Face Tracking with Multilinear (Tensor) Active Appearance Models

Weiguang Si
University of California, Los Angeles
Los Angeles, CA

Kota Yamaguchi
Stony Brook University
Stony Brook, NY 11794

M. Alex. O. Vasilescu
University of California Los Angeles
Los Angeles, CA 90049
http://www.cs.ucla.edu/~maov

Abstract

Face tracking in an unconstrained environment must contend with images that vary with viewpoint, illumination, expression, identity, and other causal factors. In a statistical approach, the multifactor nature of the image data makes the aforementioned problem amenable to analysis in a multilinear framework. In this paper, we propose Multilinear (Tensor) Active Appearance Models (MAAMs). The MAAM is a multilinear statistical model of facial appearance and shape that generalizes the linear Active Appearnce Model (AAM). As models of data variability, the latter fail to distinguish and account for the different sources of variabilty. On the other hand, our MAAMs explicitly represent the underlying processes of image formation, thus preserving attributes that are relevant to the task of tracking the human face.

1. Introduction

Statistical models of shape and texture have been succesfully employed to recognize, track, and synthesize facial images. In the literature, one commonly encounters linear models, among them the standard Principal Components Analysis (PCA) model known as Eigenfaces [15, 7], which assume fixed or nearly fixed illumination and viewpoint (such as near fronto-parallel images) [2, 4, 6]. When facial image variablity departs from the linear model assumption, researchers have attempted to accommodate the observed variation by a set of locally linear models [12, 14]. Active Appearance Models (AAMs) [2] and View-Based Active Appearance Models [1] are one of the leading linear and locally linear statistical models of facial shape and texture. However, all of these models represent data variability without attempting to distinguish between the different sources of variability. Performing dimensionality reduction and computing a subspace without discriminating among the different modes of variability implicitly assumes that the same subspace is appropriate for all applications.

Face tracking in an unconstrained environment must contend with images that vary with viewpoint, illumination, expression, identity, and other causal factors. The multifactor nature of the image data makes the aforementioned problem amenable to statistical analysis in a *multilinear* framework, using multilinear (or tensor) algebra [18, 19, 17]. Multilinear algebra generalizes linear algebra, the algebra of vectors and matrices. It yields nonlinear statistical models that are computed through tensor decomposition using a multimodal generalizaton of the SVD and dimensionality reduction.

In this paper, we propose Multilinear (Tensor) Active Appearance Models (MAAMs). These models naturally generalize the conventional, linear AAMs, which fail to distinguish between and account for the different sources of variabilty. The MAAM is a multilinear statistical model of facial appearance and shape that explicitly represents the underlying causal processes of image formation, thus preserving attributes that are relevant to the task of tracking the human face.

The remainder of this paper is organized as follows: Section 2 frames our effort in the context of the relevant prior work. Section 3 reviews conventional Active Appearance Models. Section 4 develops our Multilinar Active Appearance Models. Section 5 presents our experimental results in applying MAAMs to face tracking. Finally, Section 6 concludes the paper and discusses promising avenues for future work.

2. Related Work

Vasilescu and Terzopoulos [18, 19, 17] generalized the unifactor PCA approach to multifactor analysis using multilinear algebra and tensor decomposition. Multilinear anal-

ysis was also applied to gait recognition by Vasilescu [16] and Lee and Elgammal [9]. In work by Vlasic et al. [20], a multilinear model where 3D shape data is represented by tensors is used to transfer a face in a scene. The multilinear approach was also pursued in shape analysis by Sugano and Sato [13], where the multilinear model is used to decouple inter-personal and intra-personal variation.

AAMs have been extensively studied because of their ability to statistically model both shape and texture information simultaneously through PCA [2, 3, 11]. The drawback of PCA in this context, however, is that it is a unifactor model that cannot seperately capture the variation due to more than a single factor, such as variation in facial images due to pose, illumination, and expression (PIE). several attempts have been made to extend AAMs through multilinear analysis. Gonzalez et al. [5] introduce bilinear AAMs. The bilinear model is a two-factor special case of the multilinear model. Although it is consistent with our more general framework, their bilinear formulation uses matrices rather than tensors to formulate the model and there is no straightforward extension to the multifactor case.

Macedo et al. [10] proposed a hybrid of multilinear analysis and AAMs for facial expression transfer in photographs. Although their motivation to apply multilinear analysis is similar to ours, rather than directly integrating the multilinear model into AAMs, they use it only indirectly to factorize the parametric representation obtained via the standard AAMs. Although this is one possible way to apply multilinear analysis, important information about the observed variability in the training data can easily be lost through the linear projection in PCA. The focus of our work in this paper is the direct integration of tensors into the core of active appearance models. Unlike previous attempts to extend AAMs using multilinear analysis, we will reformulate AAMs with the use of multilinear PCA rather than conventional linear PCA.

3. Multilinear PCA

Multilinear analysis extends a conventional linear analysis using tensors. Appendix A reviews some relevant tensor fundamentals.

Let $\mathcal{D}$ be a $\Re^{I_{\mathrm{d}} \times I_{\mathrm{P}} \times I_{\mathrm{V}} \times I_{\mathrm{L}}}$ data tensor. Here, d denotes the observation mode, and P, V, and L denote the person, view, and illumination modes, respectively. Then, $\mathcal{D}$ can be decomposed in the following form [18, 8]:

$$\begin{aligned} \mathcal{D} &= \mathcal{Z} \times_{\mathrm{d}} \mathbf{U}_{\mathrm{d}} \times_{\mathrm{P}} \mathbf{U}_{\mathrm{P}} \times_{\mathrm{V}} \mathbf{U}_{\mathrm{V}} \times_{\mathrm{L}} \mathbf{U}_{\mathrm{L}} && (1) \\ &= \mathcal{T} \times_{\mathrm{P}} \mathbf{U}_{\mathrm{P}} \times_{\mathrm{V}} \mathbf{U}_{\mathrm{V}} \times_{\mathrm{L}} \mathbf{U}_{\mathrm{L}}, && (2) \end{aligned}$$

where $\mathcal{T} = \mathcal{Z} \times_{\mathrm{d}} \mathbf{U}_{\mathrm{d}}$ is the *basis tensor*, which is efficiently computed as

$$\mathcal{T} = \mathcal{D} \times_{\mathrm{P}} \mathbf{U}_{\mathrm{P}}^{T} \times_{\mathrm{V}} \mathbf{U}_{\mathrm{V}}^{T} \times_{\mathrm{L}} \mathbf{U}_{\mathrm{L}}^{T}. \quad (3)$$

One can model arbitrary many factors by adding modes to the model in (1) as necessary. For example, we can add an expression mode for face images. When we have two factors, (2) reduces to a bilinear model and for only a single factor, (2) reduces to the standard, linear PCA model. With respect to the latter, note that the data tensor decomposition (1) can be rewritten in matrix form as

$$\underbrace{\mathbf{D}_{[d]}}_{\text{Image Data } \mathbf{D}} = \underbrace{\mathbf{U}_{\mathrm{d}}}_{\text{Basis Vectors } \mathbf{B}} \underbrace{\mathbf{Z}_{[d]} (\mathbf{U}_{\mathrm{L}} \otimes \mathbf{U}_{\mathrm{V}} \otimes \mathbf{U}_{\mathrm{P}})^{T}}_{\text{Coefficients } \mathbf{C}}, \quad (4)$$

where the subscript $[\mathrm{d}]$ denotes the tensor matrixizing (flattening) operation with respect to the observation mode, which is the conventional PCA form $\mathbf{D} = \mathbf{B}\mathbf{C}$. The key difference compared to the unimodal analysis of linear PCA is that multilinear PCA further factorizes $\mathbf{C}$ to represent the additional modes of variation in the data tensor $\mathcal{D}$.

Analogous to dimensionality reduction in PCA, we can apply dimensionality reduction separately to the resulting mode matrices $\mathbf{U}_{\mathrm{P}}$, $\mathbf{U}_{\mathrm{V}}$, and $\mathbf{U}_{\mathrm{L}}$ in (5) to represent the data in lower-dimensional subspaces with with fewer parameters. Since the multilinear model is fundamentally nonlinear, however, simple truncation does not suffice and we can adopt an alternating least squares iterative procedure to compute a (locally) optimal dimensionality reduction [19].

Finally, the representation of a data vector $\mathbf{d}$ is given by $\{\mathbf{p}, \mathbf{v}\}$, as follows:

$$\mathbf{d} = \mathcal{T} \times_{\mathrm{P}} \mathbf{p}^{\mathrm{T}} \times_{\mathrm{V}} \mathbf{v}^{\mathrm{T}}, \quad (5)$$

where $\mathbf{p}$ and $\mathbf{v}$ are vectors which represent a probe of $\mathbf{d}$ for each mode. This multilinear equation is analogous to the standard linear PCA form

$$\mathbf{d} = \mathbf{B}\mathbf{c}, \quad (6)$$

where $\mathbf{c}$ is a parametric representation of $\mathbf{d}$ using the subspace matrix $\mathbf{B}$. Note that the tensor formulation in (5) has multiple decoupled parameter vectors, whereas the standard PCA has just a single parameter in (6).

4. Multilinear AAMs

In this section, we will derive the tensor formulation of Multilinear AAMs and develop an associated model fitting algorithm. We assume that the reader is familiar with the formulation and application of AAMs, as described in [3, 2].

4.1. Building MAAMs

Our training set is made up of labeled facial images in which important facial features are marked. Since the intensity data set and the shape data set were generated by the same facial geometry and imaging system, the data sets are

highly correlated and can be analyzed in an unified manner. We concatenate the shape and texture data tensors and model the underlying processes by analyzing the concatenated tensor in a multilinear framework.

4.1.1 Preprocessing

Before applying Multilinear-PCA, we usually need to normalize the training dataset to correctly extract their statistics. With regard to the shape data, we first apply Procrustes analysis to cancel out the influence of global similarity transformations, thus obtaining a normalized set of 2D points $\mathbf{s} = (x_1, y_1, x_2, y_2, \ldots, x_n, y_n)^{\mathrm{T}}$ and similarity transformation parameters $\mathbf{q} = (s_x, s_y, t_x, t_y)$ that map the normalized coordinates to the original image coordinates of each data sample. After Procrustes analysis, the data points $\mathbf{s}$ have zero mean and unit variance. Finally, we arrange each training sample $\mathbf{s}$ into a tensor $\mathcal{S}$ whose modes are shape data, people, and views.

For the texture data, we warp all the images into a uniform texture frame, extract the region of interest (the face) as a vector of pixels, and normalize these texture vectors. There are several methods to warp an image into a common coordinate system, but here we just do it by a piecewise linear transformation using the corresponding shape data and the mean shape obtained in the previous shape normalization. Once images are warped, pixels on region of interest are extracted and packed as a vector. Next, we normalize the extracted vector to have uniform variance and to compensate the influence of global illumination change across the images. That is, the resulting texture vector $\mathbf{g}$ will satisfy $\mathbf{g}^{\mathrm{T}}\mathbf{1} = 0$ and $\mathbf{g}^{\mathrm{T}}\mathbf{g} = 1$. Finally, the normalized texture vector $\mathbf{g}$ is arranged to construct a texture data tensor $\mathcal{G}$ along with different modes as we do in shape preprocessing.

Given the pre-processed shape data tensor $\mathcal{S}$ and texture data tensor $\mathcal{G}$, we concatenate them along the observation mode to form a combined data tensor $\mathcal{D} = \mathcal{I}_{\mathrm{d}} \times_{\mathrm{d}} \mathbf{D}_{[d]}$, where $\mathcal{I}_{\mathrm{d}}$ is the observation mode unit tensor and

$$\mathbf{D}_{[d]} = \begin{bmatrix} \alpha \mathbf{S}_{[d]} + \beta \mathbf{1}^{\mathrm{T}}\mathbf{1} \\ \mathbf{G}_{[d]} \end{bmatrix}, \tag{7}$$

wherein the shape and texture data tensors are concatenated in the observation mode and where α and β are scalars that scale the units between shape and texture. Because shape is defined in a coordinate space while texture is defined as pixel intensity, we set α and β to simply match the range of shape values to that of texture.

To construct a multiresolution model, we create different levels of the texture tensor $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_N$ from a pyramid

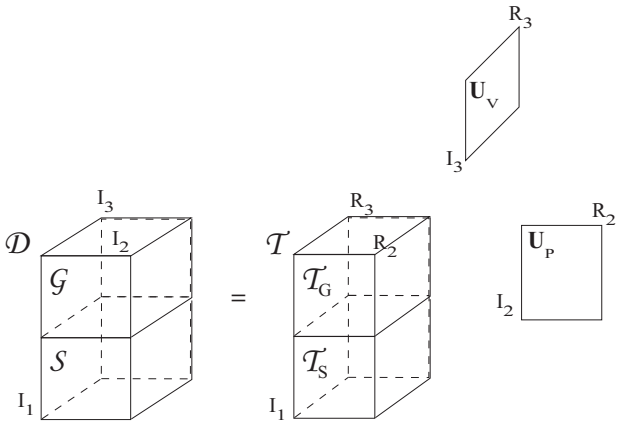


Figure 1. The MAAM decomposition

of training images. The matrixized data tensor

$$\mathbf{D}_{[d]} = \begin{bmatrix} \alpha \mathbf{S}_{[d]} + \beta \mathbf{1}^{\mathrm{T}}\mathbf{1} \\ \mathbf{G}_{1\,[d]} \\ \vdots \\ \mathbf{G}_{n\,[d]} \end{bmatrix} \tag{8}$$

is the concatenation in the observation mode of the shape tensor and the multiresolution texture tensors.

4.1.2 Computing the Basis Tensor

Once the zero-mean data tensor $\mathcal{D}$ is constructed, we apply the N-mode SVD algorithm [18] to decompose it according to (1) as

$$\mathcal{D} = \mathcal{Z} \times_{\mathrm{d}} \mathbf{U}_{\mathrm{d}} \times_{\mathrm{P}} \mathbf{U}_{\mathrm{P}} \times_{\mathrm{V}} \tag{9}$$

and obtain the mode matrices $\mathbf{U}_{\mathrm{P}}$ and $\mathbf{U}_{\mathrm{V}}$ associated with the people and view modes. This is illustrated in Figure 1. We then compute the basis tensor

$$\mathcal{T} = \mathcal{D} \times_{\mathrm{P}} \mathbf{U}_{\mathrm{P}}^{\mathrm{T}} \times_{\mathrm{V}} \mathbf{U}_{\mathrm{V}}^{\mathrm{T}}. \tag{10}$$

4.1.3 Representing Appearance

With the basis tensor $\mathcal{T}$ and coefficients for each factors $\mathbf{p}$ and $\mathbf{v}$, we can synthesize an appearance

$$\begin{aligned} \mathbf{d} &= \begin{bmatrix} \alpha \mathbf{s} + \beta \mathbf{1} \\ \mathbf{g} \end{bmatrix} \\ &= \bar{\mathbf{d}} + \mathcal{T} \times_{\mathrm{P}} \mathbf{p}^{\mathrm{T}} \times_{\mathrm{V}} \mathbf{v}^{\mathrm{T}}, \end{aligned} \tag{11}$$

where $\bar{\mathbf{d}}$ is a mean vector of all the data in $\mathcal{D}$. The resulting data vector $\mathbf{d}$ includes all the necessary information to construct an appearance.

To obtain a more convenient representation of equation (11), we split $\bar{\mathbf{d}}$ and $\mathcal{T}$ in the data mode with the corresponding length of shape and texture, as follows:

$$\bar{\mathbf{d}} \equiv \begin{bmatrix} \bar{\mathbf{s}}' \\ \bar{\mathbf{g}} \end{bmatrix} \equiv \begin{bmatrix} \alpha\bar{\mathbf{s}} + \beta\mathbf{1} \\ \bar{\mathbf{g}} \end{bmatrix}, \quad (12)$$

$$\mathbf{T}_{[d]} \equiv \begin{bmatrix} \mathbf{T}'_{\mathrm{S}[d]} \\ \mathbf{T}_{\mathrm{G}[d]} \end{bmatrix} \equiv \begin{bmatrix} \alpha\mathbf{T}_{\mathrm{S}[d]} \\ \mathbf{T}_{\mathrm{G}[d]} \end{bmatrix}. \quad (13)$$

Then, rewriting (11) in terms of shape and texture vectors yields

$$\mathbf{s} = \bar{\mathbf{s}} + \mathcal{T}_{\mathrm{S}} \times_{\mathrm{P}} \mathbf{p}^{\mathrm{T}} \times_{\mathrm{V}} \mathbf{v}^{\mathrm{T}}, \quad (14)$$

$$\mathbf{g} = \bar{\mathbf{g}} + \mathcal{T}_{\mathrm{G}} \times_{\mathrm{P}} \mathbf{p}^{\mathrm{T}} \times_{\mathrm{V}} \mathbf{v}^{\mathrm{T}}. \quad (15)$$

Equations (14) and (15) show the direct multilinear relationship between parametric representation $\{\mathbf{p}, \mathbf{v}\}$ and the corresponding shape $\mathbf{s}$ and texture $\mathbf{g}$. Note that for the multiresolution approach $\mathbf{g}$ will be replaced with the multiresolution texture vectors $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_N$.

4.2. Fitting MAAMs

We need to have a fitting algorithm to obtain a parametric representation of a new appearance using a built model. Now, let's set the purpose of model fitting to obtain shape data given a new image.

Given an new image $\mathbf{i}$, the objective of Multilinear AAM fitting is to minimize

$$\min_{\mathbf{p},\mathbf{v},\mathbf{q}} ||\mathbf{g}_{\mathrm{i}}(\mathbf{i}, N(\mathbf{s}_{\mathrm{m}}(\mathbf{p}, \mathbf{v}), \mathbf{q}) - \mathbf{g}_{\mathrm{m}}(\mathbf{p}, \mathbf{v})||, \quad (16)$$

where $\mathbf{g}_{\mathrm{i}}$ is the texture vector obtained from image $\mathbf{i}$, and $\mathbf{g}_{\mathrm{m}}$ is a texture vector given by the reprojection of parameters $\mathbf{p}$ and $\mathbf{v}$ from the model. Also, $\mathbf{s}_{\mathrm{m}}$ is a shape data given by the reprojection of parameters $\mathbf{p}$ and $\mathbf{v}$. It is used to obtain $\mathbf{g}_{\mathrm{i}}$ from $\mathbf{i}$ together with parameters of similarity transform $\mathbf{q}$. The evaluation of the error is done in texture space because we usually do not know the shape data for a new image.

Unfortunately, the equation (16) is not easy to solve because it is a nonlinear optimization. One approach is to use a generic nonlinear optimization algorithm such as Gauss-Newton or Levenberg-Marquardt method. But, the drawback of this approach is that the computation is quite expensive.

In this paper, we adopt a simple iterative approach by introducing an approximation that the residual of textures at the current estimate includes all the necessary information to refine the estimate.

4.2.1 Parameter Update

The strategy is to iteratively refine the parameters $\mathbf{p}$, $\mathbf{v}$, and $\mathbf{q}$ by small updates $\delta\mathbf{p}$, $\delta\mathbf{v}$, and $\delta\mathbf{q}$; i.e., $\mathbf{p} \leftarrow \mathbf{p} + \delta\mathbf{p}$, $\mathbf{v} \leftarrow \mathbf{v} + \delta\mathbf{v}$, and $\mathbf{q} \leftarrow \mathbf{q} + \delta\mathbf{q}$. The most straightforward way to compute these updates is to compute the derivatives of $\delta\mathbf{g} = \mathbf{g}_{\mathrm{i}} - \mathbf{g}_{\mathrm{m}}$ with respect to $\delta\mathbf{p}$, $\delta\mathbf{v}$, and $\delta\mathbf{q}$. We express the derivative in the following multilinear form:

$$\delta\mathbf{g}(\delta\mathbf{p}, \mathbf{v}, \mathbf{q}) = \mathcal{J}_{\mathrm{P}} \times_{\mathrm{P}} \delta\mathbf{p}^{\mathrm{T}} \times_{\mathrm{V}} \mathbf{v}^{\mathrm{T}} \times_{\mathrm{Q}} \mathbf{q}^{\mathrm{T}} \quad (17)$$

$$\delta\mathbf{g}(\mathbf{p}, \delta\mathbf{v}, \mathbf{q}) = \mathcal{J}_{\mathrm{V}} \times_{\mathrm{P}} \mathbf{p}^{\mathrm{T}} \times_{\mathrm{V}} \delta\mathbf{v}^{\mathrm{T}} \times_{\mathrm{Q}} \mathbf{q}^{\mathrm{T}} \quad (18)$$

$$\delta\mathbf{g}(\mathbf{p}, \mathbf{v}, \delta\mathbf{q}) = \mathcal{J}_{\mathrm{Q}} \times_{\mathrm{P}} \mathbf{p}^{\mathrm{T}} \times_{\mathrm{V}} \mathbf{v}^{\mathrm{T}} \times_{\mathrm{Q}} \delta\mathbf{q}^{\mathrm{T}}, \quad (19)$$

where $\mathcal{J}_{\mathrm{P}}$, $\mathcal{J}_{\mathrm{V}}$, and $\mathcal{J}_{\mathrm{Q}}$ are "Jacobian tensors" that relate the residual $\delta\mathbf{g}$ computed at the current estimates $\{\mathbf{p}, \mathbf{v}, \mathbf{q}\}$ to the updates $\{\delta\mathbf{p}, \delta\mathbf{v}, \delta\mathbf{q}\}$.

4.2.2 Learning Jacobian Tensors

We can compute the Jacobian tensors from the training data. Since we already know how to reproject data using (14) and (15), we can compute $\delta\mathbf{g}$ for parameters slightly displaced from those of each training set, $\mathbf{p} - \delta\mathbf{p}$, $\mathbf{v} - \delta\mathbf{v}$, and $\mathbf{q} - \delta\mathbf{q}$. We first compute diagonal matrices $\mathbf{M}_{\delta\mathrm{P}} = [\delta\mathbf{p}_1, \delta\mathbf{p}_2, \ldots, \delta\mathbf{p}_{N_p}]^{\mathrm{T}}$, $\mathbf{M}_{\delta\mathrm{V}} = [\delta\mathbf{v}_1, \delta\mathbf{v}_2, \ldots, \delta\mathbf{v}_{N_v}]^{\mathrm{T}}$, and $\mathbf{M}_{\delta\mathrm{Q}} = [\delta\mathbf{q}_1, \delta\mathbf{q}_2, \ldots, \delta\mathbf{q}_{N_q}]^{\mathrm{T}}$, where $\delta\mathbf{p}_i$, $\delta\mathbf{v}_i$, and $\delta\mathbf{q}_i$ are vectors whose elements are zeros except for the i^{th} element, which is set to the displacement δp_i, δv_i, and δq_i to be applied for the i^{th} element in $\mathbf{p}$, $\mathbf{v}$, and $\mathbf{q}$, respectively. Also we compute residual tensors $\delta\mathcal{G}_{\mathrm{P}}$, $\delta\mathcal{G}_{\mathrm{V}}$, and $\delta\mathcal{G}_{\mathrm{Q}}$, where we arrange $\delta\mathbf{g}$ computed in accordance with the combination of parameters $\{\mathbf{p}, \mathbf{v}, \mathbf{q}\}$ systematically displaced by $\delta\mathbf{p}_i$, $\delta\mathbf{v}_i$, or $\delta\mathbf{q}_i$, respectively. Then, we will have equations,

$$\delta\mathcal{G}_{\mathrm{P}} = \mathcal{J}_{\mathrm{P}} \times_{\mathrm{P}} \mathbf{M}_{\delta\mathrm{P}} \times_{\mathrm{V}} \mathbf{U}_{\mathrm{V}} \times_{\mathrm{Q}} \mathbf{U}_{\mathrm{Q}}, \quad (20)$$

$$\delta\mathcal{G}_{\mathrm{V}} = \mathcal{J}_{\mathrm{V}} \times_{\mathrm{P}} \mathbf{U}_{\mathrm{P}} \times_{\mathrm{V}} \mathbf{M}_{\delta\mathrm{V}} \times_{\mathrm{Q}} \mathbf{U}_{\mathrm{Q}}, \quad (21)$$

$$\delta\mathcal{G}_{\mathrm{Q}} = \mathcal{J}_{\mathrm{Q}} \times_{\mathrm{P}} \mathbf{U}_{\mathrm{P}} \times_{\mathrm{V}} \mathbf{U}_{\mathrm{V}} \times_{\mathrm{Q}} \mathbf{M}_{\delta\mathrm{Q}}. \quad (22)$$

Once we get (20)–(22), we can compute the Jacobian tensors

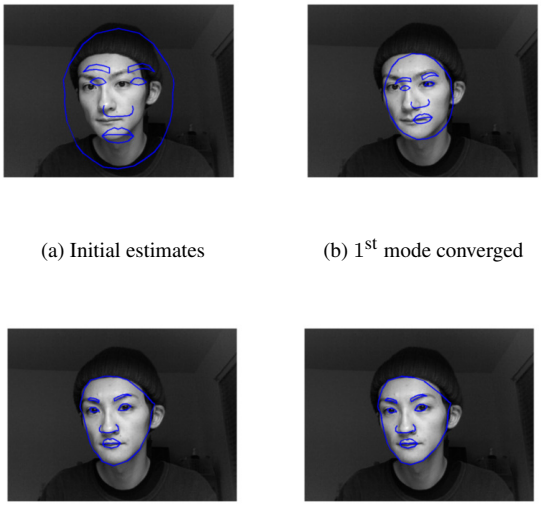
$$\mathcal{J}_{\mathrm{P}} = \delta\mathcal{G}_{\mathrm{P}} \times_{\mathrm{P}} \mathbf{M}^{+}_{\delta\mathrm{P}} \times_{\mathrm{V}} \mathbf{U}^{\mathrm{T}}_{\mathrm{V}} \times_{\mathrm{Q}} \mathbf{U}^{\mathrm{T}}_{\mathrm{Q}} \quad (23)$$

$$\mathcal{J}_{\mathrm{V}} = \delta\mathcal{G}_{\mathrm{V}} \times_{\mathrm{P}} \mathbf{U}^{\mathrm{T}}_{\mathrm{P}} \times_{\mathrm{V}} \mathbf{M}^{+}_{\delta\mathrm{V}} \times_{\mathrm{Q}} \mathbf{U}^{\mathrm{T}}_{\mathrm{Q}} \quad (24)$$

$$\mathcal{J}_{\mathrm{Q}} = \delta\mathcal{G}_{\mathrm{Q}} \times_{\mathrm{P}} \mathbf{U}^{\mathrm{T}}_{\mathrm{P}} \times_{\mathrm{V}} \mathbf{U}^{\mathrm{T}}_{\mathrm{V}} \times_{\mathrm{Q}} \mathbf{M}^{+}_{\delta\mathrm{Q}}, \quad (25)$$

where $+$ superscripts denote the pseudo-inverse. Since we have different combination of $\{\mathbf{p}, \mathbf{v}, \mathbf{q}\}$ from training data, we can use them to compute Jacobian tensors by linear regression; we can arrange $\mathbf{M}_{\delta\mathrm{P}}$, $\mathbf{M}_{\delta\mathrm{V}}$, and $\mathbf{M}_{\delta\mathrm{Q}}$ as vertically repeated diagonal matrices in accordance with the number of combination $\{\mathbf{p}, \mathbf{v}, \mathbf{q}\}$, with $\delta\mathcal{G}$ arranged in the same manner.

The amount of displacement is determined experimentally. For $\delta\mathbf{p}$ and $\delta\mathbf{v}$, we use 50% of the standard deviation for each element in $\mathbf{p}$ and $\mathbf{v}$. For the parameters of the similarity transform $\mathbf{q}$, we use 10% of the displacement for scaling and 3 to 5 pixels for translation.



(a) Initial estimates (b) 1st mode converged

(c) 1st resolution converged (d) Final result

Figure 2. Iterative fitting results

4.2.3 Iterative Model Refinement

Using the Jacobian tensors, we iteratively refine the current estimate of parameters $\mathbf{p}$, $\mathbf{v}$, and $\mathbf{q}$. The update is given by

$$\delta\mathbf{p}^{\mathrm{T}} = \delta\mathbf{g}^{\mathrm{T}}(\mathcal{J}_{\mathrm{P}} \times_{\mathrm{V}} \mathbf{v}^{\mathrm{T}} \times_{\mathrm{Q}} \mathbf{q}^{\mathrm{T}})^{+}_{[P]} \quad (26)$$

$$\delta\mathbf{v}^{\mathrm{T}} = \delta\mathbf{g}^{\mathrm{T}}(\mathcal{J}_{\mathrm{V}} \times_{\mathrm{P}} \mathbf{p}^{\mathrm{T}} \times_{\mathrm{Q}} \mathbf{q}^{\mathrm{T}})^{+}_{[V]} \quad (27)$$

$$\delta\mathbf{q}^{\mathrm{T}} = \delta\mathbf{g}^{\mathrm{T}}(\mathcal{J}_{\mathrm{Q}} \times_{\mathrm{P}} \mathbf{p}^{\mathrm{T}} \times_{\mathrm{V}} \mathbf{v}^{\mathrm{T}})^{+}_{[Q]} \quad (28)$$

Note that we need to take into account the uncertainty of the sign in the parametric representation of the multilinear projection. Since either one of the combinations $\{\mathbf{p}, \mathbf{v}\}$ or $\{-\mathbf{p}, -\mathbf{v}\}$ gives the identical data representation for the current estimate, we need to choose $\mathbf{c} \leftarrow \mathbf{c} + \delta\mathbf{c}$ or $\mathbf{c} \leftarrow \mathbf{c} - \delta\mathbf{c}$ in updating mode c of the multilinear model. The selection is simply done by choosing the one that most reduces the norm of residual of texture, and it improves the stability of numerical computation.

The iterative fitting algorithm is specified in Figure 3:

Unfortunately, different order of mode-n refinement yields different fitting results due to local minima. Our recommendation is to update modes in the order of the most influential modes that will have the largest effect. This in our case, this ordering is $\mathbf{q}$, $\mathbf{v}$, $\mathbf{p}$—the similarity transform parameters, followed by the viewpoint parameters, followed by the people parameters.

Convergence is determined by how the Jacobian tensors are learned. It is possible to control the convergence of our refinement algorithm by introducing a damping parameter, for example $k = \{1.0, 0.5\}$, and by updating $\mathbf{c}$ by $\mathbf{c} \leftarrow \mathbf{c} \pm k\delta\mathbf{c}$, we can control the convergence to obtain a better parameter fit.

MAAM Fitting Algorithm:

Iterative Model Refinement

1. Prepare initial parameter estimate $\{\mathbf{p}, \mathbf{v}, \mathbf{q}\}$.
2. Apply mode-n refinement described below to all of current estimates $\{\mathbf{p}, \mathbf{v}, \mathbf{q}\}$ in turn .
3. Return to step 2 until no update is made to either one of parameters or it reaches the maximum number of iteration.

Mode-n Refinement

1. Prepare the initial parameter estimate $\mathbf{c}$ for mode n and fix the parameters for the other modes.
2. Compute the residual of appearance $\delta\mathbf{g}$ for the current estimate.
3. Compute the update of parameter $\delta\mathbf{c}$ using (26)–(28).
4. Compute $\delta\mathbf{g}'$, the residual evaluated at $\mathbf{c} \pm \delta\mathbf{c}$.
5. Update $\mathbf{c}$ by $\mathbf{c} \leftarrow \mathbf{c} + \delta\mathbf{c}$ or $\mathbf{c} \leftarrow \mathbf{c} - \delta\mathbf{c}$ if either one of them reduces the norm of $\delta\mathbf{g}'$ from that of $\delta\mathbf{g}$.
6. Repeat from step 2 until convergence or reaching a maximum number of iterations.

Figure 3. The iterative MAAM fitting algorithm

We can incorporate a coarse-to-fine fitting approach in this iterative fitting with the use of the multiresolution model and multiresolution input image. To implement multiresolution fitting, we need to compute Jacobian tensors for each texture resolution. We start the iterative model refinement from the coarsest resolution and proceed to the finest resolution using the parameter estimates obtained from the coarser resolution level as an initial estimate for the next finer level. Figure 2 shows the example of multiresolution fitting of a two-resolution model applied to a face image.

5. Experimental Results

In this section, we present the experimental result of our tensor AAMs applied to face tracking. For training data, we prepared a set of grayscale images of size 320×240 pixels with hand-labeled 2D feature points on them. The training set has data for 50 different people and 25 different viewpoints ranging from -60 to +60 degree from the frontal (0 degree) view. We used only one illumination for training data in this experiment. Figure 4 shows the data used to

Figure 4. Training data

train the multilinear model. All the experiments are done offline.

Figure 5 shows the result of face tracking. Figure 6 depicts the 1st three components of the estimated viewing parameters $\mathbf{v}$ during the sequence, relative to viewing parameters associated with the training data. Although there is some tracking noise, the overall trajectory of the viewpoint parameters follows the plot of the viewpoint parameters of training data. However, the person parameters tend to move independently with the entire plots of training data. This indicates that $\mathbf{p}$ and $\mathbf{v}$ are decoupled in each subspace. Note that we would be able to recognize the angle of the face by classifying the viewpoint parameters using the training data.

Although tracking was successful, we encountered several issues in this experiment. First, the fitting algorithm become trapped in local minima.Since it optimizes each mode in turn, we can select the order of the modes to optimize. We can select the maximum number of iteration allowed for each mode, or the amount of displacement chosen in the learning process of the Jacobian tensor. Tracking through an extreme pose can also lead to a local minimum due to the lack of texture information during occlusion, and the static relationship between the residual and the update of the parameters. These are also a common problems when applying conventional AAMs.

6. Conclusion

We have proposed multilinear analysis to explicitly parameterize different modes of variation in appearance/shape-based models. In particular, we have developed a generalized multilinear formulation of AAMs using tensors. In addition, we have proposed an iterative fitting algorithm to fit parametric MAAM models to images. The fitting algorithm exploits the multilinear relationship between the residual of the texture and iteratively updates the individual parameters. Our experimental results with face tracking have demonstrated that the fitting algorithm successfully extracts viewpoint parameters for the motion of the face in a video sequence.

Although we have formulated a general multilinear (tensor) of AAMs that is capable of dealing with an arbitrary number of causal factors, we have initially demonstrated our framework on a third-order tensor and performed a bilinar (two-mode) decomposition involving people and views modes. In future work, we will evaluate the model with additional causal factors including changes in illumination and expression. In addition, we will improve the efficiency of the fitting process for use in real-time applications.

A. Some Tensor Fundamentals

A tensor is a higher order generalization of a vector (first-order tensor) and a matrix (second-order tensor). Tensors represent multilinear mappings from a set of domain vector spaces to a range vector space.

An N^{th} order tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ has elements denoted by $a_{i_1 \ldots i_n \ldots i_N}$, where $1 \leq i_n \leq I_n$. The mode-n vectors of an Nth order tensor $\mathcal{A}$ are the I_n-dimensional vectors obtained from A by varying index i_n while keeping the other indices fixed. In other words, the mode-n vectors are the columns of matrix $\mathbf{A}_{[i]} \in \mathbb{R}^{I_n \times (I_{n+1} \ldots I_N I_1 I_2 \ldots I_{n-1})}$, where the subscript $_{[n]}$ denotes mode-n matrixizing of a tensor.

Mode-n Product The mode-n product $\mathcal{A} \times_{\text{n}} \mathbf{M}$, where $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n \times \cdots \times I_N}$ and $\mathbf{M} \in \mathbb{R}^{J_n \times I_n}$, is the $I_1 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N$ tensor

$$(\mathcal{A} \times_{\text{n}} \mathbf{M})_{i_1 \ldots i_{n-1} j_n i_{n+1} \ldots i_N} = \sum_{i_n} a_{i_1 \ldots i_{n-1} i_n i_{n+1} \ldots i_N} m_{j_n i_n}. \quad (29)$$

The mode-n product can be expressed in terms of matrixized tensors as $\mathbf{B}_{[\text{n}]} = \mathbf{M}\mathbf{A}_{[\text{n}]}$.

N-Mode SVD The tensor decomposition, which is analogous to singular value decomposition of a matrix, is described as follows: Let $\mathcal{A}$ be a $I_1 \times I_2 \cdots \times I_n \cdots \times I_N$ tensor for $1 \leq n \leq N$. Every such tensor can be decomposed as follows:

$$\begin{aligned} \mathcal{A} &= \mathcal{Z} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \cdots \times_n \mathbf{U}_n \cdots \times_N \mathbf{U}_N \\ &= \sum_{i_1=1}^{R_1} \sum_{i_2=1}^{R_2} \cdots \sum_{i_n=1}^{R_n} \cdots \sum_{i_N=1}^{R_N} \sigma_{i_1 i_2 \ldots i_N} \\ &\qquad \mathbf{u}_1^{(i_1)} \circ \mathbf{u}_2^{(i_2)} \cdots \circ \mathbf{u}_n^{(i_n)} \cdots \circ \mathbf{u}_N^{(i_N)}, \end{aligned} \quad (30)$$

where $\mathbf{U}_n = [\mathbf{u}_n^{(1)} \mathbf{u}_n^{(2)} \ldots \mathbf{u}_n^{(i_n)} \ldots \mathbf{u}_n^{(R_n)}]$ are orthonormal mode-n matrices of dimensionality $I_n \times R_n$ for $1 \leq R_n \leq$

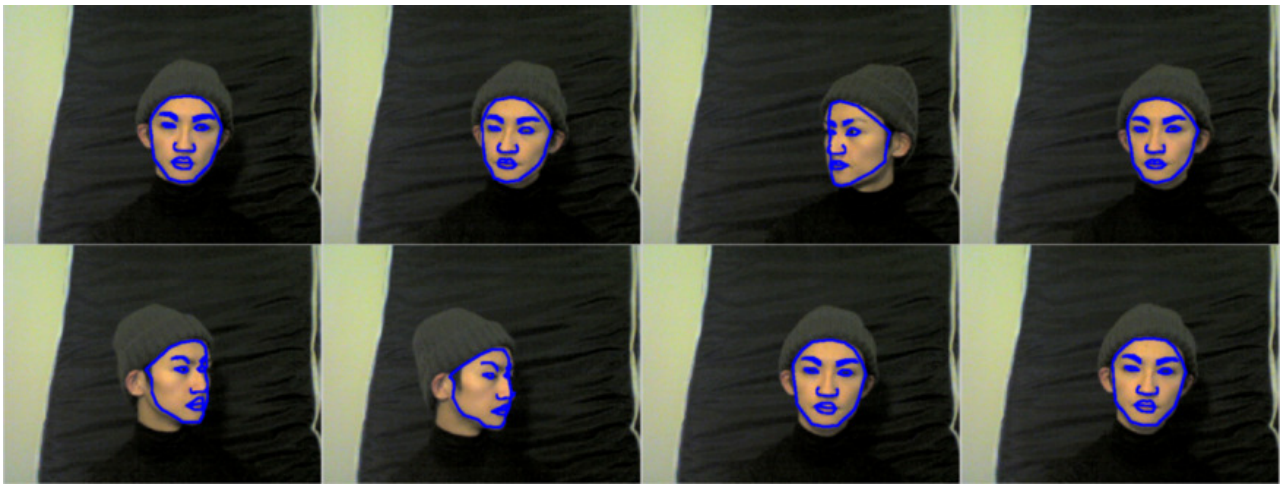
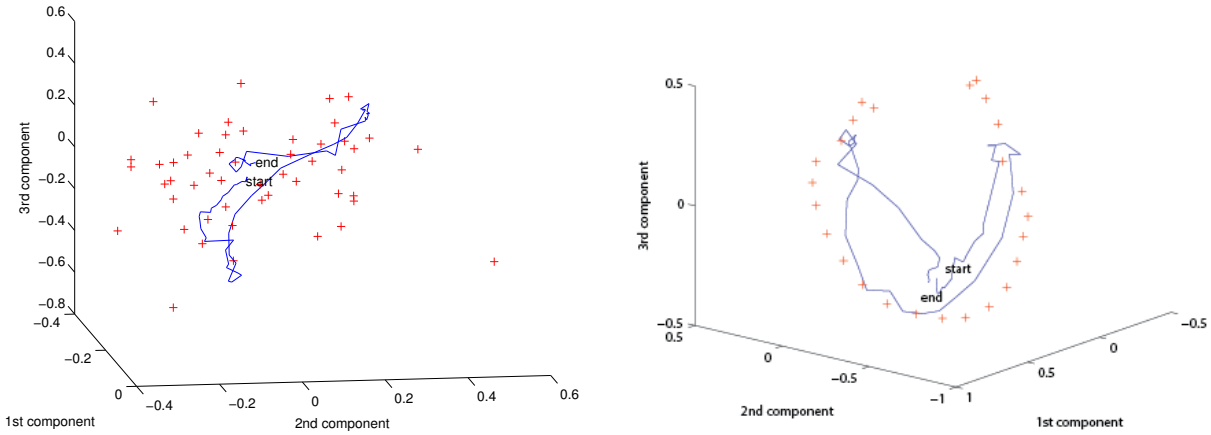


Figure 5. Result of face tracking. The initial frame (frame 1) is at the upper left, and the final frame (frame 68) is at the lower right.



(a) Person coefficients

(b) Viewpoint coefficients

Figure 6. 1st to 3rd components of the normalized parameters during face tracking. The '+' points plot the coefficients of the training data which include 50 people and 25 viewpoints ranging from 30 to 150 degrees.

I_n for $1 \leq n \leq N$ and $\mathcal{Z} \in \mathbb{R}^{R_1 \times R_2 \cdots \times R_n \cdots \times R_N}$. The subtensors $\mathcal{Z}_{i_n=a}$ and $\mathcal{Z}_{i_n=b}$ obtained by fixing the n^{th} index to a and b are orthogonal for all values of n, a, and b when $a \neq b$. The $\|\mathcal{Z}_{i_n=a}\| = \sigma_a^{(n)}$ is the a^{th} mode-n singular value of $\mathcal{A}$ and the a^{th} column vector of $\mathbf{U}_n$, such that $\|\mathcal{Z}_{i_n=1}\| \geq \|\mathcal{Z}_{i_n=2}\| \geq \ldots \|\mathcal{Z}_{i_n=R_n}\| \geq 0$.

References

[1] T. Cootes, G. Wheeler, K. Walker, and C. Taylor. View-based active appearance models. *Image and Vision Computing*, 20(9-10):657–664, 2002. 1

[2] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *Proc. 5th European Conference on Computer Vision*, 2:484–498, 1998. 1, 2

[3] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001. 2

[4] G. Edwards, C. Taylor, and T. Cootes. Interpreting face images using active appearance models. In *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 300–305, 1998. 1

[5] J. Gonzalez, F. D. l. T. Frade, R. Murthi, N. G. Mata, and E. Zapata. Bilinear active appearance models.

Proc. Workshop on Non-rigid Registration and Tracking through Learning, 2007. 2

[6] M. Jones and T. Poggio. Multidimensional morphable models: A framework for representing and matching object classes. *International Journal of Computer Vision*, 29(2):107–131, 1998. 1

[7] M. Kirby and L. Sirovich. Application of the Karhunen-Loeve procedure for thecharacterization of human faces. *IEEE Transactions on Pattern analysis and Machine intelligence*, 12(1):103–108, 1990. 1

[8] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000. 2

[9] C. Lee and A. Elgammal. Towards scalable view-invariant gait recognition: Multilinear analysis for gait. *Proc. International Conference on Audio and Video-Based Biometric Person Authentication*, pages 395–405, 2005. 2

[10] I. Macedo, E. V. Brazil, and L. Velho. Expression transfer between photographs through multilinear aam's. *Proc. 19th Brazilian Symposium on Computer Graphics and Image Processing*, pages 239–246, 2006. 2

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

[12] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. In *Proceedings 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 84–91, 1994. 1

[13] Y. Sugano and Y. Sato. Person-independent monocular tracking of face and facial actions with multilinear models. *Proc. IEEE International Workshop on Analysis and Modeling of Faces and Gestures*, 2007. 2

[14] M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999. 1

[15] M. Turk and A. Pentland. Face recognition using eigenfaces. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–591, 1991. 1

[16] M. A. O. Vasilescu. Human motion signatures: Analysis, synthesis, recognition. In *Proc. Int. Conf. on Pattern Recognition*, volume 3, pages 456–460, Quebec City, August 2002. 2

[17] M. A. O. Vasilescu and D. Terzopoulos. Multilinear Projection for Appearance-Based Recognition in the Tensor Framework. In *Proc. 11th IEEE International Conference on Computer Vision (ICCV'07)*, pages 1–8, 2007. 1

[18] M. A. O. Vasilescu and D. Terzopoulous. Multilinear analysis of image ensembles: Tensorfaces. *Proc. 7th European Conference on Computer Vision*, 1:447–460, 2002. 1, 2, 3

[19] M. A. O. Vasilescu and D. Terzopoulous. Multilinear subspace analysis of image ensembles. *Proc. 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:II–93–9 vol.2, 2003. 1, 2

[20] D. Vlasic, M. Brand, H. Pfister, and J. Popović. Face transfer with multilinear models. *ACM Trans. Graphics*, 24(3):426–433, 2005. 2