

Real-Time Non-Rigid Driver Head Tracking for Driver Mental State Estimation

CMU-RI-TR-04-10

**Simon Baker, Iain Matthews, Jing Xiao,
Ralph Gross, Takahiro Ishikawa, and Takeo Kanade**

The Robotics Institute
Carnegie Mellon University

Abstract

The non-rigid motion of a driver's head (i.e. the motion of their mouth, eye-brows, cheeks, etc) can tell us a lot about their mental state; e.g. whether they are drowsy, alert, aggressive, comfortable, tense, distracted, etc. In this paper, we describe our recent research on non-rigid face tracking. In particular, we present both 2D and 3D algorithms for tracking the non-rigid motion of the driver's head using an Active Appearance Model. Both algorithms operate at over 200 frames per second. We also present algorithms for converting a 2D model into a 3D model and for fitting with occlusion and large pose variation.

1 Introduction

Tracking the driver's head has a wide variety of applications and is one of the most important tasks for an intelligent vehicle. Two types of head tracking can be distinguished: *rigid* and *non-rigid*. In rigid tracking such as [7, 9], the head is tracked as a rigid object with 6 degrees of freedom, 3 translations and 3 rotations. In non-rigid tracking, the motion of the mouth, eye-brows, cheeks, etc, (i.e. the facial expression) are also tracked independently of the motion of the head itself. Non-rigid tracking is a more general, and far harder, problem than rigid tracking. It also has a number of applications that are not possible with rigid tracking alone. In particular, the non-rigid motion

of the driver's face can tell us a lot about their mental state; i.e. whether they are drowsy, alert, aggressive, comfortable, tense, distracted, etc.

In this paper, we describe our recent work on non-rigid face tracking. In particular, we have been working with Active Appearance Models (AAMs) [5]. AAMs are a well studied face model consisting of separate linear shape and appearance components. 2D AAMs can model both the rigid motion of the head and the non-rigid motion of the facial features. The specific problem we have addressed is the fitting problem. Tracking a driver's head with an AAM requires fitting the AAM to each frame in turn in the video sequence. The fitting algorithm therefore needs to be: (1) fast enough to operate in real-time and (2) robust enough so as not to lose track of the head. Our main contribution is an analytically derived gradient descent fitting algorithm [10]. Because our algorithm is derived analytically, it is far more robust than previous algorithms which are based on numerical approximations [5]. It is also far faster, operating at over 200 frames per second on a standard 3GHz PC, leaving plenty of time for all of the other processing needed in a complete end-to-end system, as well as the possibility of implementation on low-power devices. Finally, this fitting algorithm can be extended to cope with occlusion, such as that caused by large head rotation [6].

One limitation of standard AAMs is that they are 2D. Compared to 3D face models, such as 3D Morphable Models (3DMM) [2], 2D AAMs are at a serious disadvantage. With a 2D AAM, the 6 degree of freedom rigid motion is combined with the non-rigid facial expression into a complex combined 2D motion model. Separating the rigid motion from the non-rigid is far harder with a 2D AAM than with a 3DMM. Moreover, coping with large pose variation, which inevitably leads to self occlusion, is straightforward with a 3D model, but very difficult with a 2D model. The main problem with 3D models, however, is that they are very hard to fit. The fastest "efficient" algorithm runs at around 30 seconds per frame [11]; i.e. 900 times slower than real-time (30 frames per second.) Another difficulty with 3D models is that a large amount of 3D range data is usually required to build the model. Obtaining such data is difficult.

In the second half of this paper we describe our work on 3D non-rigid face tracking. We

have made two main contributions in this area: (1) we have developed a non-rigid structure-from-motion algorithm [15] that can be used to convert a 2D AAM into a 3D face model, and (2) we have developed an algorithm for fitting the resulting 3D model which runs at over 250 frames per second [14] (remarkably, this algorithm is even faster than the 2D algorithm).

2 2D Driver Non-Rigid Head Tracking

2.1 2D Active Appearance Models

Active Appearance Models (AAMs) [5] are generative face models. A 2D AAM consists of two components, the shape and the appearance. 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]. The Iterative Procrustes Algorithm and Principal Component Analysis are then applied to compute the the base shape \mathbf{s}_0 and the shape variation \mathbf{s}_i . An example of the base shape \mathbf{s}_0 and the first two shape modes (\mathbf{s}_1 and \mathbf{s}_2) of an AAM are shown in Figure 1(a)–(c).

The *appearance* of the AAM is defined within the base mesh \mathbf{s}_0 . Let \mathbf{s}_0 also denote the set of pixels $\mathbf{u} = (u, v)^T$ that lie inside the base mesh \mathbf{s}_0 , a convenient abuse of terminology. The

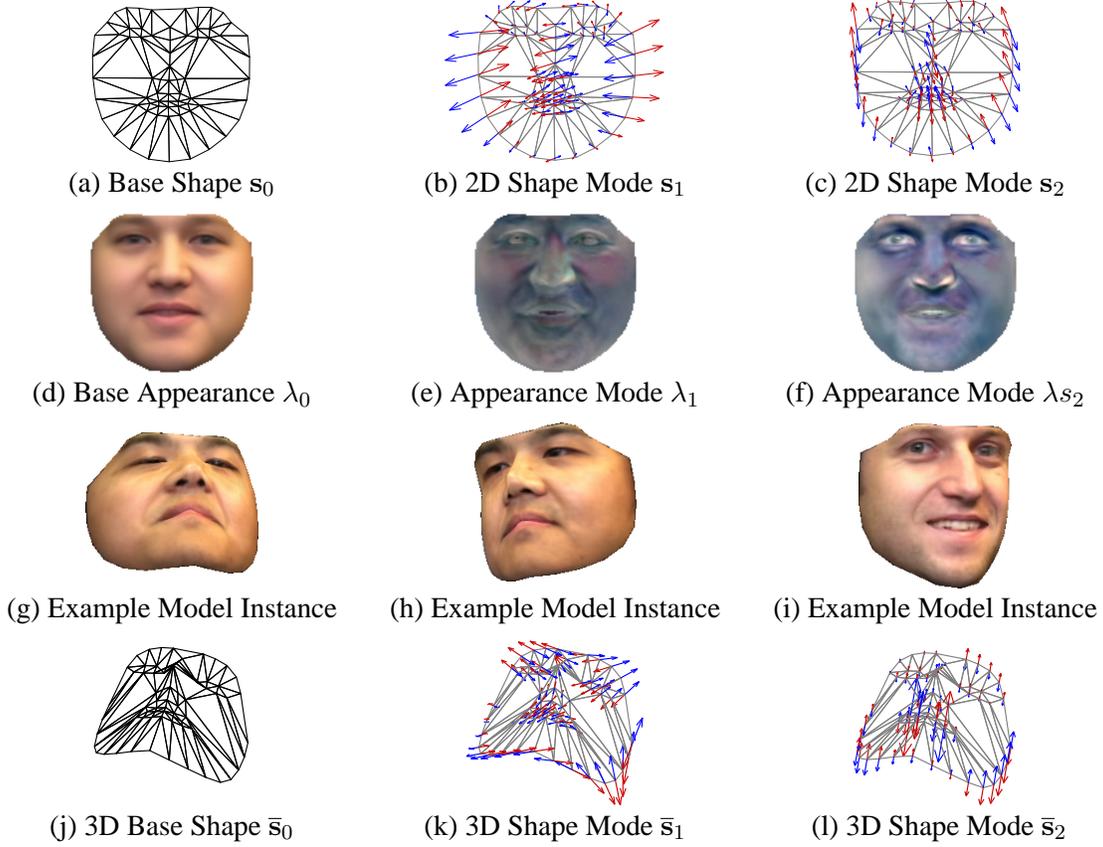


Figure 1: An example Active Appearance Model [5]. (a–c) The AAM base shape s_0 and the first two shape modes s_1 and s_2 . (d–f) The AAM base appearance λ_0 and the first two shape modes λ_1 and λ_2 . (g–i) Three example model instances demonstrating how the AAM is a generative model that generates images of faces with different poses, identities, and expressions. (j–l) The 3D shape variation.

appearance of the AAM is then an image $A(\mathbf{u})$ defined over the pixels $\mathbf{u} \in s_0$. 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]. An example of the base appearance λ_0 and the first two appearance modes (λ_1 and λ_2) of an AAM are shown in Figures 1(d)–(e).

Although Equations (2) and (3) describe the AAM shape and appearance variation, they do not

describe how to generate a *model instance*. The AAM model instance with shape parameters \mathbf{p} and appearance parameters λ_i is created by warping the appearance A from the base mesh \mathbf{s}_0 to the model shape mesh \mathbf{s} . In particular, the pair of meshes \mathbf{s}_0 and \mathbf{s} define a piecewise affine warp from \mathbf{s}_0 to \mathbf{s} which we denote $\mathbf{W}(\mathbf{u}; \mathbf{p})$. Three example model instances are included in Figures 1(g)–(i). This figure demonstrate the generative power of an AAM. The AAM can generate face images with different poses (Figures 1(g) and (h)), different identities (Figures 1(g) and (i)), and different non-rigid face motion or expression, (Figures 1(h) and (i)).

2.2 Real-Time Non-Rigid 2D Tracking with AAMs

Driver head tracking is performed by “fitting” the AAM sequentially to each frame in the input video. Given an input image I , the goal of AAM fitting is to minimize:

$$\sum_{\mathbf{u} \in \mathbf{s}_0} \left[A_0(\mathbf{u}) + \sum_{i=1}^l \lambda_i A_i(\mathbf{u}) - I(\mathbf{W}(\mathbf{u}; \mathbf{p})) \right]^2 = \left\| A_0(\mathbf{u}) + \sum_{i=1}^l \lambda_i A_i(\mathbf{u}) - I(\mathbf{W}(\mathbf{u}; \mathbf{p})) \right\|^2 \quad (4)$$

simultaneously with respect to the 2D AAM shape p_i and appearance λ_i parameters. In [10] we proposed an algorithm to minimize the expression in Equation (4) that operates at around 230 frames per second. This algorithm relies on a sequence of mathematical simplifications.

The first step is to separate the shape and appearance components. If we denote the linear subspace spanned by a collection of vectors A_i by $\text{sub}(A_i)$ and its orthogonal complement by $\text{sub}(A_i)^\perp$, the right hand side of Equation (4) can be rewritten as:

$$\left\| A_0(\mathbf{u}) + \sum_{i=1}^m \lambda_i A_i(\mathbf{u}) - I(\mathbf{W}(\mathbf{u}; \mathbf{p})) \right\|_{\text{sub}(A_i)^\perp}^2 + \left\| A_0(\mathbf{u}) + \sum_{i=1}^m \lambda_i A_i(\mathbf{u}) - I(\mathbf{W}(\mathbf{u}; \mathbf{p})) \right\|_{\text{sub}(A_i)}^2 \quad (5)$$

where $\| \cdot \|_L^2$ denotes the square of the L2 norm of the vector projected into the linear subspace L . The first of the two terms immediately simplifies. Since the norm only considers the components of vectors in the orthogonal complement of $\text{sub}(A_i)$, any component in $\text{sub}(A_i)$ itself can be dropped.

We therefore wish to minimize:

$$\|A_0(\mathbf{u}) - I(\mathbf{W}(\mathbf{u}; \mathbf{p}))\|_{\text{sub}(A_i)^\perp}^2 + \left\| A_0(\mathbf{u}) + \sum_{i=1}^m \lambda_i A_i(\mathbf{u}) - I(\mathbf{W}(\mathbf{u}; \mathbf{p})) \right\|_{\text{sub}(A_i)}^2. \quad (6)$$

The first of these two terms does not depend upon λ_i . For any \mathbf{p} , the minimum value of the second term is always 0. Therefore the minimum value can be found sequentially by first minimizing the first term with respect to \mathbf{p} alone, and then using that optimal value of \mathbf{p} as a constant to minimize the second term with respect to the λ_i . Assuming that the basis vectors A_i are orthonormal, the second minimization has a simple closed-form solution:

$$\lambda_i = \sum_{\mathbf{u} \in \mathbf{s}_0} A_i(\mathbf{u}) \cdot [I(\mathbf{W}(\mathbf{u}; \mathbf{p})) - A_0(\mathbf{u})], \quad (7)$$

the dot product of A_i with the final error image obtained after doing the first minimization.

The second step is to minimize the first term in Equation (6). This can be performed efficiently using the *inverse compositional algorithm* [1]. The usual *forwards additive Lucas-Kanade algorithm* minimizes the expression in Equation (6) by iteratively minimizing:

$$\|A_0(\mathbf{u}) - I(\mathbf{W}(\mathbf{u}; \mathbf{p} + \Delta\mathbf{p}))\|_{\text{sub}(A_i)^\perp}^2 \quad (8)$$

with respect to $\Delta\mathbf{p}$ and updates the parameters $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$. On the other hand, the inverse compositional algorithm iteratively minimizes:

$$\|A_0(\mathbf{W}(\mathbf{u}; \Delta\mathbf{p}) - I(\mathbf{W}(\mathbf{u}; \mathbf{p}))\|_{\text{sub}(A_i)^\perp}^2 \quad (9)$$

with respect to $\Delta\mathbf{p}$ and updates the warp $\mathbf{W}(\mathbf{u}; \mathbf{p}) \leftarrow \mathbf{W}(\mathbf{u}; \mathbf{p}) \circ \mathbf{W}(\mathbf{u}; \Delta\mathbf{p})^{-1}$. In [1] we proved that the inverse compositional algorithm and the forwards additive (Lucas-Kanade) algorithm are equivalent. The advantage of using the inverse compositional algorithm is that the Gauss-Newton

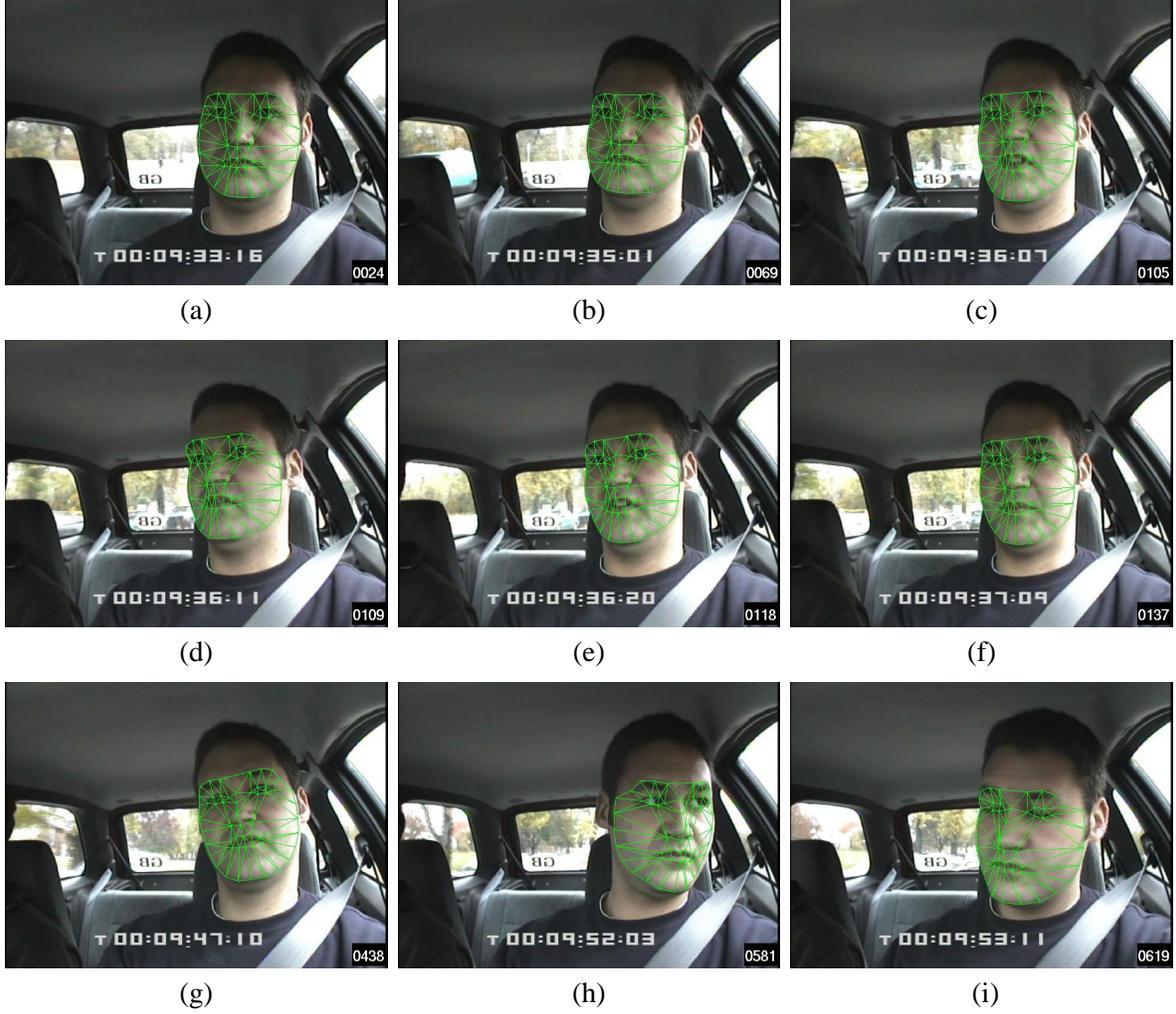


Figure 2: Driver head tracking results with a 2D AAM. Notice how the AAM tracks both the rigid motion of the head (f-i) and the non-rigid motion of the mouth (b-e). Figure best viewed in color.

approximation for $\Delta \mathbf{p}$ is:

$$\Delta \mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{u}} \left[\nabla_{A_0} \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]_{\text{sub}(A_i)^\perp}^T [I(\mathbf{W}(\mathbf{u}; \mathbf{p})) - A_0(\mathbf{u})] \quad (10)$$

where:

$$\mathbf{H} = \sum_{\mathbf{u}} \left[\nabla_{A_0} \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]_{\text{sub}(A_i)^\perp}^T \left[\nabla_{A_0} \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]_{\text{sub}(A_i)^\perp}. \quad (11)$$

In this expression, both the steepest descent images $[\nabla_{A_0} \frac{\partial \mathbf{W}}{\partial \mathbf{p}}]_{\text{sub}(A_i)^\perp}$ and the Hessian \mathbf{H} are constant and so can be precomputed. This results in a substantial computational saving, and means that the algorithm can be implemented in real time. See [1, 10] for more details.

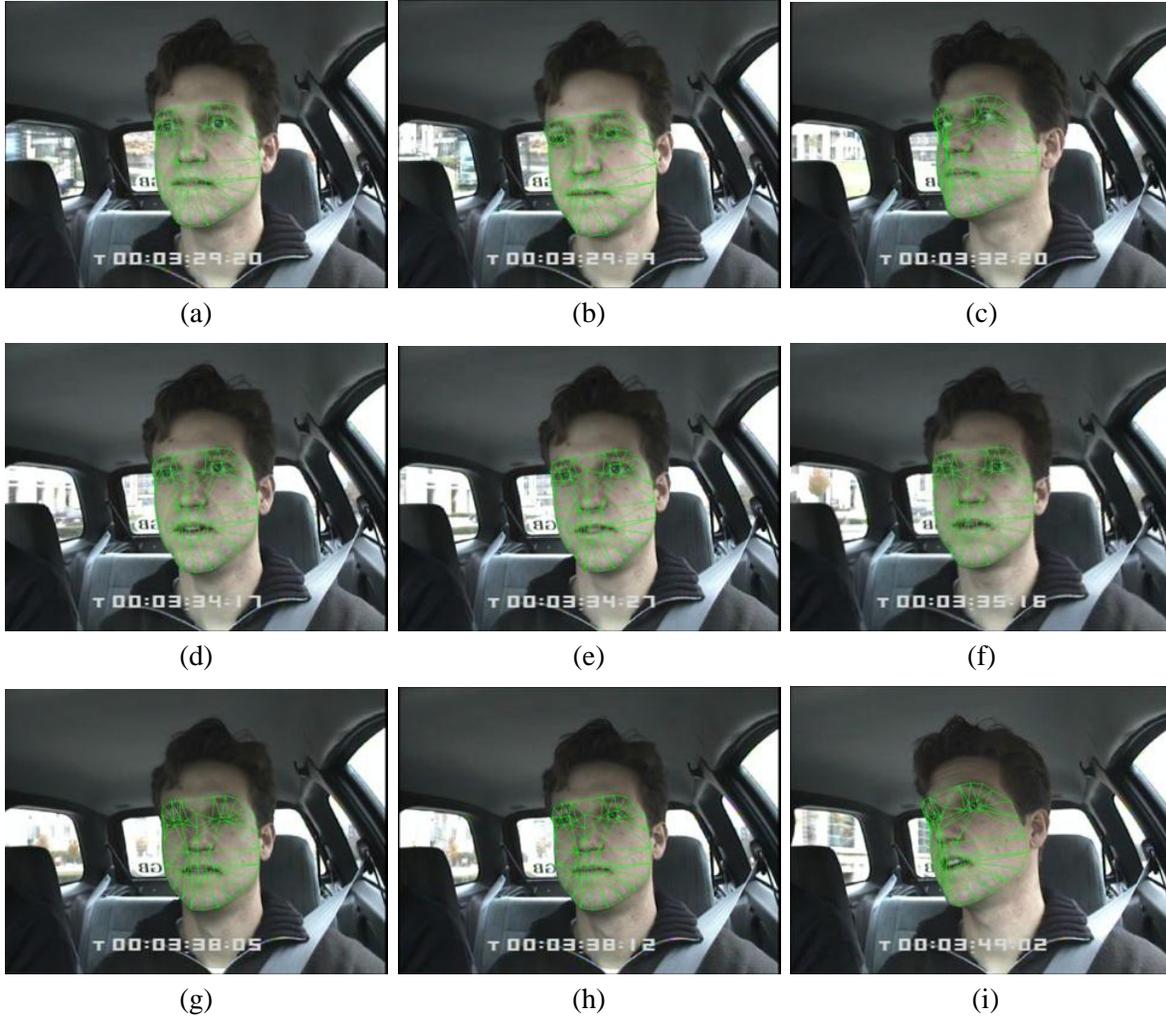


Figure 3: Driver head tracking results with a 2D AAM. Notice how the AAM tracks both the rigid motion of the head (a–c), the non-rigid motion of the mouth (d–e), and the blink (f–h). View in color.

2.2.1 Experimental Results

In [10] we quantitatively compared our algorithm with three variants of the original AAM fitting algorithm and showed it to be both faster and more robust [5]. In Figures 2 and 3 we include a number of frames from two sequences of a 2D AAM being used to track two different drivers faces. (These figures are best viewed in color.) Note how the AAM tracks both the rigid motion of the driver’s head as well as the non-rigid motion of their mouth and eyes.

2.3 Real-Time Non-Rigid Tracking With Occlusion

When fitting a head model, it is important to be able to cope with occlusion. Occlusion is inevitable, for example, when the driver puts their hand in front of their face, or when they turn their head to look in their “blind-spot”, a one of the passengers, etc. We have extended our real-time AAM fitting algorithm to cope with occlusion [6]. The description in this paper is in terms of a 2D AAM, however it is perhaps more relevant, and also straightforward to use this algorithm with the 3D face models described in the following section.

Occluded pixels in the input image can be viewed as “outliers”. In order to deal with outliers in a least-squares optimization framework a robust error function can be used [8]. The goal for *robustly* fitting a 2D AAM is then to minimize:

$$\sum_{\mathbf{u} \in \mathbf{s}_0} \varrho \left(\left[A_0(\mathbf{u}) + \sum_{i=1}^m \lambda_i A_i(\mathbf{u}) - I(\mathbf{W}(\mathbf{u}; \mathbf{p})) \right]^2 \right) \quad (12)$$

with respect to the shape \mathbf{p} and appearance $\boldsymbol{\lambda}$ parameters where $\varrho(t)$ is a symmetric *robust error function* [8]. In comparison to the non-robust described algorithm above, the expressions for the incremental parameter update $\Delta \mathbf{p}$ (Equation 10) and the Hessian H (Equation 11) have to be weighted by the error function $\varrho'(E(\mathbf{u})^2)$, where:

$$E(\mathbf{u}) = A_0(\mathbf{u}) + \sum_{i=1}^m \lambda_i A_i(\mathbf{u}) - I(\mathbf{W}(\mathbf{u}; \mathbf{p})) \quad (13)$$

Equation (10) then becomes:

$$\Delta \mathbf{p} = -H_\rho^{-1} \sum_{\mathbf{u} \in \mathbf{s}_0} \varrho'(E(\mathbf{u})^2) \left[\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T E(\mathbf{u}) \quad (14)$$

and Equation (11) becomes:

$$H_\rho = \sum_{\mathbf{u} \in \mathbf{s}_0} \varrho'(E(\mathbf{u})^2) \left[\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[\nabla A_0 \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]. \quad (15)$$

The appearance variation must also be treated differently. See [6] for an explanation of why. Instead of being solved sequentially, as described above, it must be solved for in every iteration of the algorithm. The incremental additive updates to the appearance parameters $\Delta\boldsymbol{\lambda} = (\Delta\lambda_1, \dots, \Delta\lambda_m)^T$ are computed by minimizing:

$$\sum_{\mathbf{u}} \varrho' \left(E(\mathbf{u})^2 \right) \left[E(\mathbf{u}) + \sum_{i=1}^m \Delta\lambda_i A_i(\mathbf{u}) \right]^2. \quad (16)$$

The least squares minimum of this expression is:

$$\Delta\boldsymbol{\lambda} = -H_{\mathbf{A}}^{-1} \sum_{\mathbf{u}} \varrho' \left(E(\mathbf{u})^2 \right) \mathbf{A}^T(\mathbf{u}) E(\mathbf{u}) \quad (17)$$

where $\mathbf{A}(\mathbf{u}) = (A_1(\mathbf{u}), \dots, A_m(\mathbf{u}))$ and $H_{\mathbf{A}}$ is the appearance Hessian:

$$H_{\mathbf{A}} = \sum_{\mathbf{u}} \varrho' \left(E(\mathbf{u})^2 \right) \mathbf{A}(\mathbf{u})^T \mathbf{A}(\mathbf{u}). \quad (18)$$

In order to retain the efficiency of the original algorithm, a variety of approximations need to be made to speed up the computation of the two Hessians H_{ρ} and $H_{\mathbf{A}}$. See [6] for the details.

2.3.1 Experimental Results

In Figure 4 we include a few frames of our algorithm being used to track a person's face in the presence of occlusion. The figure contains examples of both self occlusion caused by head rotation, and occlusion caused by other objects. In some of the cases almost half the face is occluded.

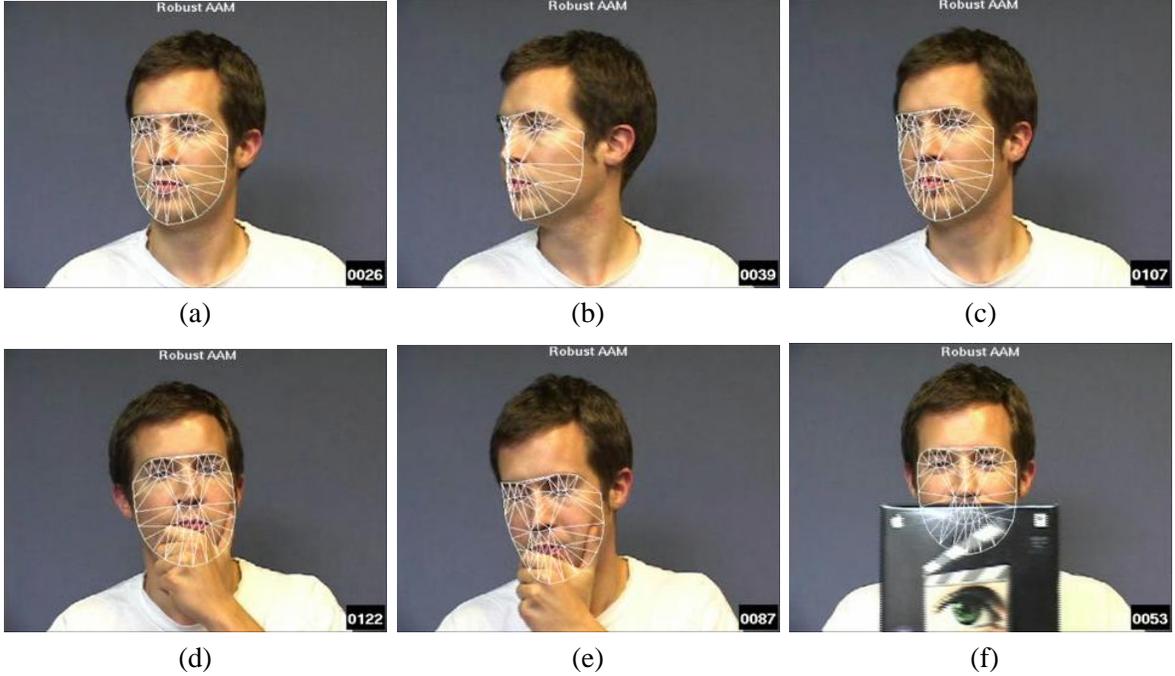


Figure 4: Tracking with occlusion. Our algorithm is robust to occlusion caused by pose variation (a–c), the person’s hand (d–e), and other objects (f). Note that the algorithm tracks the eye-blink in (e).

3 3D Driver Non-Rigid Head Tracking

3.1 3D Active Appearance Models

The shape of a 2D AAM is 2D which makes driver head pose estimation difficult. On the other hand, 3D Active Appearance Models (3D AAM) [14] have a 3D linear shape model:

$$\bar{\mathbf{s}} = \bar{\mathbf{s}}_0 + \sum_{i=1}^{\bar{m}} \bar{p}_i \bar{\mathbf{s}}_i \quad (19)$$

where the coefficients \bar{p}_i are the 3D shape parameters and $\bar{\mathbf{s}}$ and $\bar{\mathbf{s}}_i$ are the 3D coordinates:

$$\bar{\mathbf{s}} = \begin{pmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \\ z_1 & z_2 & \dots & z_n \end{pmatrix}. \quad (20)$$

To fit such a 3D model to a 2D image, we need an image formation model. We use the weak perspective imaging model defined by:

$$\mathbf{u} = \mathbf{P} \mathbf{x} = \begin{pmatrix} i_x & i_y & i_z \\ j_x & j_y & j_z \end{pmatrix} \mathbf{x} + \begin{pmatrix} o_x \\ o_y \end{pmatrix}. \quad (21)$$

where (o_x, o_y) is an offset to the origin and the projection axes $\mathbf{i} = (i_x, i_y, i_z)$ and $\mathbf{j} = (j_x, j_y, j_z)$ are equal length and orthogonal: $\mathbf{i} \cdot \mathbf{i} = \mathbf{j} \cdot \mathbf{j}$; $\mathbf{i} \cdot \mathbf{j} = 0$.

3.2 Constructing a 3D AAM from a 2D AAM

If we have a 2D AAM, a sequence of images $I^t(\mathbf{u})$ for $t = 0, \dots, N$, and have tracked the 2D AAM through the sequence, then the tracked 2D AAM shape parameters at time t are $\mathbf{p}^t = (p_1^t, \dots, p_m^t)^\top$.

Using Equation (2) we can then compute the 2D AAM shape vectors \mathbf{s}^t :

$$\mathbf{s}^t = \begin{pmatrix} u_1^t & u_2^t & \dots & u_n^t \\ v_1^t & v_2^t & \dots & v_n^t \end{pmatrix}. \quad (22)$$

A variety of non-rigid structure-from-motion algorithms have been proposed to convert the tracked feature points in Equation (22) into a 3D AAM. Bregler et al. [4] proposed a factorization method to simultaneously reconstruct the non-rigid shape and camera projection matrices. This method was extended to a trilinear optimization approach in [13]. 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 [12]. In [15] 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 [15] can be used to compute a 3D AAM from a 2D AAM. (Any of the other algorithms could be used instead, although with worse results.)

We stack the AAM shape vectors (see Equation (22)) in all N images into a measurement matrix:

$$W = \begin{pmatrix} u_1^0 & u_2^0 & \dots & u_n^0 \\ v_1^0 & v_2^0 & \dots & v_n^0 \\ \vdots & \vdots & \vdots & \vdots \\ u_1^N & u_2^N & \dots & u_n^N \\ v_1^N & v_2^N & \dots & v_n^N \end{pmatrix}. \quad (23)$$

If this measurement data can be explained by a 3D AAM, then the matrix W can be represented:

$$W = MB = \begin{pmatrix} \mathbf{P}^0 & p_1^0 \mathbf{P}^0 & \dots & p_{\bar{m}}^0 \mathbf{P}^0 \\ \mathbf{P}^1 & p_1^1 \mathbf{P}^1 & \dots & p_{\bar{m}}^1 \mathbf{P}^1 \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{P}^N & p_1^N \mathbf{P}^N & \dots & p_{\bar{m}}^N \mathbf{P}^N \end{pmatrix} \begin{pmatrix} \bar{s}_0 \\ \vdots \\ \bar{s}_{\bar{m}} \end{pmatrix} \quad (24)$$

where M is a $2(N+1) \times 3(\bar{m}+1)$ scaled projection matrix and B is a $3(\bar{m}+1) \times n$ shape matrix (setting the number of 3D AAM vertices \bar{n} to equal the number of 2D vertices n .) Since \bar{m} is usually small, the rank of W is at most $3(\bar{m}+1)$.

We perform a Singular Value Decomposition (SVD) on W and factorize it into the product of a $2(N+1) \times 3(\bar{m}+1)$ matrix \tilde{M} and a $3(\bar{m}+1) \times n$ matrix \tilde{B} . This decomposition is not unique, and is only determined up to a linear transformation. Any non-singular $3(\bar{m}+1) \times 3(\bar{m}+1)$ 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:

$$M = \tilde{M} \cdot G, \quad B = G^{-1} \cdot \tilde{B} \quad (25)$$

where G is the corrective matrix. See [15] for the details of how to compute G using orthonormality and basis constraints. Once G has been determined, M and B can be recovered. In summary, the 3D AAM shape variation \bar{s}_i is automatically computed from the 2D AAM shape variation s_i and

the 2D AAM tracking results. An example of the 3D base shape \bar{s}_0 and the first two 3D shape modes (\bar{s}_1 and \bar{s}_2) of the AAM in Figure 1 are shown in Figure 1(j)–(l).

3.3 Real-Time Non-Rigid 3D Tracking

We perform 3D tracking (i.e. 3D AAM fitting) by minimizing:

$$\left\| A_0(\mathbf{u}) + \sum_{i=1}^l \lambda_i A_i(\mathbf{u}) - I(\mathbf{W}(\mathbf{u}; \mathbf{p})) \right\|^2 + K \cdot \left\| \mathbf{s}_0 + \sum_{i=1}^m p_i \mathbf{s}_i - \mathbf{P} \left(\bar{\mathbf{s}}_0 + \sum_{i=1}^{\bar{m}} \bar{p}_i \bar{\mathbf{s}}_i \right) \right\|^2 \quad (26)$$

simultaneously with respect to p_i , λ_i , \mathbf{P} , and \bar{p}_i , where K is a large constant weight. Equation (26) should be compared to Equation (4) and can be interpreted as follows. The first term is the 2D AAM fitting criterion; i.e. Equation (4). The second term enforces the (heavily weighted soft) constraints that the 2D shape \mathbf{s} equals the projection of the 3D shape $\bar{\mathbf{s}}$ with projection matrix \mathbf{P} . Once the expression in Equation (26) has been minimized, the driver head pose can be extracted from the weak perspective projection matrix \mathbf{P} and the 3D non-rigid shape variation is contained in the 3D shape parameters \bar{p}_i . In [14] we extended our 2D AAM fitting algorithm [10] to minimize the expression in Equation (26). Although somewhat more complex, our 3D algorithm operates at around 286Hz, about 20% faster than the 2D algorithm [14]. This speed up is due to a reduction of about 40% in the number of iterations required because of the additional constraints imposed by the 3D model. This more than compensates for the approximately 20% increase in the computational time per iteration.

3.3.1 Experimental Results

In Figures 5 and 6 we include a few frames of our algorithm being used to track the faces of two people. The main points to note are: (1) the tracking is real-time (over 250 frames per second), (2) the 3D head pose (motion) of the head is recovered explicitly (displayed in the top left of each frame), and (3) the 3D non-rigid motion of the face is captured independently (the recovered 3D shape is shown from two viewpoints in the top right.) Note how our algorithm recovers both the

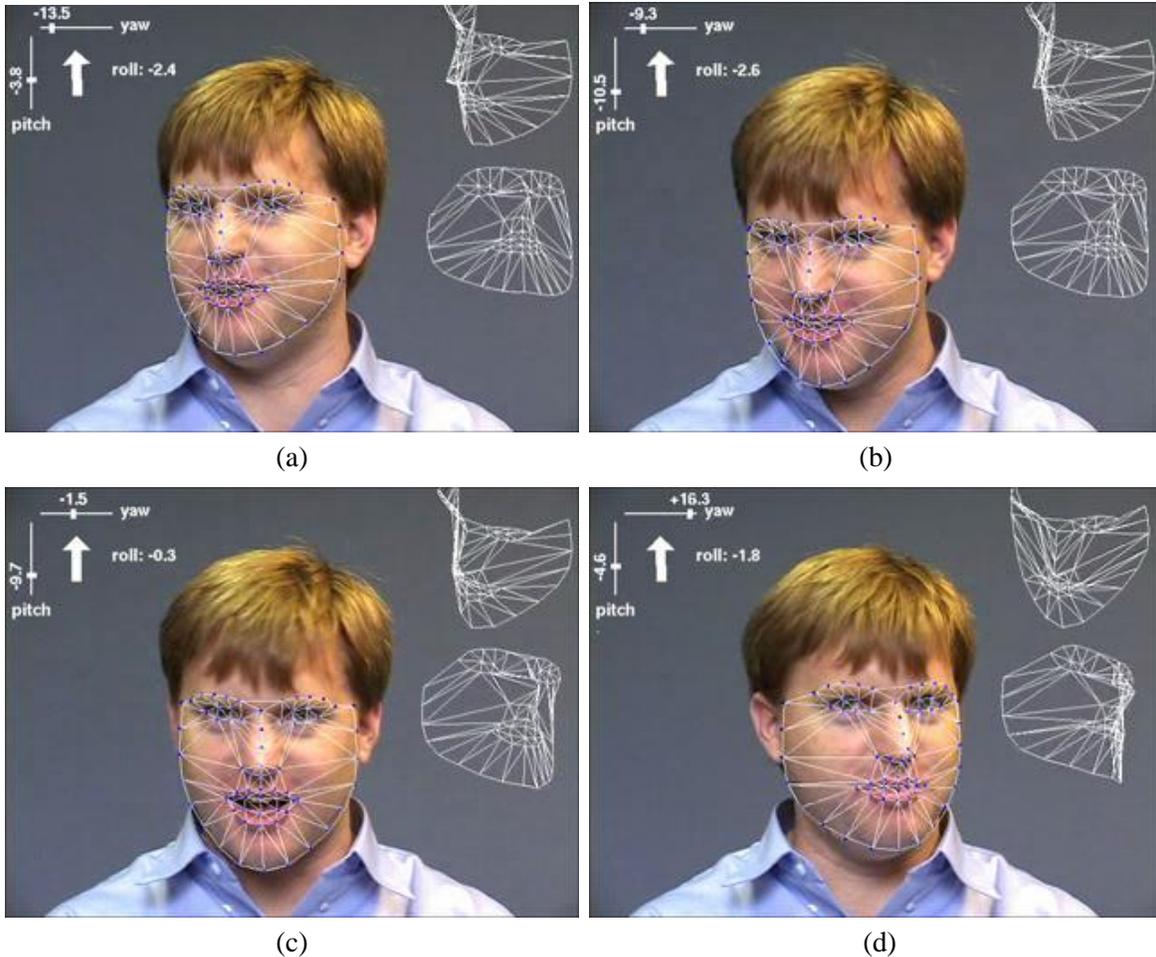


Figure 5: 3D head tracking. Our real-time non-rigid tracking algorithm estimates the 3D pose of the head (top left of each frame) and the non-rigid 3D motion of the face (top right of each frame).

3D head motion, and the non-rigid motion of the face, in particular the opening of the mouth and the blinking of the eyes.

4 Conclusion

In this paper we have described our recent non-rigid face tracking research using AAMs. We have described four algorithms: (1) a real-time (over 200 frames per second) 2D AAM fitting algorithm, (2) a real-time robust extension to this algorithm, (3) an algorithm for building a 3D non-rigid face model from a 2D AAM, and (4) a real-time (over 250 frames per second) algorithm for fitting the 3D non-rigid model. Our real-time algorithms allow the efficient implementation of a promising

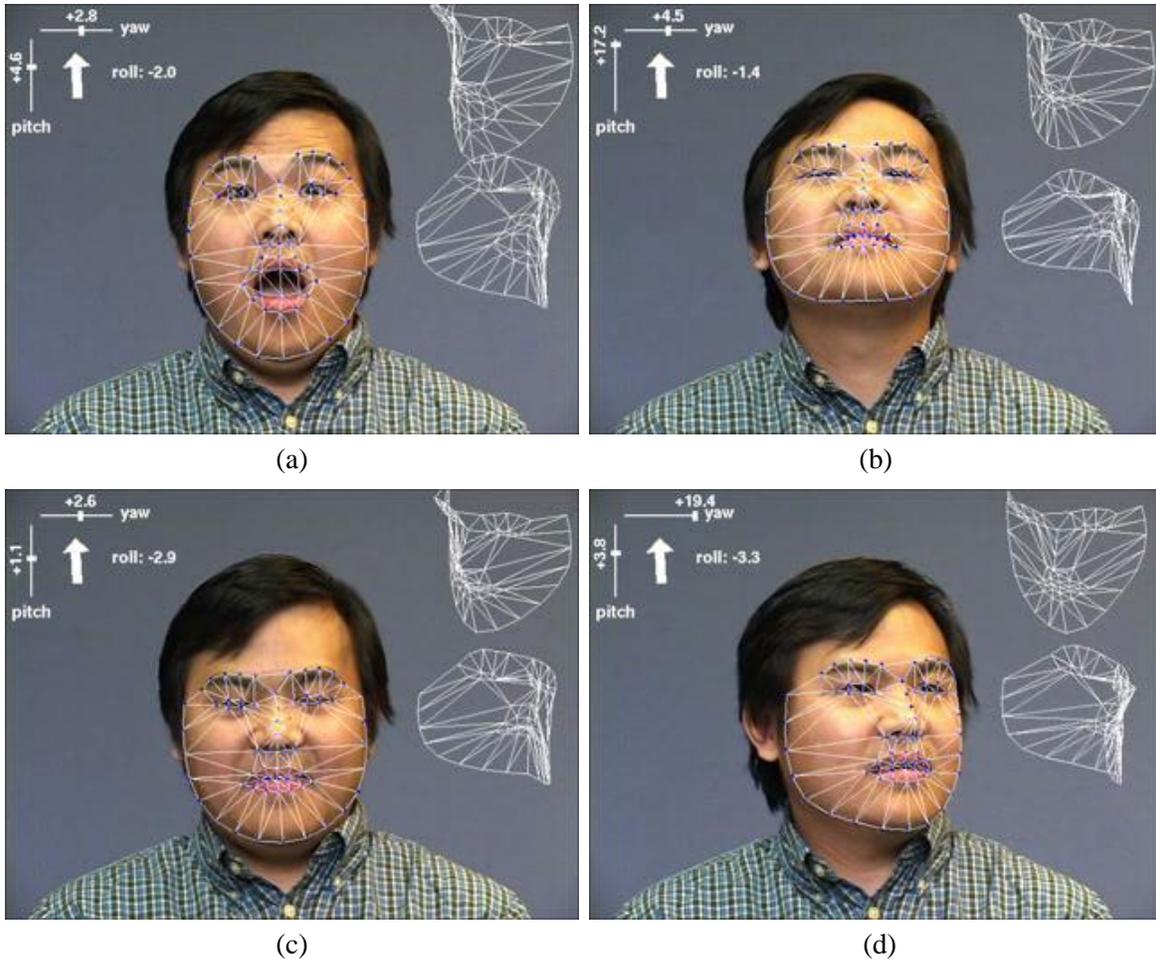


Figure 6: 3D head tracking. Our real-time non-rigid tracking algorithm estimates the 3D pose of the head (top left of each frame) and the non-rigid 3D motion of the face (top right of each frame).

and well studied technology (AAMs) on standard PCs. They are so fast that implementation on low-power devices is also be possible.

Acknowledgments

The research described in this paper was supported by Denso Corporation, Japan. Partial support was also provided by the U.S. Department of Defense under contract N41756-03-C4024.

References

- [1] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–254, 2004.
- [2] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In *Proceedings of Computer Graphics, Annual Conference Series (SIGGRAPH)*, pages 187–194, 1999.
- [3] M. Brand. Morphable 3D models from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2001.
- [4] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3D shape from image streams. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2000.
- [5] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, June 2001.
- [6] R. Gross, I. Matthews, and S. Baker. Constructing and fitting Active Appearance Models with occlusion. In *Submitted to the IEEE Conference on Computer Vision and Pattern Recognition*, 2004.
- [7] J. Heinzmann and A. Zelinsky. 3-D facial pose and gaze point estimation using a robust real-time tracking paradigm. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, pages 142–147, 1998.
- [8] P.J. Huber. *Robust Statistics*. Wiley & Sons, 1981.
- [9] Y. Matsumoto and A. Zelinsky. An algorithm for real-time stereo vision implementation of head pose and gaze direction measurement. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, pages 499–505, 2000.

- [10] I. Matthews and S. Baker. Active Appearance Models revisited. *International Journal of Computer Vision*, 2004. (Accepted subject to minor revisions, Previously appeared as CMU Robotics Institute Technical Report CMU-RI-TR-03-02).
- [11] S. Romdhani and T. Vetter. Efficient, robust and accurate fitting of a 3D Morphable Model. In *Proceedings of the International Conference on Computer Vision*, 2003.
- [12] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.
- [13] L. Torresani, D. Yang, G. Alexander, and C. Bregler. Tracking and modeling non-rigid objects with rank constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2001.
- [14] J. Xiao, S. Baker, I. Matthews, and T. Kanade. Real-time combined 2D+3D Active Appearance Models. In *Submitted to the IEEE Conference on Computer Vision and Pattern Recognition*, 2004.
- [15] J. Xiao, J. Chai, and T. Kanade. A closed-form solution to non-rigid shape and motion recovery. In *Proceedings of the European Conference on Computer Vision*, 2004.