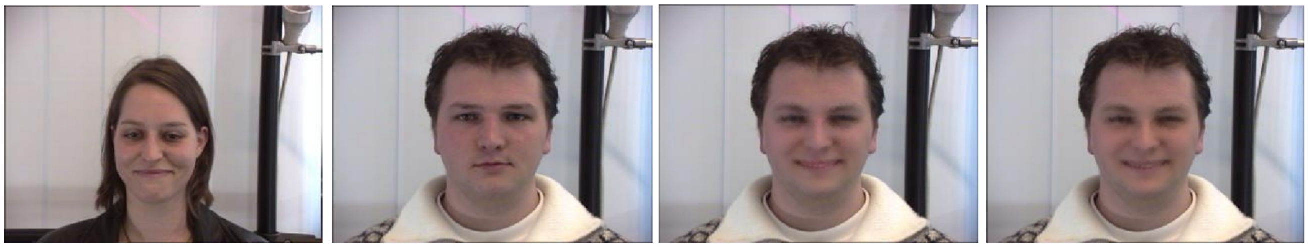
Expression Transfer between Photographs through Multilinear AAM's

Ives Macêdo Emilio Vital Brazil Luiz Velho
IMPA – Instituto Nacional de Matemática Pura e Aplicada
E-mail: {ijamj,emilio,lvelho}@visgraf.impa.br



Source Expression ***Target Identity*** ***L*a*b* Exchange*** ***RGB Exchange***

Abstract

Expression Transfer is a method for mapping a photographed expression performed by a given subject onto the photograph of another person's face. Building on well succeeded previous works by the vision researchers (facial expression decomposition, active appearance models and multilinear analysis, we propose a novel approach for expression transfer based on color images. We attack this problem with methods developed by the computer vision community for facial expression analysis and recognition. Combining active appearance models and multilinear analysis, it's possible to suitably represent *and* analyze *expressive facial images, while separating both style (subject's identity) and content (expressive flavor) from the captured performance.*

1. Introduction

The face is our primary focus of attention, playing a major role in conveying identity and emotion. Developing a computational model is a quite difficult task, because faces are complex, multidimensional and meaningful visual stimuli [9]. Facial expression analysis and synthesis has applications in areas such as human-computer interaction and data-driven animation. Expression Transfer is a method for mapping a photographed expression performed by a given subject onto the photograph of another person's face.

Contributions. In this paper we propose the *Multilinear AAM Expression Transfer* method to resynthesize new face images from two different subjects, where the first gives "expression" and the second gives "identity". We use the six "basic expressions" (anger, disgust, suprise, fear, sadness and happiness) plus a neutral face to build our database. Our approach employs *Active Appearance Models* [2, 3] to represent a database of facial images and creates two third-order tensors to describe variations due to identity and expressions. In this method, the *Higher-Order Singular Value Decomposition* (HOSVD) [10] is used to factorize the tensors along both identity and expression modes of variation. After projection onto the "tensor space", resynthesis is performed by swapping the expression factors between the subjects followed by evaluation of a multilinear operator (core tensor).

Related work. There has been a lot of work on facial image analysis and synthesis, our approach was influenced by two works, Vlasic *et al.* [12] presents a method to transfer faces, expressions, visemes and three-dimensional poses from monocular video or film, from which we adopt some ideas to 2D models. The second work, of Wang and Ahuja [14], about facial expression decomposition, was based on results of Cootes *et al.* (AAM's) [3] and Vasilescu and Terzopoulos (Multilinear Analysis) [10]. It proposes a novel approach for facial expression decomposition, obtaining good results for analysis but not exploring resynthesis applications. Our work presents a method to learn and to transfer expression with color image synthesis.

Zhang *et al.* [15] proposes a technique to map a facial expression using an example-based approach, they resynthesize a subject that is in the training database, while our method works even if both target and source subjects aren't in the database. Liu *et al.* [6] shows a technique, called expression ratio images, to enhance facial expression mapping, capturing the subtle but visually important details of facial expressions. Their method needs two source images one with the same expression of the target and another with the final expression.

Turk and Pentland [9] used principal component analysis to describe face images in terms of *Eigenfaces*, but this method is not robust to shape changes due to expressions, and does not deal well with variability in pose and illumination. Cootes *et al.* [3] present the *Active Appearance Model* (AAM), that learns valid shape and intensity variations from a training set. AAM is a compact and simple model for the appearance of objects and is capable of generating synthetic examples very similar to those in the training set. Vasilescu and Terzopoulos [10] demonstrated the power of the HOSVD on ensembles of facial images, yielding the *TensorFaces* framework.

Paper outline. An overview of *active appearance models* for facial images representation is presented in the folowing section. Some concepts and results from *tensor algebra* used in *multilinear analysis* are discussed in section 3. Our method to expression transfer is detailed in section 4 and some of its results are presented in section 5. Section 6 concludes with a discussion about current limitations and some directions for future research.

2. Active Appearance Models

Active Appearance Models [3] learn shape and intensity variations from their training set. There are two models in AAM, the *shape model* and *appearance model* of a subject, they are principal components (PCA) model learned from training data, hence both are linear models (subspaces).

The shape of a AAM example is a set of 2D coordinates of n landmark points, $P = (p_1, p_2, \cdots, p_n)$. After *Procrustes analysis* we normalize the points' coordinates and then use the PCA to describe points in terms of a set of coefficients s_i. Any shape P can then be approximated as a linear combination of an orthonormal basis [2].

$$\mathbf{s}(P) = \mathbf{s}_0 + \sum_{i=1}^{n} \sigma_i s_i$$

where $\mathbf{s}_0$ is the mean shape, and σ_i's form a set of orthogonal modes of variations and s_i's form a set of shape parameters.

To build a PCA model of the appearance, we warp each example image so that its control points match the "Procrustes' shape" (figure 3 shows the template triangulation). We then sample the pixels information I_{im} from the shape-normalized image over the region covered by the "Procrustes' shape". By applying PCA to the normalized data, we obtain a linear model to an image:

$$\mathbf{a}(I_{im}) = \mathbf{a}_0 + \sum_{i=1}^{n} \alpha_i a_i$$

where $\mathbf{a}_0$ is the mean of the normalized images, and α_i's form a set of orthogonal modes of variations and a_i form a set of appearance parameters. To represent a subject we just need to know the set of shapes and appearance parameters (a_i's and s_i's), thats reduces the computational cost to manipulate subjects.

In this paper, we consider the simpler case of independent AAM's [7], where the statistical dependence between the shape and appearance is not considered. After AAM training we construct a "shape tensor" and an "appearance tensor" with coefficients provided by the PCA step then we apply the Multilinear PCA. These topics will be presented in the next section.

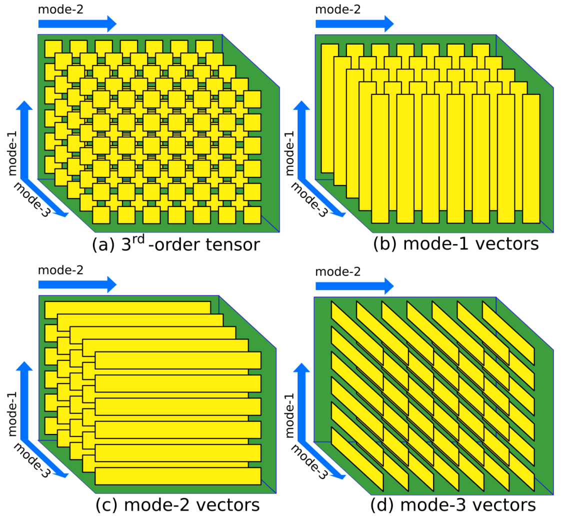
3. Multilinear Analysis

Introduced to the computer vision and graphics communities by Vasilescu and Terzopoulos, Multilinear Algebra is the algebra of higher-order tensors, which define multilinear operators over a set of vector spaces. Multilinear Analysis offers a unifying mathematical framework suitable for addressing a variety of computer vision and graphics problems [10, 11].

Multilinear anlysis has a basic object, the tensor, that is a natural generalization of vectors (first-order tensors) and matrices $m \times n$ (second order tensor) to multiple indices. With tensor, multidimensional matrix, we give more structure of information to an ensemble of images and then a best analysis and synthesis. We use bold lower-case letters $(\mathbf{v}, \mathbf{x}, \mathbf{w} \cdots)$ for vectors, bold upper case-letters $(\mathbf{A}, \mathbf{B}, \mathbf{C} \cdots)$ for matrices and calligraphic upper-case $(\mathcal{T}, \mathcal{U}, \mathcal{V} \cdots)$ for higher-order tensors. The order of tensor $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_N}$ is N. An element of $\mathcal{T}$ is denoted as $t_{i_1 i_2 \cdots i_N}$.

A tensor $\mathcal{T}$ has rank-1 when $\mathcal{T} = \mathbf{v_1} \otimes \mathbf{v_2} \otimes \cdots \otimes \mathbf{v_N}$ where $\otimes$ denotes the tensor product and $\mathbf{v_i}$'s are vectors in $\mathbb{R}^{d_i}$, note that $\mathcal{T}$ has dimensions $d_1 \times d_2 \times \cdots \times d_N$. The element of tensor $\mathcal{T}$ is expressed as $t_{i_1 i_2 \cdots i_N} = v_{1 i_1} v_{2 i_2} \cdots v_{N i_N}$ where $v_{j i_j}$ is the i^{th} component of $\mathbf{v_j}$. The rank of a tensor, R=$rank\,(\mathcal{T})$ is the minimal number of rank-1 tensors ($\mathcal{T}_i$) which we can write $\mathcal{T}$ like a linear combination of $\mathcal{T}_i$'s: $\sum_{i=1}^{R} \alpha_i \mathcal{T}_i$.

We can generalize the definition of column and row spaces of matrices. The mode-n vectors of a tensor, $\mathcal{T} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$, are the vectors that we fixed the indexes $i_1, \cdots, i_{n-1}, i_{n+1}, \cdots, i_N$ and varying the index i_n, as depicted in figure 1 (e.g. mode-1 and mode-2 vectors of a matrix correspond to its columns and rows, respectively). Mode-n flattening is the stacking of mode-n



(a) 3rd-order tensor (b) mode-1 vectors (c) mode-2 vectors (d) mode-3 vectors

Figure 1. Mode-n vectors of a 3^{rd}-order tensor of dimensions $6 \times 7 \times 5$.

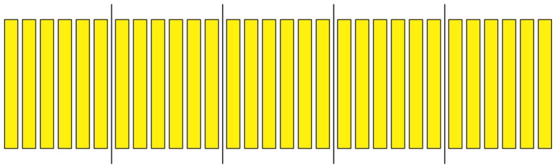


Figure 2. $\mathcal{T}_{(2)}$ flattening matrix 7×30 of the 3^{rd}-order tensor in figure 1.

vectors of a tensor as columns in a matrix, denoted as a subscript (n) after the tensor name: $\mathcal{T}_{(n)}$ or $\mathbf{T}_{(n)}$, figure 2 shows an example based on the third-order tensor of figure 1(c).

A product of two matrices is generalized to a product of a matrix ($\mathbf{M}$) and a tensor ($\mathcal{T}$), which is denominated mode-n product ($\mathcal{T} \times_n \mathbf{M}$). It is a linear transformation on all mode-n vectors for $\mathbf{M}$, then the matrix has dimensions $i \times j$ where $i = d_n$ and d_n is the n^{th} dimension of $\mathcal{T} \in \mathbb{R}^{d_1 \times \cdots \times d_n \times \cdots \times d_N}$, in terms of flattened matrices: $\mathcal{P}_{(n)} = \mathbf{M}\mathcal{T}_{(n)}$. Given a tensor ($\mathcal{T} \in \mathbb{R}^{d_1 \times \cdots \times d_m \times \cdots \times d_n \times \cdots \times d_N}$) and two matrices $\mathbf{M}_{i_m \times j_m}, \mathbf{N}_{i_n \times j_n}$ where $j_m = d_m$ and $j_n = d_n$, $n \neq m$ the following property holds true:

$$\mathcal{T} \times_m \mathbf{M} \times_n \mathbf{N} = \mathcal{T} \times_n \mathbf{N} \times_m \mathbf{M}$$

The *Singular Value Decomposition Theorem* (SVD) [5] states that any matrix $\mathbf{M}$ can be decomposed as:

$$\mathbf{M} = \mathbf{U}\Sigma\mathbf{V}^T$$

where Σ is diagonal and $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, in tensor notation it is written as $\mathbf{M} = \Sigma \times_1 \mathbf{U} \times_2 \mathbf{V}$. The *N-mode SVD* (*higher-order SVD*) is a generalization of the *SVD* assuring that any tensor $\mathcal{T}$ can be decomposed as:

$$\mathcal{T} = \mathcal{C} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \cdots \times_N \mathbf{U}_N$$

where $\mathbf{U}_i$ is the equivalent to matrix $\mathbf{U}$ of SVD on $\mathcal{T}_{(i)} = \mathbf{U}\Sigma\mathbf{V}^T$ (from the *SVD theorem*). Unlike in the SVD, the tensor $\mathcal{C}$ (core tensor) isn't a diagonal tensor (this is an example of some properties which dont generalize well to tensors). The *N-mode SVD algorithm* from [10]:

```
1. For $n = 1, \cdots, N$, compute matrix $\mathbf{U}_n$ by
   computing the SVD of the flattened
   matrix $\mathcal{T}_{(n)}$ and setting $\mathbf{U}_n$ to be the
   left matrix of the SVD
2. Solve for the core tensor ($\mathcal{C}$) as
   follows:
```

$$\begin{aligned} \mathcal{C} &= \mathcal{T} \times_1 \mathbf{U}_1^{-1} \times_2 \mathbf{U}_2^{-1} \times_3 \cdots \times_N \mathbf{U}_N^{-1} \\ &= \mathcal{T} \times_1 \mathbf{U}_1^T \times_2 \mathbf{U}_2^T \times_3 \cdots \times_N \mathbf{U}_N^T \end{aligned}$$

4. Multilinear AAM Expression Transfer

Following the assumption that "*a good analysis is the first step to a good (re)synthesis*", we approach the expression transfer problem with methods developed by the computer vision community for facial expression analysis and recognition [2, 3, 10, 11, 14]. Combining active appearance models and multilinear analysis, it's possible to suitably *represent* and *analyze* expressive facial images, while separating both style (subject's identity) and content (expressive flavor) from the captured performance [8].

4.1. Method overview

Our method is divided in three major steps (the first two are performed off-line): *data acquisition*, *training* and *expression transfer*.

Data acquisition. As an example-based approach, our method relies on a collection of images with different subjects performing a variety of expressions under a controlled capture session. After the image acquisition, the photographs are annotated according to subject *identity*, *expression* performed (*neutral* plus the six basic expressions [4]: *anger*, *fear*, *surprise*, *disgust*, *sadness* and *happiness*) and *landmark points* positions. Figure 3 depicts the positioning of the landmark points on one of our training examples.

Training. Given the annotated images, a two-phase training process begins. First, an *Independent AAM* statistical model of the facial images is built [2, 3, 7]. After that, we have a compact representation for each training example

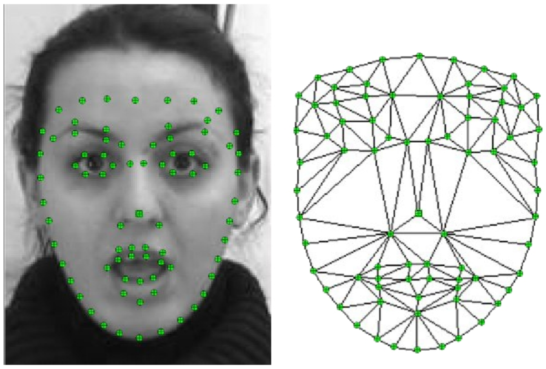


Figure 3. Landmark points and template triangulation.

which are structured in two (*shape* and *appearance*) third-order tensors whose mode spaces correspond to the annotations (*Identity* × *Expression* × *AAM Coefficients*). Applying a *Multilinear Analysis* procedure over these two tensors, we are able to separate the *identity* and *expression* factors hidden in the *coefficients* [10].

Expression transfer. Given a pair of photographs of two subjects performing different expressions, the *Independent AAM*'s coefficients are estimated for each image [7, 1]. These coefficients are then projected on each tensor to separate the identity and expression parameters. After this analysis, the expression transfer process is straightforward: the new AAM coefficients are reconstructed with the expression parameters exchanged between the two subjects and are used in the AAM reconstruction process to give the new shape and appearance of the transfered faces.

In the following, we detail the core steps of our *Multilinear AAM Expression Transfer* method.

4.2. Training (*AAM's* + *Multilinear Analysis*)

Our training methodology is inspired by the work of [12] and highly influenced by the approach presented in [14], althought it is adapted to fit more naturaly with fast AAM parameter estimation methods like the one proposed by [7, 1]. It is subdivided in the training of *Independent AAM*'s, for facial image representation, and *Multilinear Analysis*, to separate style and content in data.

Training Independent AAM's. With the annotated exemplars in hand, the first step in the independent AAM's training process is the *Procrustes analysis* of the landmark points sets (resulting in a database of landmarks aligned to a common coordinate frame, the *shape database*). After that, the exemplar images are warped to the mean of the aligned landmarks sets and [optionaly] have their color space converted, providing a database of aligned facial images, the *appearance database*. A *Principal Component Analysis* (PCA) is performed [separately] in both the shape and appearance databases resulting in:

- a mean shape ($\mathbf{s}_0$), an orthonormal basis ($\{\mathbf{s}_k\}$) for the facial shape space and the coefficients ($\{\sigma_i\}$) for each element from the shape database, with respect to $\{\mathbf{s}_k\}$;
- a mean appearance ($\mathbf{a}_0$), an orthonormal basis ($\{\mathbf{a}_k\}$) for the facial appearance space and the coefficients ($\{\alpha_i\}$) for each element from the appearance database, with respect to $\{\mathbf{a}_k\}$.

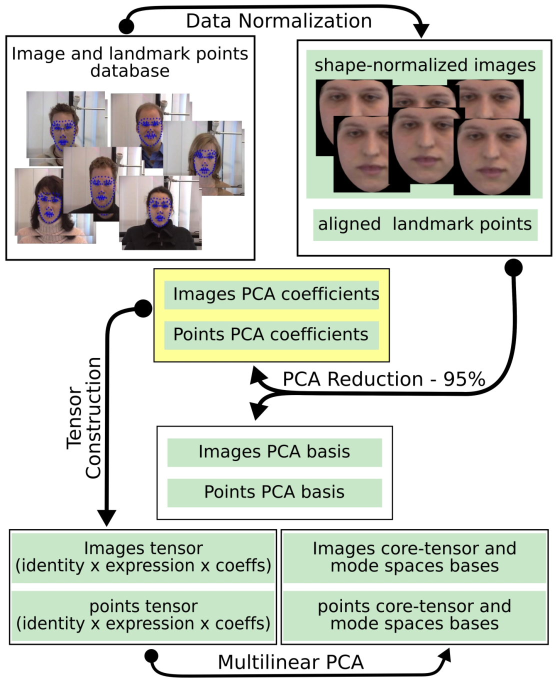

Data Normalization
Image and landmark points database
shape-normalized images
aligned landmark points
Images PCA coefficients
Points PCA coefficients
PCA Reduction - 95%
Tensor Construction
Images PCA basis
Points PCA basis
Images tensor (identity x expression x coeffs)
Images core-tensor and mode spaces bases
points tensor (identity x expression x coeffs)
points core-tensor and mode spaces bases
Multilinear PCA

Figure 4. Training pipeline (Independent AAM's + Multilinear Analysis).

Multilinear Analysis. With the shape and appearance coefficients from the AAM training (and their respective identity and expression annotations), two third-order tensors are built and *Multilinear Analysis* is applied to each of them, providing:

- a third-order (*Identity* × *Expression* × *AAM Shape Coefficients*) *shape core tensor* ($\mathcal{S}$) and two mode-space basis matrices ($\mathbf{U}_{SI}$, for the *identity mode*, and $\mathbf{U}_{SE}$, for the *expression mode*);
- a third-order (*Identity* × *Expression* × *AAM Appearance Coefficients*) *appearance core tensor* ($\mathcal{A}$) and two mode-space basis matrices ($\mathbf{U}_{AI}$, for the *identity mode* and $\mathbf{U}_{AE}$, for the *expression mode*).

The whole training process is depicted in Figure 4.

4.3. Expression Transfer

Our expression transfer procedure relies on the training phase to be able to analyze the input photographs and extract information about the style and content from them, making possible to feed the learned generative statistical model of the face space with the target style (*identity*) and the source content (*expression*). The proposed method is subdivided in two basic steps: *analysis* and *(re)synthesis*.

Analysis. To transfer the expression from one performance to another, it's necessary to extract this expression from the image (separated from the identity "footprint" of the person that is performing it). Therefore, we estimate the identity and expression parameters that best "explain" the input photographs (as well as the location, alignment parameters, of the faces in these images).

Independent AAM fitting. The first step in the analysis process consists in fitting the AAM coefficients (and the alignment transformation matrix $\mathbf{T}$) to the inputs[1]. Methods like the one proposed by [7, 1] estimate all these parameters simultaneous and automaticaly by an optimization procedure. As a proof-of-concept system, we have implemented this step as semi-automatic process: a user adjusts the template triangulation (depicted in Figure 3) to both photographs and the system aligns the resulting mesh to the mean shape acquired in the training process and warps the images to this mean (in the same manner as done at the training step). After that, both the shape and appearance coefficients are calculated by projecting the aligned shape (minus $\mathbf{s}_0$) and warped image (minus $\mathbf{a}_0$) onto their respective basis ($\{\mathbf{s}_k\}$ and $\{\mathbf{a}_k\}$), computed in the training process, this procedure is detailed below:

$$\begin{aligned}\sigma &= \mathbf{S}^T \cdot (\mathbf{s} - \mathbf{s}_0)\\ \alpha &= \mathbf{A}^T \cdot (\mathbf{a} - \mathbf{a}_0)\end{aligned}$$

where σ and α are, respectively, the shape and appearance coefficients (i.e. the projection of $\mathbf{s}$, the aligned user-marked shape, and $\mathbf{a}$, the warped image), $\mathbf{S}$ and $\mathbf{A}$ are matrices whose columns comprise the orthonormal bases for the shape and appearance spaces learned in the training process.

Tensor projection. Having the AAM shape and appearance coefficients, a second projection is performed to separate the identity and expression parameters from them. This step consists of calculating $\mathbf{i}_s, \mathbf{e}_s, \mathbf{i}_a$ and $\mathbf{e}_a$ such that:

$$(\mathbf{i}_s, \mathbf{e}_s) = \underset{(\mathbf{i},\mathbf{e})}{\operatorname{argmin}} \|\sigma - \mathcal{S} \times_I \mathbf{i} \times_E \mathbf{e}\|$$

$$(\mathbf{i}_a, \mathbf{e}_a) = \underset{(\mathbf{i},\mathbf{e})}{\operatorname{argmin}} \|\alpha - \mathcal{A} \times_I \mathbf{i} \times_E \mathbf{e}\|$$

We have tried the "projection tensor" operator of [11] but had many problems with its quality. Further analyzing this operator, we verified that the tensors resulting from an application of the "projection tensor" have full effective rank, so rank one approximations of them tend to produce poor projection results. To overcome this issue, we have developed a method that requires from the user a "calibration image" for each performer where he is photographed making one of the six basic expressions (in our implementation, we require a neutral expression photograph for each performer). From each calibration image, we extract its AAM coefficients (following the same procedure described previously) and calculate the identity vectors ($\mathbf{i}_s$ and $\mathbf{i}_a$) by solving the linear systems (by standard linear least squares):

$$\begin{aligned}\sigma_{calibration} &= \mathcal{S} \times_I \mathbf{i}_s \times_E \mathbf{n}_s\\ &= (\mathcal{S} \times_E \mathbf{n}_s) \times_I \mathbf{i}_s\\ &= I_s(\mathbf{i}_s)\\ \alpha_{calibration} &= \mathcal{A} \times_I \mathbf{i}_a \times_E \mathbf{n}_a\\ &= (\mathcal{A} \times_E \mathbf{n}_a) \times_I \mathbf{i}_a\\ &= I_a(\mathbf{i}_a)\end{aligned}$$

where $\mathbf{n}_s$ and $\mathbf{n}_a$ denote, respectively, the columns of $\mathbf{U}_{SE}$ and $\mathbf{U}_{AE}$ that correspond to the neutral expression, I_s and I_a are the *linear* operators defined as:

$$\begin{aligned}I_s(\mathbf{i}_s) &= (\mathcal{S} \times_E \mathbf{n}_s) \times_I \mathbf{i}_s\\ I_a(\mathbf{i}_a) &= (\mathcal{A} \times_E \mathbf{n}_a) \times_I \mathbf{i}_a\end{aligned}$$

With the identity coefficients of each performer, we are able to calculate the expression vectors ($\mathbf{e}_s$ and $\mathbf{e}_a$) for any image in an analogous manner to that applied in the computation of i_s and i_a:

$$\begin{aligned}\sigma &= (\mathcal{S} \times_I \mathbf{i}_s) \times_E \mathbf{e}_s\\ &= E_s(\mathbf{e}_s)\\ \alpha &= (\mathcal{A} \times_I \mathbf{i}_a) \times_E \mathbf{e}_a\\ &= E_a(\mathbf{e}_a)\end{aligned}$$

where E_s and E_a are the *linear* operators defined as:

$$\begin{aligned}E_s(\mathbf{e}_s) &= (\mathcal{S} \times_I \mathbf{i}_s) \times_E \mathbf{e}_s\\ E_a(\mathbf{e}_a) &= (\mathcal{A} \times_I \mathbf{i}_a) \times_E \mathbf{e}_a\end{aligned}$$

Therefore, by requiring one calibration image for each different performer, we are able to calculate $\mathbf{i}_s, \mathbf{e}_s, \mathbf{i}_a$ and $\mathbf{e}_a$ solving just a couple of overconstrained linear systems. Note that, if the performers don't change in repeated sessions, just two *linear* least squares must be computed for each expression transfer (i.e. the calibration step needs to be done just once).

[1] Note that, if the example images had their color spaces converted prior to the AAM training process, the inputs to the expression transfer procedure must have their color spaces converted accordingly.

(Re)synthesis. After the analysis phase, the process of expression transfer is straightforward. It consists of using the target identity coefficients ($\mathbf{i}_s^t$ and $\mathbf{i}_a^t$) along with the source expression vectors ($\mathbf{e}_s^s$ and $\mathbf{e}_a^s$) to reconstruct the new AAM parameters that will be responsible to generate the new facial image.

AAM parameters reconstruction. The reconstruction of the new AAM parameters (σ' and α') can be performed by evaluating the multilinear operators $\mathcal{S}$ and $\mathcal{A}$ on the pairs $(\mathbf{i}_s^t, \mathbf{e}_s^s)$ and $(\mathbf{i}_a^t, \mathbf{e}_a^s)$. This calculation is realized as two mode products for each core tensor:

$$\sigma' = \mathcal{S} \times_I \mathbf{i}_s^t \times_E \mathbf{e}_s^s$$
$$\alpha' = \mathcal{A} \times_I \mathbf{i}_a^t \times_E \mathbf{e}_a^s$$

Facial image resynthesis. At this point, we have both the shape and appearance coefficients for an image of the target identity performing the source expression. To reconstruct the aligned shape ($\mathbf{s}'$) and the normalized appearance ($\mathbf{a}'$), we need to evaluate the trained active appearance model on the calculated parameters:

$$\mathbf{s}' = \mathbf{s}_0 + \mathbf{S} \cdot \sigma'$$
$$\mathbf{a}' = \mathbf{a}_0 + \mathbf{A} \cdot \alpha'$$

with this data[2] (and the alignment transformation $\mathbf{T}'$, extracted in the analysis step), we are able to calculate the location of the new landmark points and warp $\mathbf{a}'$ to it. Overlaying this warped image on the input photograph, the expression transfer task is done. Figure 5 depicts this process.

5. Results

We have performed a number of different experiments to evaluate the quality of our method. Each one of these was designed to assess a specific step in our pipeline and was executed on a large number of cases. After describing the training and test sets, we discuss some of the representative results obtained on each experiment.

Test images. The image database we have used is a subset of *"The FG-Net Facial Expressions and Emotions Database"*, kindly provided by Prof. Frank Wallhof from the *Technische Universität München* [13]. It features images of 19 subjects performing the 6 basic expressions (plus neutral) rougly under the same illumination conditions (but with "very soft" restrictions to head pose). From this data set, we used the images of 17 subjects to train our model and the remaining 2 to apply the test experiments (i.e. all the presented results were evaluated on subjects that weren't used in the training process). Figure 6 shows some of the original test images.

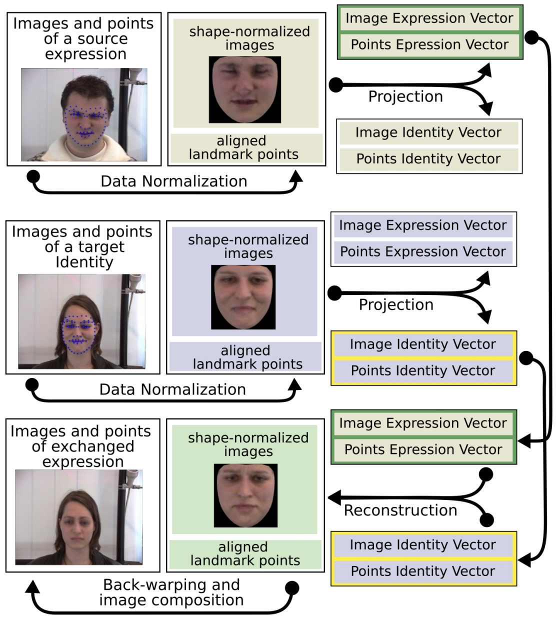


Figure 5. Multilinear AAM Expression Transfer pipeline.

Note: All these experiments were performed on 6 differently trained models. The tested settings varied in the color space processing prior to training (*RGB* or *luminance*) and in the amount of total variance kept in the two principal component analyses (*95%*, *98%* or *100%*). To be more practical, without sacrificing method, the results shown in this paper were drawn from those in that the PCAs kept only *95%* of the total variance.

Projections. In this experiment we evaluate the quality of using our combination of AAM's and Multilinear Analysis to model the expressive facial images space. The inputs are processed by the analysis phase in the expression transfer procedure but are resynthesized with their own estimated identity and expression vectors (the swap of expression coefficients is not applied). This experiment allows us to assess the amount of loss induced by the analysis phase. Figure 7 exhibit the projections of the test images in the figure 6.

"Basic expression" assignment. The intent of this experiment is to evaluate both the ability of the analysis phase in separating style and content in a given image, as well as the quality of resynthesizing a performance with one of the basic expressions. To this end, the inputs are processed by the analysis phase in the expression transfer procedure but are resynthesized with the coefficients of the basic expressions

[2] Again, if the example images had their color spaces converted prior to the AAM training process, the output $\mathbf{a}'$ must have its color space converted accordingly.

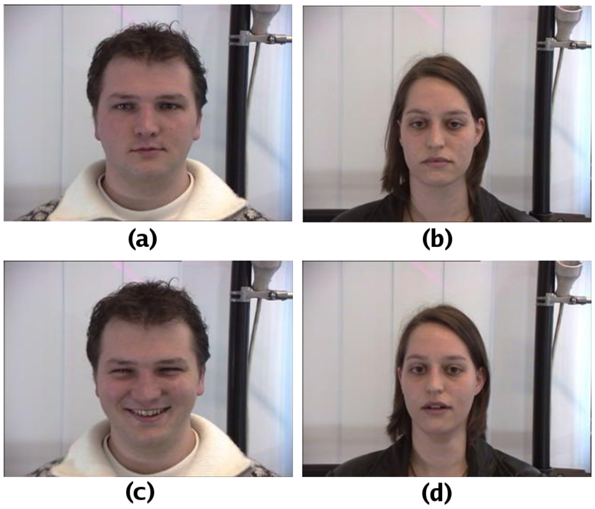


Figure 6. Original test photographs.

(the columns of $\mathbf{U}_{SE}$ and $\mathbf{U}_{AE}$). Figure 8 shows some assignments.

Expression transfer. This experiment is the final application of our algorithm. Photographs of both test subjects are taken where each one is performing a different expression, the goal is to swap the expression between them in such a way that the underlying emotion might be "perceived" as natural in the resynthesized performer's face (not just a direct mapping of facial motions). Figure 9 shows some of our results.

6. Conclusion

We have presented a method to transfer expressions between photographs based on a combination of *active appearance models* and *multilinear analysis*. As evidenced by our results, our data-driven approach recognizes expressions and resynthesizes color images of faces from different unknown subjects. As little difference was observed between *RGB* and *L*a*b** transfers, and working with luminance-based AAM's saves about $\frac{2}{3}$ of storage requirements, a good balance between quality and resource consumption can be achieved with an *L*a*b**-based approach.

The use of methods designed primarily for analysis is a good starting point for example-based resynthesis, but our choices incur some limitations (to be worked on). Active appearance models provide an effective way to represent facial images, however their fitting is very sensitive to variations in illumination conditions. Although multilinear analysis is a very good approach to model problems involving multiple factors, a great amount of data is needed to complete a full tensor, leading to various challenges in data acquisition, storage and management.

Future work. Experiment with video rewrite and puppetry (and problems related to the modeling of the complex dynamics of facial expressions). Evaluate a rigid-motion invariant representation for feature-points (eliminate spurious variations from the alignment transformations). Incorporate a new viseme mode to our model (comprising a 4^{th}-order tensor) in a manner similar to that proposed in [12]. Evaluate the quality of other non-linear dimensionality reduction methods. Extend theses approaches to model and animate 3D puppets from monocular video streams.

Acknowledgements. The FGNet-Consortium and Prof. Frank Wallhof, for the *Facial Expressions and Emotions Database from Technical University of Munich (FEEDTUM)*. This research has been developed in the VISGRAF Laboratory at IMPA. The Laboratory is supported by the agencies CAPES, CNPq, FINEP, and FAPERJ and also by gifts from the companies IBM Brasil and Alias.

References

[1] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, Feb. 2004.

[2] T. Cootes, C. Taylor, D. Cooper, and J. Graham. Active shape models: Their training and application. 61(1):38–59, January 1995.

[3] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *5th European Conference on Computer Vision (ECCV 1998)*, pages 484–498, June 1998.

[4] P. Ekman and W. V. Friesen. Pictures of facial affect. Palo Alto, CA, 1978.

[5] G. H. Golub and C. F. Van Loan. *Matrix Computations,* 2nd ed. Johns Hopkins University Press, Baltimore, 1989.

[6] Z. Liu, Y. Shan, and Z. Zhang. Expressive expression mapping with ratio images. *In Computer Graphics, Annual Conference Series - Siggraph*, pages 271–276, 2001.

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

[8] J. B. Tenenbaum and W. T. Freeman. Separating style and content. In *NIPS*, pages 662–668, 1996.

[9] M. Turk and A. Pentland. Face recognition using eigenfaces. *Journal of Cognitive Neuroscience*, 3:71–86, 1991.

[10] M. A. O. Vasilescu and D. Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. In *7th European Conference on Computer Vision*, pages 447–460, 2002.

[11] M. A. O. Vasilescu and D. Terzopoulos. Multilinear independent components analysis. In *2005 Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, pages 547–553, June 2005.

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

[13] F. Wallhoff and FG-Net. Facial Expressions and Emotions Database from Technical University of Munich, 2005.

[14] H. Wang and N. Ahuja. Facial expression decomposition. In *ICCV*, pages 958–965, 2003.

[15] Q. Zhang, Z. Liu, B. Guo, D. Terzopoulos, and H. Shum. Geometry-driven photorealistic facial expression synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 12(1):48–60, 2006.

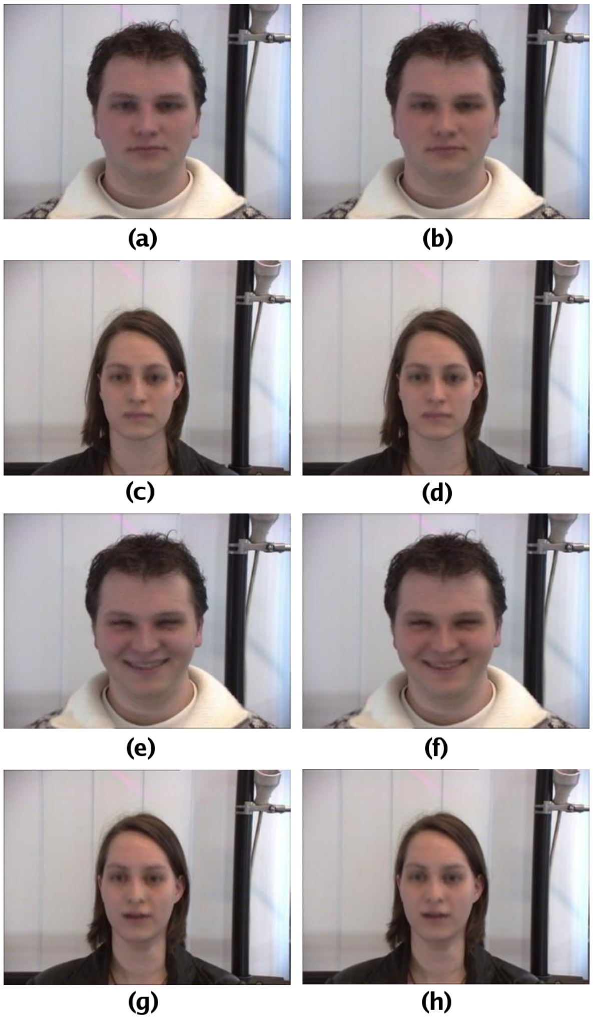


Figure 7. Projections of test images from figure 6. Right column: *L*a*b* luminance* database; Left column: *RGB* trained model.

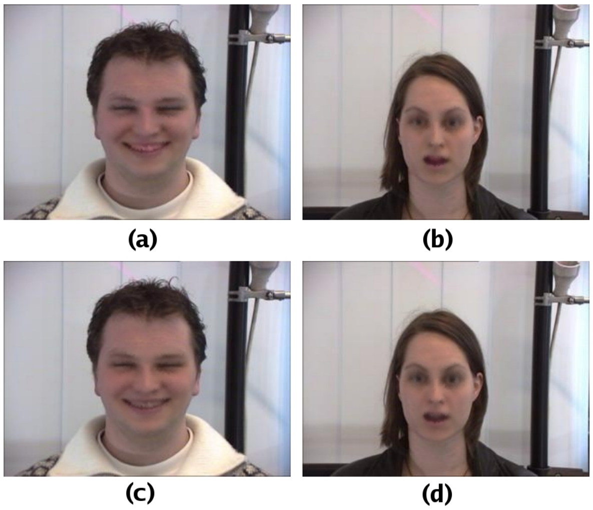


Figure 8. Assigning a happy expression to the neutral image of the male subject at “figure 6 (a)” and an surprise to the female’s at “figure 6 (b)”; Top row: *L*a*b* luminance* trained model; Bottom row: *RGB* trained model.

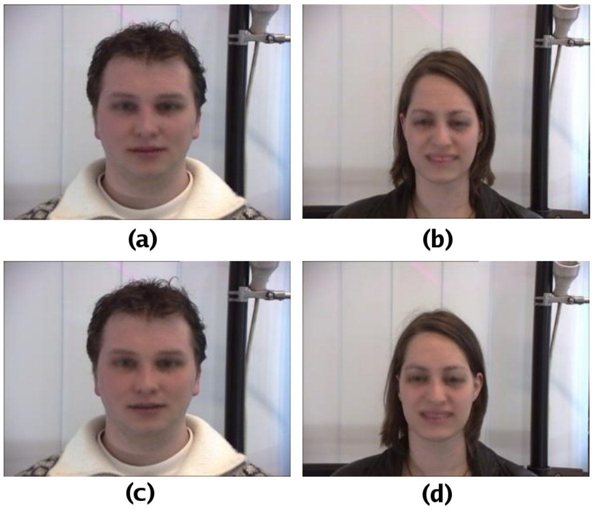


Figure 9. Exchanging the expressions performed between both subjects “figure 6 (c) ⇄ figure 6 (d)”; Top row: *L*a*b* luminance* trained model; Bottom row: *RGB* trained model.