Real-Time Combined 2D+3D Active Appearance Models

Jing Xiao, Simon Baker, Iain Matthews, and Takeo Kanade

The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213

Abstract

Active Appearance Models (AAMs) are generative models commonly used to model faces. Another closely related type of face models are 3D Morphable Models (3DMMs). Although AAMs are 2D, they can still be used to model 3D phenomena such as faces moving across pose. We first study the representational power of AAMs and show that they can model anything a 3DMM can, but possibly require more shape parameters. We quantify the number of additional parameters required and show that 2D AAMs can generate model instances that are not possible with the equivalent 3DMM. We proceed to describe how a non-rigid structure-from-motion algorithm can be used to construct the corresponding 3D shape modes of a 2D AAM. We then show how the 3D modes can be used to constrain the AAM so that it can only generate model instances that can also be generated with the 3D modes. Finally, we propose a real-time algorithm for fitting the AAM while enforcing the constraints, creating what we call a "Combined 2D+3D AAM."

1 Introduction

Active Appearance Models (AAMs) [5] are generative models commonly used for faces [7]. Another class of face models are 3D Morphable Models (3DMMs) [2]. Although AAMs and 3DMMs are very similar in many respects, one major difference (although not the only one) between them is that the shape component of an AAM is 2D whereas the shape component of a 3DMM is 3D. The fact that AAMs are 2D, however, does not mean that they do not contain 3D information or cannot represent 3D phenomena.

We begin in Section 2 by briefly reviewing AAMs and 3DMMs emphasizing the sense in which their respective shape models are 2D and 3D. We then study the extent to which AAMs are 3D. Section 3 discusses the representational power of 2D shape models. We first show that (under weak perspective imaging) 2D models can represent the same 3D phenomena that 3D models can, albeit with a larger number of parameters. We also quantify the number of extra parameters required. Because the equivalent 2D model requires more parameters than the 3D, it must be able to generate model instances that are impossible with the 3D model. In Section 3.2 we give a concrete example.

Whether being able to generate these extra model instances is a good thing or not is open to debate. One argument is that these model instance are "impossible" cases of the underlying 3D object. It would therefore be preferable if we could constrain the AAM parameters so that the AAM cannot generate these "impossible" cases. Ideally we would like the AAM to be only able to generate model instances that could have been generated by the equivalent 3D shape modes. This should improve fitting performance.

There are two other advantages of doing this, rather than directly using the equivalent 3DMM. The first advantage is fitting speed. Currently the fastest AAM fitting algorithms operate at over 200 frames per second [8]. We would like to combine the benefits of the 3D shape parameterization (such as explicit 3D shape and pose recovery) and the fitting speed of a 2D AAM. The second advantage is ease of model construction. AAMs can be computed directly from 2D images, whereas constructing a 3DMM usually requires 3D range data [2] (although there are exceptions, e.g. [3].)

In Section 4 we describe how to constrain a 2D AAM with the equivalent 3D shape modes to create what we call a "Combined 2D+3D AAM." In Section 4.1 we describe how a non-rigid structure-from-motion algorithm can be used to compute the equivalent 3D shape modes from an AAM. In Section 4.2 we show how these 3D shape modes can be used to constrain the AAM shape parameters so that the AAM can only generate model instances that could have been generated by the 3D shape modes. Finally, in Section 4.3 we propose a real-time fitting algorithm that enforces these constraints. While fitting, this algorithm explicitly recovers the 3D pose and 3D shape of the face.

2 Background

We begin with a brief review of Active Appearance Models (AAMs) [5] and 3D Morphable Models (3DMMs) [2]. We have taken the liberty to simplify the presentation and change the notation from [5] and [2] to highlight the similarities and differences between the two types of models.

2.1 Active Appearance Models: AAMs

The *2D shape* of an AAM is defined by a 2D triangulated mesh and in particular the vertex locations of the mesh.

Mathematically, we define the shape $\mathbf{s}$ of an AAM 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 p_i 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 [5]. Principal Component Analysis (PCA) is then applied to the training meshes. The base shape $\mathbf{s}_0$ is the mean shape and the matrices $\mathbf{s}_i$ are the (reshaped) 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 abuse of terminology. 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 $A_i(\mathbf{u})$:

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

where the coefficients λ_i are the appearance parameters. As with the shape, the base appearance A_0 and appearance images A_i are usually computed by applying PCA to the (shape normalized) training images [5].

Although Equations (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 $\mathbf{q} = (q_1, \dots, q_4)^{\mathrm{T}}$ contains the rotation, translation, and scale parameters [8]. Given the AAM shape parameters $\mathbf{p} = (p_1, \dots, p_m)^{\mathrm{T}}$, Equation (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 $\mathbf{N}(\mathbf{s}; \mathbf{q})$, another convenient abuse of terminology. Similarly, Equation (3) is used to generate the AAM appearance $A(\mathbf{u})$ from the AAM appearance parameters $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_l)^{\mathrm{T}}$. 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 $\mathbf{W}(\mathbf{u}; \mathbf{p}; \mathbf{q})$. 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 from one set of vertices to the other.

2.2 3D Morphable Models: 3DMMs

The *3D shape* of a 3DMM is defined by a 3D triangulated mesh and in particular the vertex locations of the mesh. Mathematically, we define the shape $\overline{\mathbf{s}}$ of a 3DMM as the 3D coordinates of the $\overline{n}$ vertices that make up the mesh:

$$\overline{\mathbf{s}} = \begin{pmatrix} x_1 & x_2 & \dots & x_{\overline{n}} \\ y_1 & y_2 & \dots & y_{\overline{n}} \\ z_1 & z_2 & \dots & z_{\overline{n}} \end{pmatrix}. \quad (4)$$

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 $\overline{\mathbf{s}}_i$:

$$\overline{\mathbf{s}} = \overline{\mathbf{s}}_0 + \sum_{i=1}^{\overline{m}} \overline{p}_i \, \overline{\mathbf{s}}_i \quad (5)$$

where the coefficients $\overline{p}_i$ are the shape parameters.

3DMMs are normally computed from training data consisting of a number of range images with the mesh vertices (hand) marked in them [2]. 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 largest eigenvalues.

The *appearance* of a 3DMM is defined within a 2D triangulated mesh that has the same topology (vertex connectivity) as the base mesh $\overline{\mathbf{s}}_0$. Let $\overline{\mathbf{s}}_0^*$ denote the set of pixels $\mathbf{u} = (u, v)^{\mathrm{T}}$ that lie inside this 2D mesh. The appearance is then an image $\overline{A}(\mathbf{u})$ defined over $\mathbf{u} \in \overline{\mathbf{s}}_0^*$. 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 $\overline{A}_i(\mathbf{u})$:

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

where the coefficients $\overline{\lambda}_i$ are the appearance parameters. As with the shape, the base appearance $\overline{A}_0$ and the appearance images $\overline{A}_i$ are usually computed by applying PCA to the texture components of the training range images, appropriately warped onto the 2D triangulated mesh $\overline{\mathbf{s}}_0^*$ [2].

To generate a 3DMM *model instance* we need an image formation model to convert the 3D shape $\overline{\mathbf{s}}$ into a 2D mesh. As in [9], we use the weak perspective model (which is an adequate approximation unless the face is very close to the camera) defined by the matrix:

$$\begin{pmatrix} i_x & i_y & i_z \\ j_x & j_y & j_z \end{pmatrix} \quad (7)$$

and the offset of the origin $(o_x, o_y)^{\mathrm{T}}$. 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 $\mathbf{x} = (x, y, z)^{\mathrm{T}}$ is:

$$\mathbf{u} = \mathbf{P}\,\mathbf{x} = \begin{pmatrix} i_x & i_y & i_z \\ j_x & j_y & j_z \end{pmatrix} \mathbf{x} + \begin{pmatrix} o_x \\ o_y \end{pmatrix}. \quad (8)$$

Note that the projection $\mathbf{P}$ 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 Equation (4). Each 3D vertex $(x_i, y_i, z_i)^{\mathrm{T}}$ is then mapped to a 2D vertex using the imaging model in Equation (8). (Note that during this process the visibility of the triangles in the mesh should be respected.) The appearance is then computed using Equation (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 applying the image formation model (Equation (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, Equations (2) and (5) are almost identical. Equations (3) and (6) are also almost identical. The main difference between the two types of model is that the shape component of the AAM is 2D (see Equation (1)) whereas that of the 3DMM is 3D (see Equation (4)).

Note, however, that there are other differences between AAMs [5] and 3DMMs [2]. (1) 3DMMs are usually constructed to be denser; i.e. consist of more triangles. (2) Because of their 3D shape and density, 3DMMs can also use the surface normal in their appearance model. (3) Because of their 3D shape, 3DMMs can model occlusion, whereas 2D AAMs cannot. In this paper, we ignore these differences and focus on the dimensionality of the shape model.

3 Representational Power

We now study the representational power of 2D and 3D shape models. We first show that a 2D shape model can represent anything a 3D model can. We then show that 2D models can generate many model instances that are not possible with an otherwise equivalent 3D model.

3.1 Can 2D Shape Models Represent 3D?

Given a 3D shape model, is there a 2D shape model that can generate the same set of model instances? In this section, we show that the answer to this question is yes.

The shape variation of a 2D model is described by Equation (2) and $\mathbf{N}(\mathbf{u}; \mathbf{q})$. That of an 3D model is described by Equations (5) and (8). We can ignore the offset of the origin $(o_x, o_y)^{\mathrm{T}}$ in the weak perspective model for the 3D 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 given by:

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

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:

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

Equation (9) is therefore a linear combination of:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \overline{\mathbf{s}}_i, \quad \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \overline{\mathbf{s}}_i, \ldots, \quad (11)$$

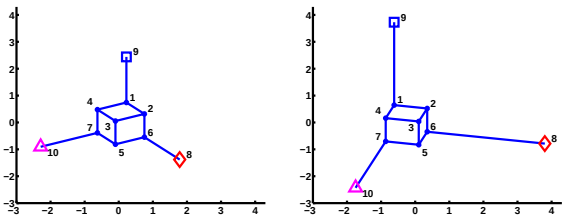
for $i = 0, 1, \ldots, \overline{m}$, and similarly for the other 4 constant matrices in Equation (10). The linear shape variation of the 3D model can therefore be represented by an appropriate set of 2D shape vectors. For example:

$$\mathbf{s}_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \overline{\mathbf{s}}_0, \; \mathbf{s}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \overline{\mathbf{s}}_1, \ldots \quad (12)$$

and so on. 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 $\overline{m}$ shape vectors. Although many more shape vectors may be needed, the main point is that 2D model can represent any phenomena that the 3D model can. (Note that although more than 6 times as many shape vectors may be needed to model the same phenomenon, in practice often not that many are required.)

3.2 Do 2D Models Generate Invalid Cases?

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. If the parameters of the 2D model are chosen so that the orthogonality constraints on the corresponding 3D projection axes **i** and **j** do



(a) (b)

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 moved.

not hold, the 2D model instance is not realizable with the 3D model. An example of this is presented in Figure 1.

The scene in Figure 1 consists of a static cube (of which 7 vertices are visible) and 3 moving points (marked with a diamond, a triangle, and a 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 points (diamond, triangle, square.)

We randomly generated 60 sets of shape parameters $\overline{p}_1$ (see Equation (5)) and camera projection matrices $\mathbf{P}$ (see Equation (8)) and synthesized 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 a large number of shapes that are impossible to generate with the 3D model. One concrete example is the base shape of the 2D model $\mathbf{s}_0$. In our experiment, $\mathbf{s}_0$ turns out to be: $\mathbf{s}_0 =$

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

We now need to show that s_0 is not a 3D model instance. It is possible to show that $\mathbf{s}_0$ can be *uniquely* decomposed into: $\mathbf{s}_0 = \mathbf{P}_0\bar{\mathbf{s}}_0 + \mathbf{P}_1\bar{\mathbf{s}}_1 =$

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

It is easy to see that $\mathbf{P}_0$ is not a constant multiple of $\mathbf{P}_1$, and neither of $\mathbf{P}_0$ and $\mathbf{P}_1$ are legitimate weak perspective matrices (i.e. composed of two equal length, orthogonal vectors). Therefore, we have shown that the 2D model instance $\mathbf{s}_0$ is not a valid 3D model instance.

4 Combined 2D+3D AAMs

We now describe how to constrain an AAM with the equivalent 3D shape modes and create what we call a "Combined 2D+3D AAM." We also derive a real-time fitting algorithm for a Combined 2D+3D AAM that explicitly recovers the 3D pose and 3D shape of the face.

4.1 Computing 3D Shape from an AAM

If we have a 2D AAM, a sequence of images $I^t(\mathbf{u})$ for $t = 0, \ldots, N$, and have tracked the face through the sequence with the AAM, then denote the AAM shape parameters at time t by $\mathbf{p}^t = (p_1^t, \ldots, p_m^t)^{\mathrm{T}}$. Using Equation (2) we can compute the 2D AAM shape vector $\mathbf{s}^t$ for each time t:

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

A variety of non-rigid structure-from-motion algorithms have been proposed to convert the tracked feature points in Equation (15) into 3D linear shape modes. Bregler et al. [4] proposed a factorization method to simultaneously reconstruct the non-rigid shape and camera matrices. This method was extended to a trilinear optimization approach in [11]. The optimization process involves three types of unknowns, shape vectors, shape parameters, and projection matrices. At each step, two of the unknowns are fixed and the third refined. Brand [3] proposed a similar non-linear optimization method that used an extension of Bregler's method for initialization. All of these methods only use the usual orthonormality constraints on the projection matrices [10]. In [12] we proved that only enforcing the orthonormality constraints is ambiguous and demonstrate that it can lead to an incorrect solution. We now outline how our algorithm [12] can be used to compute 3D shape modes from an AAM. (Any of the other algorithms could be used instead, although with worse results.) We stack the 2D AAM shape vectors in all N images into a measurement matrix:

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

If this data can be explained by a set of 3D linear shape modes, then W can be represented: $W = MB =$

$$\begin{pmatrix} \mathbf{P}^0 & p_1^0\mathbf{P}^0 & \ldots & p_{\overline{m}}^0\mathbf{P}^0 \\ \mathbf{P}^1 & p_1^1\mathbf{P}^1 & \ldots & p_{\overline{m}}^1\mathbf{P}^1 \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{P}^N & p_1^N\mathbf{P}^N & \ldots & p_{\overline{m}}^N\mathbf{P}^N \end{pmatrix} \begin{pmatrix} \bar{\mathbf{s}}_0 \\ \vdots \\ \bar{\mathbf{s}}_{\overline{m}} \end{pmatrix} \quad (17)$$

where 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 vertices $\overline{n}$ to equal 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, and 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 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 the corrective matrix. In [12] we proposed additional *basis* constraints to compute G. See [12] for the details. Once G has been determined, M and B can be recovered. In summary, the 3D shape modes have been computed from the 2D AAM shape modes and the 2D AAM tracking results. Note that the tracking data is needed.

4.1.1 Experimental Results

We illustrate the computation of the 3D shape modes from an AAM in Figure 2. We first constructed an AAM for 5 people using 20 training images of each person. In Figures 2(a–c) we include the AAM mean shape $\mathbf{s}_0$ and the first 2 (of 17) AAM shape variation modes $\mathbf{s}_1$ and $\mathbf{s}_2$. Figures 2(d–f) illustrate the mean AAM appearance λ_0 and the first 2 (of 42) AAM appearance variation modes λ_1 and λ_2. The AAM is then fit to short videos (in total 900 frames) of each of the 5 people and the results used to compute the 3D shape modes. The mean shape $\overline{\mathbf{s}}_0$ and first 2 (of 15) shape modes $\overline{\mathbf{s}}_1$ and $\overline{\mathbf{s}}_2$ are illustrated in Figures 2(g–i).

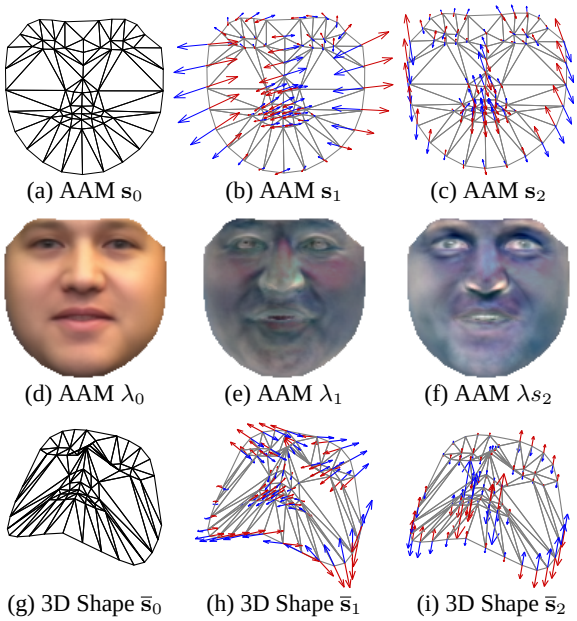
4.2 Constraining an AAM with 3D Shape

The 3D shape modes just computed are a 3D model of the same phenomenon that the AAM modeled. We now derive constraints on the 2D AAM shape parameters $\mathbf{p} = (p_1, \ldots, p_m)$ that force the AAM to only move in a way that is consistent with the 3D shape modes. If we denote:

$$\mathbf{P} \begin{pmatrix} x_1 & \ldots & x_n \\ y_1 & \ldots & y_n \\ z_1 & \ldots & z_n \end{pmatrix} = \left(\mathbf{P} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \ldots \mathbf{P} \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} \right) \tag{19}$$

then the 2D shape variation of the 3D shape modes over all imaging conditions is:

$$\mathbf{P} \left(\overline{\mathbf{s}}_0 + \sum_{i=1}^{\overline{m}} \overline{p}_i \, \overline{\mathbf{s}}_i \right) \tag{20}$$



(a) AAM $\mathbf{s}_0$ (b) AAM $\mathbf{s}_1$ (c) AAM $\mathbf{s}_2$

(d) AAM λ_0 (e) AAM λ_1 (f) AAM λs_2

(g) 3D Shape $\overline{\mathbf{s}}_0$ (h) 3D Shape $\overline{\mathbf{s}}_1$ (i) 3D Shape $\overline{\mathbf{s}}_2$

Figure 2: An example of the computation of 3D shape modes from an AAM. The figure shows the AAM shape (a–c) and appearance (d–f) variation, and the first three 3D shape modes (g–i).

where $\mathbf{P}$ and $\overline{\mathbf{p}} = (\overline{p}_i, \ldots, \overline{p}_{\overline{m}})$ vary over their allowed values. (Note that $\mathbf{P}$ is a function $\mathbf{P} = \mathbf{P}(o_x, o_y, \mathbf{i}, \mathbf{j})$ and so it is the parameters $o_x, o_y, \mathbf{i}, \mathbf{j}$ that are actually varying.)

The constraints on the AAM shape parameters $\mathbf{p}$ that we seek to impose are that there exist legitimate values of $\mathbf{P}$ and $\overline{\mathbf{p}}$ such that the 2D projected 3D shape equals the 2D shape of the AAM. These constraints can be written:

$$\min_{\mathbf{P}, \overline{\mathbf{p}}} \left\| \mathbf{N} \left(\mathbf{s}_0 + \sum_{i=1}^{m} p_i \, \mathbf{s}_i; \mathbf{q} \right) - \mathbf{P} \left(\overline{\mathbf{s}}_0 + \sum_{i=1}^{\overline{m}} \overline{p}_i \, \overline{\mathbf{s}}_i \right) \right\|^2 = 0 \tag{21}$$

where $\| \cdot \|^2$ denotes the sum of the squares of the elements of the matrix. The only quantities in Equation (21) that are not either known (m, $\overline{m}$, $\mathbf{s}_i$, $\overline{\mathbf{s}}_i$) or optimized ($\mathbf{P}$, $\overline{\mathbf{p}}$) are $\mathbf{p}$, $\mathbf{q}$. Equation (21) is therefore a set of constraints on $\mathbf{p}$, $\mathbf{q}$.

4.3 Fitting with 3D Shape Constraints

We now briefly outline our algorithm to fit an AAM while enforcing these constraints. In particular, we extend our real-time AAM fitting algorithm [8]. The result is an algorithm that turns out to be even faster than the 2D algorithm. The goal of AAM fitting [8] is to minimize:

$$\sum_{\mathbf{u} \in \mathbf{s}_0} \left[A_0(\mathbf{u}) + \sum_{i=1}^{l} \lambda_i A_i(\mathbf{u}) - I(\mathbf{W}(\mathbf{u}; \mathbf{p}; \mathbf{q})) \right]^2 \tag{22}$$

simultaneously with respect to the AAM shape $\mathbf{p}$, appearance $\boldsymbol{\lambda}$, and normalization $\mathbf{q}$ parameters. We impose the

constraints in Equation (21) as soft constraints on Equation (22) with a large weight K; i.e. we re-pose AAM fitting as simultaneously minimizing:

$$\sum_{\mathbf{u}\in\mathbf{s}_0}\left[A_0(\mathbf{u})+\sum_{i=1}^{l}\lambda_i A_i(\mathbf{u})-I(\mathbf{W}(\mathbf{u};\mathbf{p};\mathbf{q}))\right]^2+$$
$$K\left\|\mathbf{N}\left(\mathbf{s}_0+\sum_{i=1}^{m}p_i\,\mathbf{s}_i;\mathbf{q}\right)-\mathbf{P}\left(\overline{\mathbf{s}}_0+\sum_{i=1}^{\overline{m}}\overline{p}_i\,\overline{\mathbf{s}}_i\right)\right\|^2 \quad (23)$$

with respect to $\mathbf{p}$, $\mathbf{q}$, $\boldsymbol{\lambda}$, $\mathbf{P}$, and $\overline{\mathbf{p}}$. In the limit $K\rightarrow\infty$ the constraints become hard constraints. In practice, a suitably large value for K results in the system being solved approximately as though the constraints are hard.

The technique in [8] (proposed in [6]) to *sequentially* optimize for the AAM shape $\mathbf{p}$, $\mathbf{q}$, and appearance $\boldsymbol{\lambda}$ parameters can also be used on the above equation. We optimize:

$$\|A_0(\mathbf{u})-I(\mathbf{W}(\mathbf{u};\mathbf{p};\mathbf{q}))\|^2_{\mathrm{span}(A_i)^\perp}+$$
$$K\left\|\mathbf{N}\left(\mathbf{s}_0+\sum_{i=1}^{m}p_i\,\mathbf{s}_i;\mathbf{q}\right)-\mathbf{P}\left(\overline{\mathbf{s}}_0+\sum_{i=1}^{\overline{m}}\overline{p}_i\,\overline{\mathbf{s}}_i\right)\right\|^2 \quad (24)$$

with respect to $\mathbf{p}$, $\mathbf{q}$, $\mathbf{P}$, and $\overline{\mathbf{p}}$, 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:

$$\lambda_i \;=\; \sum_{\mathbf{u}\in\mathbf{s}_0}A_i(\mathbf{u})\cdot[I(\mathbf{W}(\mathbf{u};\mathbf{p};\mathbf{q}))-A_0(\mathbf{u})] \quad (25)$$

where the parameters $\mathbf{p}$, $\mathbf{q}$ are the result of the previous optimization. (Note that Equation (25) assumes that the appearance vectors $A_i(\mathbf{u})$ are orthonormal.) The optimality criterion in Equation (24) is of the form:

$$\|A_0(\mathbf{u})-I(\mathbf{W}(\mathbf{u};\mathbf{p};\mathbf{q}))\|^2_{\mathrm{span}(A_i)^\perp}+\;F(\mathbf{p};\mathbf{q};\mathbf{P};\overline{\mathbf{p}}). \quad (26)$$

In a recent journal paper [8] we showed how to minimize $\|A_0(\mathbf{u})-I(\mathbf{W}(\mathbf{u};\mathbf{p};\mathbf{q}))\|^2_{\mathrm{span}(A_i)^\perp}$ using the *inverse compositional* algorithm; i.e. by iteratively minimizing:

$$\|A_0(\mathbf{W}(\mathbf{u};\Delta\mathbf{p};\Delta\mathbf{q}))-I(\mathbf{W}(\mathbf{u};\mathbf{p};\mathbf{q}))\|^2_{\mathrm{span}(A_i)^\perp} \quad (27)$$

with respect to $\Delta\mathbf{p}$, $\Delta\mathbf{q}$ and then updating the current estimate of the warp using $\mathbf{W}(\mathbf{u};\mathbf{p};\mathbf{q})\leftarrow\mathbf{W}(\mathbf{u};\mathbf{p};\mathbf{q})\circ\mathbf{W}(\mathbf{u};\Delta\mathbf{p};\Delta\mathbf{q})^{-1}$. When using the inverse compositional algorithm we effectively change [1] the incremental updates to the parameters from $(\Delta\mathbf{p},\Delta\mathbf{q})$ to $\mathbf{J}(\Delta\mathbf{p},\Delta\mathbf{q})$ where:

$$\mathbf{W}(\mathbf{u};(\mathbf{p},\mathbf{q})+\mathbf{J}(\Delta\mathbf{p},\Delta\mathbf{q}))\;\approx$$
$$\mathbf{W}(\mathbf{u};\mathbf{p};\mathbf{q})\circ\mathbf{W}(\mathbf{u};\Delta\mathbf{p};\Delta\mathbf{p})^{-1} \quad (28)$$

to a first order approximation, and $\mathbf{J}$ is an $(m+4)\times(m+4)$ matrix. In general $\mathbf{J}$ depends on the warp parameters $(\mathbf{p},\mathbf{q})$ but can easily be computed. Equation (28) means that to optimize the expression in Equation (26) using the inverse compositional algorithm, we must iteratively minimize:

$$G(\Delta\mathbf{p},\Delta\mathbf{q})\;+\;F((\mathbf{p},\mathbf{q})+\mathbf{J}(\Delta\mathbf{p},\Delta\mathbf{q});\mathbf{P}+\Delta\mathbf{P};\overline{\mathbf{p}}+\Delta\overline{\mathbf{p}}) \quad (29)$$

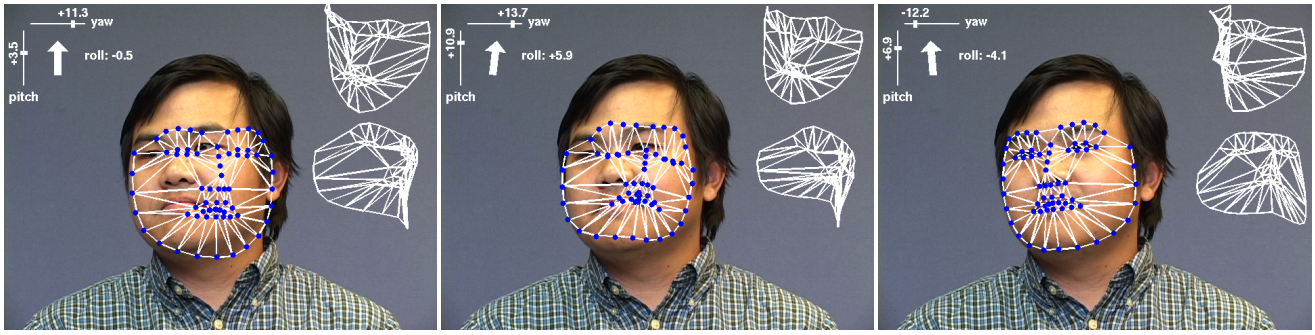
simultaneously with respect to $\Delta\mathbf{p}$, $\Delta\mathbf{q}$, $\Delta\mathbf{P}$, and $\Delta\overline{\mathbf{p}}$, where $G(\Delta\mathbf{p},\Delta\mathbf{q})$ is the expression in Equation (27).

The reason for using the inverse compositional algorithm is that the Gauss-Newton Hessian of the expression in Equation (27) is a constant and so can be precomputed [8]. It is easy to show (see [1] for the details) that the Gauss-Newton Hessian of the expression in Equation (29) is the sum of the Hessian for G and the Hessian for F. (This relies on the two terms each being a sum of squares.) Similarly, the Gauss-Newton steepest-descent parameter updates for the entire expression are the sum of the updates for the two terms separately. An efficient optimization algorithm can therefore be built based on the inverse compositional algorithm.

The Hessian for G is precomputed as in [8]. The Hessian for F is computed in the online phase and added to the Hessian for G. Since nothing in F depends on the images, the Hessian for F can be computed very efficiently. The steepest-descent parameter updates for G are also computed exactly as in [8] and added to the steepest-descent parameter updates for F. The final Gauss-Newton parameter updates can then be computed by inverting the combined Hessian and multiplying by the combined steepest-descent parameter updates. The warp parameters $\mathbf{p},\mathbf{q}$ are then updated $\mathbf{W}(\mathbf{u};\mathbf{p};\mathbf{q})\leftarrow\mathbf{W}(\mathbf{u};\mathbf{p};\mathbf{q})\circ\mathbf{W}(\mathbf{u};\Delta\mathbf{p};\Delta\mathbf{q})^{-1}$ and the other parameters additively $\mathbf{P}\leftarrow\mathbf{P}+\Delta\mathbf{P}$ and $\overline{\mathbf{p}}\leftarrow\overline{\mathbf{p}}+\Delta\overline{\mathbf{p}}$. One minor detail is the fact that G and F have different parameter sets to be optimized, $(\Delta\mathbf{p},\Delta\mathbf{q})$ and $(\Delta\mathbf{p};\Delta\mathbf{q};\Delta\mathbf{P};\Delta\overline{\mathbf{p}})$. The easiest way to deal with this is to think of G as a function of $(\Delta\mathbf{p};\Delta\mathbf{q};\Delta\mathbf{P};\Delta\overline{\mathbf{p}})$. All terms in both the Hessian and the steepest-descent parameter updates that relate to either $\Delta\mathbf{P}$ or $\Delta\overline{\mathbf{p}}$ are set to zero.

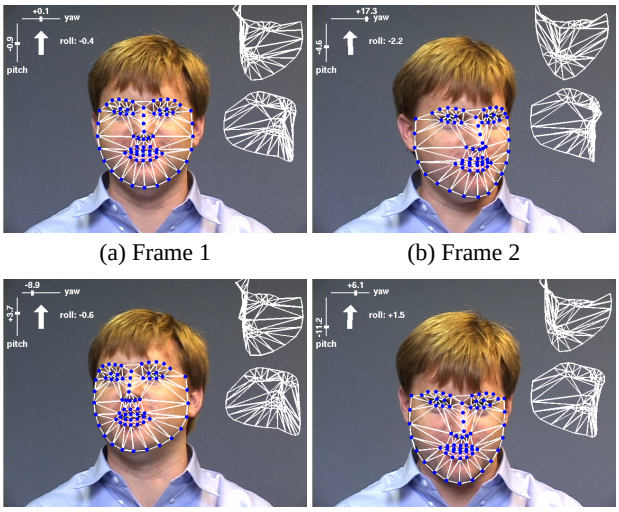
4.3.1 Experimental Results

In Figure 3 we include an example of our algorithm fitting to a single input image. Figure 3(a) displays the initial configuration, Figure 3(b) the results after 30 iterations, and Figure 3(c) 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, we also include renderings of the 3D shape from two different viewpoints. In the top left we display estimates of the 3D pose extracted from the current estimate of the weak perspective camera matrix $\mathbf{P}$. We also display the current estimate of the 2D AAM shape projected onto the input image as blue dots. Note that as the AAM is fit, we simultaneously estimate the 3D shape (white mesh),



(a) Initialization (b) After 30 Iterations (c) Converged

Figure 3: An example of our algorithm fitting to a single image. We display the 3D shape estimate (white) projected onto the original image and also from a couple of other viewpoints (top right). We also display the 2D AAM shape (blue dots.)



(a) Frame 1 (b) Frame 2

(c) Frame 3 (d) Frame 4

Figure 4: The results of using our algorithm to track a face in a 180 frame video sequence by fitting the model to each frame.

the 2D shape (blue dots), and the camera matrix/3D pose (top left). In the example in Figure 3, we start the AAM relatively far from the correct solution and so a relatively large number of iterations are required. Averaged across all frames in a typical sequence, only 6 iterations per image were required for convergence. In Figure 4 we include 4 frames of our algorithm being used to track a face in a 180 frame video by fitting the model successively to each frame.

A useful feature of the "2D+3D AAM" is the ability to render the 3D model from novel viewpoints. Figure 5 shows an example image and the 2D+3D AAM fit result in the top row. The bottom left shows the 3D shape and appearance reconstruction from the model parameters. The bottom right shows model reconstructions from two new viewpoints. Note the sides of the face appear flat. The current mesh used to model the face does not include any points on the cheeks (there are no reliable landmarks for hand placement) so there is no depth information there.

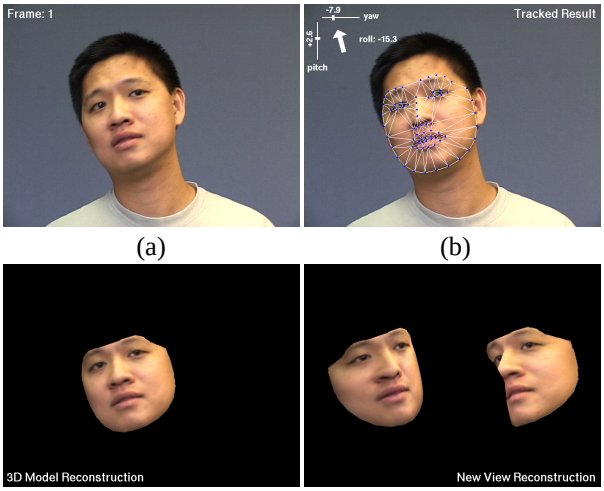
Finally we compare the fitting speed of 2D+3D AAMs with that of 2D AAMs. See Table 1. Our 2D AAM fitting algorithm [8] operates at 4.9 frames per second in Matlab and over 200 frames per second in C, both on a 3GHz Dual Pentium 4. Currently we have only had time to implement the 2D+3D AAM algorithm in Matlab. In Matlab the new algorithm runs at 6.1 frames per second. Note however, that the time per iteration for the 2D algorithm is approximately 20% less than for the 2D+3D algorithm but it requires more iterations to reach the same level of convergence. Assuming the improvement in fitting speed will be the same in C (there is no reason to suspect otherwise, the new code is very similar in style to the old), the C implementation of the 2D+3D algorithm should run at well over 250 frames per second (and faster than the 2D code.)

5 Conclusion

5.1 Summary

In Section 3 we compared the shape representational power of 2 of the most popular face models: Active Appearance Models (AAMs) [5] and 3D Morphable Models (3DMMs) [2]. Even though AAMs are 2D whereas 3DMMs are 3D, we showed that AAMs can represent any phenomena that 3DMMs can, albeit possibly at the expense of requiring up to 6 times as many shape parameters. Because they, in general, have more shape parameters, AAMs can generate many model instances that are not possible with the corresponding 3DMMs. One interpretation of this fact is that the AAM has too much representational power and can generate "impossible" instances that are not physically realizable.

In Section 4 we first showed how to compute the equivalent 3D shape modes of a 2D AAM. We used a linear non-rigid structure-from-motion algorithm [12] that does not suffer from the local minima that other non-linear algorithms do. We then showed how the 3D shape modes can be used to impose constraints on the 2D AAM parameters that force it to generate only model instances that can also be generated with the 3D shape modes. Finally, we extended our real-time AAM fitting algorithm [8] to impose these



(a) (b) (c) (d)

Figure 5: 2D+3D AAM model reconstruction. (a) shows the input image, (b) shows the tracked result, (c) the 2D+3D AAM model reconstruction and (d) shows two new view reconstructions.

constraints. While fitting, this algorithm: (1) ensures that the model instance is realizable with the 3D shape modes, and (2) explicitly recovers the 3D pose and 3D shape of the face. Combining these steps, we have extended AAMs to what we call "Combined 2D+3D AAMs."

5.2 Discussion and Future Work

We have tried to combine the best features of AAMs and 3DMMs: real-time fitting (AAMs) and a parameterization consisting of a camera matrix (including 3D pose) and 3D shape (3DMMs). In particular, we started with a AAM and computed 3D shape modes. It is also possible to start with a 3DMM, compute 2D shape modes, and then fit the 2D AAM in real-time while imposing the equivalent constraints that the 2D AAM instance is a valid 3DMM instance.

In constraining an AAM with the corresponding 3D shape modes, we increased the number of parameters. Although somewhat counter-intuitive, increasing the number of parameters in this way actually reduces the flexibility of the model because the AAM parameters and the 3D shape parameters are tightly coupled. We have presented results which show that this reduced flexibility can lead to faster convergence. More experiments are needed and we plan to quantitatively compare the robustness and speed of fitting 2D AAMs and 2D+3D AAMs in a future paper.

We have not discussed occlusion in this paper so that we can focus on the dimensionality of the shape model. The treatment of occlusion is another major difference between AAMs and 3DMMs. AAMs do not model occlusion whereas 3DMMs do. Note, however, that once we have an explicit 3D shape and pose it is relatively straight-forward to model the self occlusion of the model. In future papers, we plan to extend our real-time 2D+3D AAM fitting algorithm to cope with both self and other forms of occlusion.

Table 1: Fitting speed on a 3GHz Dual Pentium 4 Xeon. These results show number of frames per second (fps) or number of iterations per second (ips) for an AAM with 17 2D shape parameters, 42 appearance parameters, and 30,000 color pixels. The 2D+3D AAM has an extra 15 3D shape and 5 camera parameters.

	2D AAM Fitting	2D+3D AAM Fitting
Matlab	87 ips	71 ips
Matlab	4.9 fps	6.1 fps
C	230 fps	≈286 fps (est.)

Acknowledgments

The research described in this paper was conducted under U.S. Department of Defense contract N41756-03-C4024.

References

[1] S. Baker, R. Gross, and I. Matthews. Lucas-Kanade 20 years on: A unifying framework: Part 4. Technical Report CMU-RI-TR-04-14, Robotics Institute, Carnegie Mellon University, 2004.

[2] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In *SIGGRAPH*, 1999.

[3] M. Brand. Morphable 3D models from video. In *Proceedings of CVPR*, 2001.

[4] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3D shape from image streams. In *Proceedings of CVPR*, 2000.

[5] T.F. Cootes, G.J. Edwards, and C.J. Taylor. Active appearance models. *PAMI*, 23(6):681–685, June 2001.

[6] G. Hager and P. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *PAMI*, 20:1025–1039, 1998.

[7] A. Lanitis, C. Taylor, and T. Cootes. Automatic interpretation and coding of face images using flexible models. *PAMI*, 19(7):742–756, 1997.

[8] I. Matthews and S. Baker. Active Appearance Models revisited. *IJCV*, 2004. In Press.

[9] S. Romdhani and T. Vetter. Efficient, robust and accurate fitting of a 3D morphable model. In *ICCV*, 2003.

[10] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *IJCV*, 9(2):137–154, 1992.

[11] L. Torresani, D. Yang, G. Alexander, and C. Bregler. Tracking and modeling non-rigid objects with rank constraints. In *CVPR*, 2001.

[12] J. Xiao, J. Chai, and T. Kanade. A closed-form solution to non-rigid shape and motion recovery. In *ECCV*, 2004.