosh G5 with 4 GB of RAM. We include results for both the model in Dataset 1 (8 2D shape modes, 4 similarity parameters, 40 appearance modes, 4 3D shape modes, 6 camera parameters, and 29,976 color pixels) and the model in Dataset 2 (9 2D shape modes, 4 similarity parameters, 59 appearance modes, 4 3D shape modes, 6 camera parameters, and 30,000 color pixels). Although the 3D algorithm is slightly slower than the 2D algorithm, it still operates comfortably in real-time, and requires slightly fewer iterations to converge.

	2D fitting		3D fitting	
	Frames per second	Iterations per frame	Frames per second	Iterations per frame
Dataset 1	73.1	2.74	65.4	2.72
Dataset 2	80.9	2.65	71.2	2.26

5.7. *Summary*

Although fitting 2D face models in real-time is possible using the inverse compositional algorithm, we have outlined in Section 5.2 why naively extending this algorithm to 3D is not possible. We also presented a solution to this difficulty, the simultaneous fitting of both a 2D and a 3D model. The result is a real-time 3D model fitting algorithm. Experimental results show 3D model fitting to be both inherently more robust and converge in fewer iterations than 2D model fitting, confirming that 3D is a more natural parameterization of faces than 2D.

5.8. *Related Approaches*

Another possible approach to 3D model fitting is the one taken in Romdhani and Vetter (2003). Due to the density of their models and a variety of other details, the current implementation of their algorithm is quite slow, requiring several seconds per frame. There seems to be no theoretical reason why the algorithm could not be implemented in real-time for the models of the complexity used in this paper. Romdhani and Vetter avoid the difficulty in 3D fitting described in Section 5.2 by "un-rolling" the 2D surface in 3D space and posing the problem as mapping from a 2D texture space to a 2D image. The 3D shape $\overline{\mathbf{p}}$ and the camera matrix $\mathbf{P}$ are embedded in a 2D shape warp $\mathbf{W}(\mathbf{u};\mathbf{p})$ rather than a 3D shape warp $\mathbf{W}(\mathbf{x};\overline{\mathbf{p}})$. The 2D inverse compositional algorithm is not directly applicable to the un-rolled texture maps because the identity warp is not one of the warps $\mathbf{W}(\mathbf{u};\mathbf{p})$ (Baker and Matthews, 2004). Romdhani and Vetter avoid this problem by generalizing the inverse compositional algorithm to allow a warp update:

$$\mathbf{W}(\mathbf{u};\mathbf{p}) \;\leftarrow\; \mathbf{W}(\mathbf{u};\mathbf{p}) \circ \mathbf{W}(\mathbf{u};\mathbf{p}^{\dagger})^{-1} \circ \mathbf{W}(\mathbf{u};\mathbf{p}^{\dagger}+\Delta\mathbf{p}) \tag{47}$$

where $\mathbf{p}^{\dagger}$ is a fixed vector of parameters. This equation should be compared with the usual update in Eq. (32). The incremental warp is $\mathbf{W}(\mathbf{u};\mathbf{p}^{\dagger}) \circ \mathbf{W}(\mathbf{u};\mathbf{p}^{\dagger}+\Delta\mathbf{p})^{-1}$ rather than $\mathbf{W}(\mathbf{u};\Delta\mathbf{p})$. This definition of the update leads to the Jacobians of the warp being evaluated at $\mathbf{p}^{\dagger}$ leading to a dependence on $\mathbf{p}^{\dagger}$. Empirically, the authors find that the algorithm performs better when $\mathbf{p} \approx \mathbf{p}^{\dagger}$. To ameliorate this undesirable property, the authors pre-compute the steepest descent images and Hessian for multiple different $\mathbf{p}^{\dagger}$ and use the closest one in the online phase of the algorithm. A direct comparison between our 3D fitting algorithm and the algorithm in Romdhani and Vetter (2003) on the same 3D model is outside the scope of this paper which focuses on comparing 2D and 3D face models. However, such a comparison is an interesting area for future work.

6. Conclusion

We have compared 2D and 3D face models along a variety of axes. The main conclusion is that 3D models are overall preferable to 2D models. The 3D parameterization is more compact (up to 6 times fewer parameters), more natural (i.e. head pose and non-rigid shape deformation are separated), 3D model fitting is more robust and requires fewer iterations to converge, and 3D occlusion reasoning is more powerful. Two of the frequently cited negatives of 3D models, slow fitting and construction requiring range data, have been addressed. We have presented a linear non-rigid structure-from-motion algorithm (Xiao et al., 2004) for the construction of a 3D model from 2D tracking results and a real-time 3D model fitting algorithm (Xiao et al., 2004). One final benefit of 3D models is that multi-camera model fitting is far easier (Koterba et al., 2005).

On the other hand, the differences between 2D and 3D models are not as great as is sometimes claimed. Ignoring the size of the model and assuming a scaled orthographic camera, 2D models have the same representational power as 3D, separating the head pose from the non-rigid head shape deformation is possible, and approximate occlusion reasoning is possible. The one limitation of 2D models that is hard to avoid is that fitting them is inherently less robust than 3D. The underlying cause of this problem is the "over-parameterization" of 2D models and the resulting ability to generate numerous "physically unrealizable" model instances. See Section 3.2.

7. Retrospective

We end by briefly describing how the research described in this paper was influenced by the work and philosophy of Takeo Kanade.

7.1. Application Area

This paper falls into the application area of "computer analysis of human faces," one of the most active areas in computer vision. This area was opened up by Takeo's PhD thesis (Kanade, 1973) which proposed the world's first computerized face recognition system. Since then, Takeo has published a number of other seminal papers in face processing, opening up new sub-areas to study, including face detection (Rowley et al., 1998) and facial expression recognition (Tian et al., 2001).

7.2. Techniques

Our algorithm for constructing 3D face models from 2D face models is a "factorization" algorithm. This non-rigid structure-from-motion algorithm is an extension of Takeo's original "factorization" algorithm for rigid structure-from-motion (Tomasi and Kanade, 1992). Our 2D and 2D + 3D face model fitting algorithms are both image alignment algorithms, and ultimately extensions of Takeo's original "Lucas-Kanade" algorithm (Lucas and Kanade, 1981).

7.3. Philosophy

On a philosophical level, Takeo has emphasized the importance of "registration" in face processing; i.e. the accurate mapping of image pixels onto the anatomical parts of the face to which they correspond. He has argued that accurate registration is a pre-requisite for robust face recognition, and most other face processing tasks. Our interest in the problems of face model construction and fitting was initially aroused by the potential use of face models as a general purpose registration algorithm, by which the image pixels are registered with anatomical locations in the face model.

Another philosophical point that Takeo has argued is the importance of using the "appropriate" models in computer vision. For example, his work on "factorization" (Tomasi and Kanade, 1992) illustrates the benefits of using a linear camera model, which although rarely a perfect model, is often sufficiently accurate, has the benefits of simplicity, and is often the most "appropriate" model. In this paper, we have attempted to characterize the benefits of 2D and 3D face models along various axes, a characterization that we hope will help readers determine whether 2D or 3D models are the most appropriate for their application.

Appendix A: Computing the Steepest Descent Geometry $\mathbf{SD}_{F_{t_i}}$

In Section 5.3 we omitted the details of how the steepest descent geometry constraints:

$$\mathbf{SD}_{F_{t_i}} = \left(\frac{\partial F_{t_i}}{\partial \mathbf{p}} \mathbf{J_p} \, \frac{\partial F_{t_i}}{\partial \mathbf{q}} \mathbf{J_q} \, \frac{\partial F_{t_i}}{\partial \overline{\mathbf{p}}} \, \frac{\partial F_{t_i}}{\partial \mathbf{P}} \, \frac{\partial F_{t_i}}{\partial o_u} \, \frac{\partial F_{t_i}}{\partial o_v} \right) \quad (48)$$

are computed. The first two components of $\mathbf{SD}_{F_{t_i}}$ are computed as follows. Since the only component of F_{t_i} that includes $\mathbf{p}$ or $\mathbf{q}$ is $\mathbf{N}(\mathbf{s}_0 + \sum_{i=1}^{m} p_i \, \mathbf{s}_i; \mathbf{q})$, $\frac{\partial F_{t_i}}{\partial \mathbf{p}} \mathbf{J_p}$ is the (negative of the) rate of change of the destination of the warp in the right hand side of Eq. (43) with respect to $\Delta\mathbf{p}$. Similarly, $\frac{\partial F_{t_i}}{\partial \mathbf{q}} \mathbf{J_q}$ is the (negative of the) rate of change of the destination of the warp in the right hand side of Eq. (43) with respect to $\Delta\mathbf{q}$. These two expressions can be estimated by setting each of the parameters in $\Delta\mathbf{p}$ and $\Delta\mathbf{q}$ in turn to be a small value (say, 1.0), setting all the other parameters to be 0.0, and then using the warp composition (see Matthews and Baker, 2004) to estimate the perturbation to the destination of the right-hand side of Eq. (43).

The other components of $\mathbf{SD}_{F_{t_i}}$ are somewhat simpler:

$$\frac{\partial F_{t_i}}{\partial \overline{p}_j} = \mathbf{P}\mathbf{s}_j, \qquad \frac{\partial F_{t_i}}{\partial o_u} = \begin{pmatrix} 1 & \dots & 1 \\ 0 & \dots & 0 \end{pmatrix}, \qquad \frac{\partial F_{t_i}}{\partial o_v} = \begin{pmatrix} 0 & \dots & 0 \\ 1 & \dots & 1 \end{pmatrix}. \quad (49)$$

The camera matrix $\mathbf{P}$ is more complicated because it has 6 variables, but only 4 degrees of freedom. We perform the update similarly to how 3D rotations are treated in Shum and Szeliski (2000). We first extract the scale from the matrix:

$$\mathbf{P} = \sigma \begin{pmatrix} i_x & i_y & i_z \\ j_x & j_y & j_z \end{pmatrix} \quad (50)$$

and constrain the projection axes $\mathbf{i} = (i_x, i_y, i_z)$ and $\mathbf{j} = (j_x, j_y, j_z)$ to be orthonormal. We compute the update to the scale $\Delta\sigma$ using:

$$\frac{\partial F_{t_i}}{\partial \sigma} = \begin{pmatrix} i_x & i_y & i_z \\ j_x & j_y & j_z \end{pmatrix} \mathbf{s} \quad (51)$$

where $\mathbf{s} = (\overline{\mathbf{s}}_0 + \sum_{i=1}^{\overline{m}} \overline{p}_i \, \overline{\mathbf{s}}_i)$ and then update the scale $\sigma \leftarrow \sigma + \Delta\sigma$. We also compute small angle updates $\Delta\theta_x$, $\Delta\theta_y$, $\Delta\theta_z$ to the camera matrix using:

$$\frac{\partial F_{t_i}}{\partial \Delta\theta_x} = \mathbf{P} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \mathbf{s}, \; \frac{\partial F_{t_i}}{\partial \Delta\theta_y} = \mathbf{P} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \mathbf{s},$$

$$\frac{\partial F_{t_i}}{\partial \Delta\theta_y} = \mathbf{P} \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{s} \quad (52)$$

update the camera matrix:

$$\mathbf{P} \leftarrow \mathbf{P} \begin{pmatrix} 1 & -\Delta\theta_z & \Delta\theta_y \\ \Delta\theta_z & 1 & -\Delta\theta_x \\ -\Delta\theta_y & \Delta\theta_x & 1 \end{pmatrix} \quad (53)$$

and finally reinforce the orthonormality constraints on $\mathbf{P}$.

Notes

1. Note that this result is required for the Constrained 3D Fitting algorithm in Section 5.3 to be meaningful.
2. If the first $(\overline{m} + 1)$ shape vectors are not independent, we can find $(\overline{m} + 1)$ frames in which the shapes are independent by examining the singular values of their image projections. We can then reorder the images.
3. A quick note on this interpretation of the algorithm. We optimize a function $G^2(\text{2D}) + K \cdot F^2(\text{2D, 3D})$ where G models how well the 2D model matches the image, F models how well the 2D model matches the projection of the 3D model, and K is a large weight. One interpretation of optimizing this function is that F is a just prior on G. If F is such that given 3D, there is just a single set 2D such that $F^2(\text{2D, 3D}) = 0$ (which is the case for us), then an equally valid interpretation of the algorithm is that G is optimized with respect to the 3D parameters through the 2D ones.

Acknowledgments

The research described in this paper was conducted while all three authors were with The Robotics Institute at Carnegie Mellon University, and was supported in part by Denso Corporation, U.S. Department of Defense contract N41756-03-C4024, and National Institute of Mental Health grant R01 MH51435. A preliminary version appeared in the 2004 IEEE Conference on Computer Vision and Pattern Recognition (Xiao et al., 2004). We thank the reviewers of that paper, and this, for their feedback. We also thank Tim Cootes, Sami Romdhani, and Thomas Vetter for their comments. We also thank our collaborators at CMU for their input, including Ralph Gross, Seth Koterba, Jeff Cohn, Changbo Hu, Raju Patil, Goksel Dedeoglu, Krishnan Ramnath, Takahiro Ishikawa, German Cheung, Adrian Broadhurst, Junya Inada, and last but not least, Takeo Kanade.

References

1. Baker, S., Gross, R., and Matthews, I. 2003. Lucas-Kanade 20 years on: A unifying framework: Part 3. Technical Report CMU-RI-TR-03-35, Carnegie Mellon University Robotics Institute.
2. Baker, S., Gross, R., and Matthews. I. 2004. Lucas-Kanade 20 years on: Part 4. Technical Report CMU-RI-TR-04-14, Carnegie Mellon University Robotics Institute.
3. Baker, S. and Matthews, I. 2001. Equivalence and efficiency of image alignment algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1, pp. 1090–1097.
4. Baker, S. and Matthews, I. 2004. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56:221–255.
5. Baker, S., Patil, R., Cheung, K.M., and Matthews, I. 2004. Lucas-Kanade 20 years on: Part 5. Technical Report CMU-RI-TR-04-64, Carnegie Mellon University Robotics Institute.
6. Blanz, V., Basso, C., Poggio, T., and Vetter, T. 2003. Reanimating faces in images and video. In *Proceedings of EUROGRAPHICS*.
7. Blanz, V. and Vetter, T. 1999. A morphable model for the synthesis of 3D faces. In *Computer Graphics, Annual Conference Series (SIGGRAPH)*, pp. 187–194.
8. Brand, M. 2001. Morphable 3D models from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 2, pp. 456–463.
9. Bregler, C., Hertzmann, A., and Biermann, H. 2000. Recovering non-rigid 3D shape from image streams. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
10. Cootes, T., Edwards, G., and Taylor, C. 2001 Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685.
11. Gross, R., Matthews, I., and Baker, S. 2004. Constructing and fitting active appearance models with occlusion. In *Proceedings of the IEEE Workshop on Face Processing in Video*, pp. 72–79.
12. Hager, G. and Belhumeur, P. 1998. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039.
13. Ishikawa, T., Baker, S., Matthews, I., and Kanade, T. 2004. Passive driver gaze tracking with active appearance models. In *Proceedings of the 11th World Congress on Intelligent Transportation Systems*, URL: http://www.ri.cmu.edu/pubs/pub_4705.html.
14. Kanade, T. 1973. Picture Processing System by Computer Complex and Recognition of Human Faces. PhD thesis, Kyoto University, November.
15. Koterba, S., Baker, S., Matthews, I., Hu, C., Xiao, J., Cohn, J., and Kanade. T. 2005. Multi-view AAM fitting and camera calibration. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 511–518.
16. Lanitis, A., Taylor, C., and Cootes, T. 1997. Automatic interpretation and coding of face images using flexible models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):742–756.
17. Lucas, B. and Kanade, T. 1981. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679.
18. Matthews, I. and Baker, S. 2004. Active Appearance Models revisited. *International Journal of Computer Vision*, 60(2):135–164.
19. Matthews, I., Cootes, T., Bangham, A., Cox, S., and Harvery, R. 2002. Extraction of visual features for lipreading. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):198–213.
20. Matthews, I., Xiao, I., and Baker, S. 2006. On the dimensionality of deformable face models. Technical Report CMU-RI-TR-06-12, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.
21. Romdhani, S. and Vetter, T. 2003. Efficient, robust and accurate fitting of a 3D morphable model. In *Proceedings of the International Conference on Computer Vision*, 1:59–55.
22. Rowley, H.A., Baluja, S., and Kanade, T. 1998. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38.
23. Shum, H.Y. and Szeliski, R. 2000. Construction of panoramic image mosaics with global and local alignment. *International Journal of Computer Vision*, 16(1):63–84.
24. Tian, Y., Kanade, T., and Cohn, J. 2001. Recognizing action units for facial expression analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):97–115.
25. Tomasi, C. and Kanade, T. 1992. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision*, 9(2):137–154.
26. Torresani, L., Yang, D., Alexander, G., and Bregler, C. 2001. Tracking and modeling non-rigid objects with rank constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1:493–500.
27. Xiao, J., Baker, S., Matthews, I., and Kanade, T. 2004a. Real-time combined 2D + 3D Active Appearance Models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 2, pp. 535–542.
28. Xiao, J., Chai, J., and Kanade, T. 2004b. A closed-form solution to non-rigid shape and motion recovery. In *Proceedings of the European Conference on Computer Vision*, Vol. 4, pp. 573–587.
29. Xiao, J., Chai, J., and Kanade, T. 2006. A closed-form solution to non-rigid shape and motion recovery. *International Journal of Computer Vision*, 67(2):233–246.