A Multi-Affine Model for Tensor Decomposition

Yiqing Yang
UW Madison
breakds@cs.wisc.edu

Li Zhang
UW Madison
lizhang@cs.wisc.edu

Sen Wang
Kodak Research Laboratories
sen.wang@kodak.com

Hongrui Jiang
UW Madison
hongrui@engr.wisc.edu

Chris J. Murphy
UC Davis
cjmurphy@ucdavis.edu

Jim Ver Hoeve
UW Madison
verhoeve@wisc.edu

Abstract

Higher-order Singular Value Decomposition (HOSVD) for tensor decomposition is widely used in multi-variate data analysis, and has shown applications in several areas in computer vision in the last decade. Conventional multi-linear assumption in HOSVD is not translation invariant — translation in different tensor modes can yield different decomposition results. The translation is difficult to remove as preprocessing when the tensor data has missing data entries. In this paper we propose a more general multi-affine model by adding appropriate constant terms in the multi-linear model. The multi-affine model can be computed by generalizing the HOSVD algorithm; the model performs better for filling in missing values in data tensor during model training, as well as for reconstructing missing values in new mode vectors during model testing, on both synthetic and real data.

1. Introduction

Matrix decomposition for linear dimension reduction is a widely used technique in high dimensional data analysis. A matrix is a two-dimensional array; as a generalization of matrix decomposition, tensor decomposition aims to decompose a multi-dimensional array, a.k.a., an order-N data tensor. The decomposition result produces a model that expresses each data entry (scalar or vector) as a multi-linear function of parameters in several lower dimensional subspaces.

The conventional multi-linear model does not include constant terms, making the model estimation sensitive to large translation offset in the input data. In matrix decomposition, Principle Component Analysis (PCA) deals with the translation offset by incorporating a mean vector in the model. In the case of tensor decomposition, the removal of translation offset (or mean vector) is more intriguing as each mode of the input tensor can have its own translation offset. The estimation of the translation offsets can be further complicated by the possible missing values in the input data tensor, which are quite common in computer vision applications.

In this paper we present a multi-affine model that handles the translation offset in each tensor mode in a principled fashion. Using Taylor expansion, we derive our multi-affine model as a generalization of PCA with mean removal. From the Taylor expansion perspective, we can show what the nonlinear effects are that are neglected by both the multi-linear and multi-affine model. We formulate the model training as an optimization problem and solve it using an alternating SVD algorithm similar to the multi-linear HOSVD. With small modifications, we extend this algorithm to deal with data tensor with missing data.

The estimated model parameterizes vectors in each mode as a multi-affine function of lower dimensional vectors. Given a new mode vector with missing and noisy data entries, this model allows us to fill in the missing values. We show that the multi-affine model outperforms the multi-linear model for this data completion task, especially when the noise level and missing data rate is high.

1.1. Denotations

Throughout this paper, we use different font styles to distinguish between scalars, vectors, matrices and tensors:

$x, y, z, \cdots$	as scalars
$\mathbf{x}, \mathbf{y}, \mathbf{z}, \cdots$	as vectors
$X, Y, Z, \cdots$	as matrices
$\mathcal{X}, \mathcal{Y}, \mathcal{Z} \cdots$	as tensors

We use $\mathbf{1}$ to denote the column vector $\begin{bmatrix}1 & 1 & \cdots & 1\end{bmatrix}^{\mathrm{T}}$. In some places we specify the dimension of $\mathbf{1}$ as m by using $\mathbf{1}_m$. We use

$$\hat{\mathbf{1}}_m = \frac{\mathbf{1}_m}{|\mathbf{1}_m|} \tag{1}$$

to denote the normalized $\mathbf{1}_m$.

2. Related Work

Tensor decomposition has been well studied in the field of applied mathematics [4, 3]. Give an order-N data tensor $\mathcal{T}$, the decomposition aims to find an outer product representation of $\mathcal{T}$,

$$\mathcal{T} = \mathcal{Q} \times_1 U_1 \times_2 U_2 \cdots \times_N U_N \tag{2}$$

where $\mathcal{Q}$ is also an order-N tensor and $\{U_i\}$ are matrices.

Tucker's first method is the most widely used algorithm for multi-linear tensor decomposition, in which the decom-

position is achieved by minimizing the error term with respect to $\mathcal{Q}$ and $\{U_i\}$

$$\min_{\mathcal{Q},\{U_i\}} |\mathcal{T} - \mathcal{Q} \times_1 U_1 \times_2 U_2 \cdots \times_N U_N|^2 \tag{3}$$

The algorithm minimizes this error term by updating each U_i alternately while keeping other $U_j, j \neq i$ fixed until it converges. It is better known as Higher-Order Singular Value Decomposition (HOSVD) today due to the work of De Lathauwer et al. [4].

Tensor analysis has found applications in many computer vision and graphics problems, such as general factor analysis [5], geometric analysis [1], texture modeling [7, 10], face image recognition [9], 3D face transfer [8], and most recently face contour modeling [6].

All these works use a multi-linear model to represent their input data. Two common operations in these works are training models with missing data and expressing a mode vector as a multi-linear function of low dimensional parameter vectors. Our multi-affine model can potentially benefit all these application problems. In our experiments, we show that it works better than a multi-linear model for filling in missing values in 2D face landmark data.

3. Multi-affine Model

We assume input data is sampled from a continuous unknown function. The function can be a scalar or a vector valued function. Using Taylor expansion, we can derive an affine model for the first order approximation and a multi-affine model for the higher order approximation. The affine model can be estimated using PCA and the multi-affine model and its estimation are our major contributions.

3.1. First Order: Affine Model

A real valued scalar function f can be approximated by its first order Taylor series if we ignore the higher order terms in its Taylor expansion:

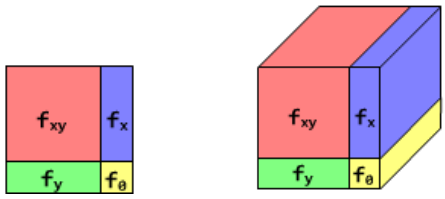
$$f(\mathbf{x}) \approx f(0) + \nabla f(0)^{\mathrm{T}}\mathbf{x} = \begin{bmatrix} f_{\mathbf{x}} & f_0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \tag{4}$$

where $f_{\mathbf{x}} = \nabla f(0)^{\mathrm{T}}$, and $f_0 = f(0)$. This result can be generalized for a vector function $\mathbf{f}$ as

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{f}_0 + \mathbf{f}_{\mathbf{x}}\mathbf{x} = \begin{bmatrix} \mathbf{f}_{\mathbf{x}} & \mathbf{f}_0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \tag{5}$$

where $\mathbf{f}_{\mathbf{x}}$ is the Jacobian matrix of $\mathbf{f}$ at 0, and $\mathbf{f}_0 = \mathbf{f}(0)$.

Let $\{\mathbf{y}_i = \mathbf{f}(\mathbf{x}_i)\}$ be a set of samples of the function $\mathbf{f}$. We form two matrices $Y = \begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_n \end{bmatrix}$, $X = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \end{bmatrix}$ and assume $\{\mathbf{x}_i\}$ is distributed symmetrically around 0, i.e., $\sum_{i=1}^{n} \mathbf{x}_i = 0$. Note that this assumption is general, because we can redefine $\mathbf{f}$ by shfiting the origin to the centroid of all $\{\mathbf{x}_i\}$. Under this assumption, we can estimate $\mathbf{f}_0 = \overline{\mathbf{y}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{y}_i$ and apply SVD on $\widetilde{Y} = Y - \overline{\mathbf{y}} \cdot \mathbf{1}^{\mathrm{T}}$ to get $\mathbf{f}_{\mathbf{x}}$. Note that each column in $\widetilde{Y}$ is $\mathbf{y}_i - \overline{\mathbf{y}}$, which is simply the result of the mean removal operation in PCA.



(a) Core Matrix (b) Core Tensor

Figure 1: (a) The matrix that serves as the core matrix in Eq. (6) for scalar functions. (b) The core matrix for scalar functions now becomes a core tensor.
Best viewed electronically in color.

3.2. Higher Order: Multi-Affine Model

We now discuss higher order approximation of the function. To simplify notation but without loss of generality, we use an order-3 tensor as an example to illustrate multi-affine tensor analysis.

Consider a real valued function $f(\mathbf{x}, \mathbf{y})$ with two groups of variables. Taking into account the second order terms, Eq. (4) becomes:

$$\begin{aligned} f(\mathbf{x}, \mathbf{y}) \approx & f_0 + f_{\mathbf{x}}^{\mathrm{T}}\mathbf{x} + f_{\mathbf{y}}^{\mathrm{T}}\mathbf{y} \\ & + \frac{1}{2} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} f_{\mathbf{xx}} & f_{\mathbf{xy}} \\ f_{\mathbf{xy}} & f_{\mathbf{yy}} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \end{aligned} \tag{6}$$

where $f_{\mathbf{x}}$, $f_{\mathbf{y}}$ are the first order partial derivatives of f at 0, $f_0 = f(0, 0)$ and $f_{\mathbf{xx}}$, $f_{\mathbf{yy}}$, $f_{\mathbf{xy}}$ are the second order partial and mixed derivatives of f at 0.

Note that in Eq. (6) $f_{\mathbf{xy}}$ captures the interaction between $\mathbf{x}$ and $\mathbf{y}$. Assuming f's second order partial derivatives at 0 with respect to $\mathbf{x}$ and $\mathbf{y}$ is small, i.e., $f_{\mathbf{xx}} \approx 0$ and $f_{\mathbf{yy}} \approx 0$, we ignore these two terms in Eq. (6) and keep only the mixed derivative $f_{\mathbf{xy}}$, and the equation then becomes

$$f(\mathbf{x}, \mathbf{y}) \approx \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^{\mathrm{T}} \underbrace{\begin{bmatrix} f_{\mathbf{xy}} & f_{\mathbf{x}} \\ f_{\mathbf{y}} & f_0 \end{bmatrix}}_{Q} \begin{bmatrix} \mathbf{y} \\ 1 \end{bmatrix} \tag{7}$$

The structure of the core matrix Q in Eq. (7) is illustrated in Figure 1a. If $\mathbf{f}(\mathbf{x}, \mathbf{y})$ is a vector valued function, Eq. (7) is valid for each component of $\mathbf{f}$. Stacking the matrix Q in Eq. (7) together forms an order-3 tensor $\mathcal{Q}$ (e.g., a cube, whose structure is illustrated in Figure 1b. We can then write Eq. (7) for the vector valued $\mathbf{f}(\mathbf{x}, \mathbf{y})$ using the tensor product as follows:

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) \approx \mathcal{Q} \times_1 \begin{bmatrix} \mathbf{x}^{\mathrm{T}} & \mathbf{1} \end{bmatrix} \times_2 \begin{bmatrix} \mathbf{y}^{\mathrm{T}} & \mathbf{1} \end{bmatrix} \tag{8}$$

If we have a set of samples of $\{\mathbf{f}(\mathbf{x}_i, \mathbf{y}_j)\}$ with $i = 1 \cdots m_i$ and $j = 1 \cdots m_j$, we can combine Eq. (8) for all

$\{(\mathbf{x}_i, \mathbf{y}_j)\}$ into a single tensor product equation as follows. Define an order-3 tensor $\mathcal{T}$ as

$$\mathcal{T}(i, j, :) = \mathbf{f}(\mathbf{x}_i, \mathbf{y}_j) \tag{9}$$

then,

$$\mathcal{T} \approx \mathcal{Q} \times_1 \begin{bmatrix} X^{\mathrm{T}} & \mathbf{1} \end{bmatrix} \times_2 \begin{bmatrix} Y^{\mathrm{T}} & \mathbf{1} \end{bmatrix} \tag{10}$$

where the matrix $X = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{m_1} \end{bmatrix}$, and the matrix $Y = \begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_{m_2} \end{bmatrix}$.

In general, we might also want to decompose the data tensor $\mathcal{T}$ along mode-3, i.e., mode-3 is no longer modeled as a special mode. The vector valued function $\mathbf{f}(\mathbf{x}, \mathbf{y})$ is now viewed as a trivariate scalar valued function $f(\mathbf{x}, \mathbf{y}, \mathbf{z})$, where $\mathbf{z}$ is the component index of $\mathbf{f}$. Then Eq. (10) becomes

$$\mathcal{T} = \mathcal{Q} \times_1 \begin{bmatrix} X^{\mathrm{T}} & \mathbf{1} \end{bmatrix} \times_2 \begin{bmatrix} Y^{\mathrm{T}} & \mathbf{1} \end{bmatrix} \times_3 \begin{bmatrix} Z^{\mathrm{T}} & \mathbf{1} \end{bmatrix} \tag{11}$$

where Z is defined in a similar way as X and Y in Eq. (10).

Performing the original HOSVD [4] on $\mathcal{T}$ does not give a solution to the multi-affine model in Eq. (11), because of the additional $\mathbf{1}$ vectors. In the next section, we show how to extend HOSVD for the multi-affine model.

4. Training Algorithms

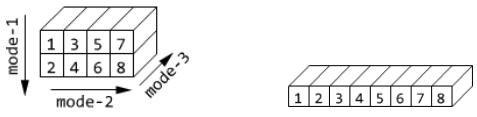
We now define the multi-affine model training problem. Given an input order-N tensor $\mathcal{T}$, we seek to find a core tensor $\mathcal{Q}$ and matrices $U_1, U_2, \cdots, U_N$ that minimizes the following decomposition error

$$\begin{aligned} &\min_{\mathcal{Q}, \{U_i\}} \left| \mathcal{T} - \mathcal{Q} \times_1 \begin{bmatrix} U_1 & \hat{\mathbf{1}}_{m_1} \end{bmatrix} \cdots \times_N \begin{bmatrix} U_N & \hat{\mathbf{1}}_{m_N} \end{bmatrix} \right|^2 \\ &\text{s.t.} \quad \mathbf{1}_{m_i}^{\mathrm{T}} U_i = 0, \quad i = 1, 2, \cdots, N \quad \text{(zero mean)} \\ &\qquad U_i^{\mathrm{T}} U_i = I, \quad i = 1, 2, \cdots, N \quad \text{(orthonormal)} \end{aligned} \tag{12}$$

where $\mathcal{T}$ is an $m_1 \times m_2 \times \cdots \times m_N$ tensor, $\mathcal{Q}$ is a $(k_1 + 1) \times (k_2 + 1) \times \cdots \times (k_N + 1)$ tensor, U_i is of size $m_i \times k_i$ and $m_i > k_i$ for all i. The constraints are added to reduce the ambiguity between core tensor $\mathcal{Q}$ and the basis matrices $\{U_i\}$; they also lead to efficient model estimation as shown later in Eq. (14) in this section. $\hat{\mathbf{1}}_{m_i}$ is defined in Eq. (1). Under these constraints, every matrix $\begin{bmatrix} U_i & \hat{\mathbf{1}}_{m_i} \end{bmatrix}$ is orthonormal.

If there are missing elements in $\mathcal{T}$ (denoted as $\mathcal{T}_{\mathrm{U}}$), we also count them as the free variables of the optimization problem. In this case Eq. (12) becomes

$$\begin{aligned} &\min_{\mathcal{Q}, \{U_i\}, \mathcal{T}_{\mathrm{U}}} \left| \mathcal{T} - \mathcal{Q} \times_1 \begin{bmatrix} U_1 & \hat{\mathbf{1}}_{m_1} \end{bmatrix} \cdots \times_N \begin{bmatrix} U_N & \hat{\mathbf{1}}_{m_N} \end{bmatrix} \right|^2 \\ &\text{s.t.} \quad \mathbf{1}_{m_i}^{\mathrm{T}} U_i = 0, \quad i = 1, 2, \cdots, N \quad \text{(zero mean)} \\ &\qquad U_i^{\mathrm{T}} U_i = I, \quad i = 1, 2, \cdots, N \quad \text{(orthonormal)} \end{aligned} \tag{13}$$



(a) Before Unfold (b) After Unfold

Figure 2: Illustration of a mode-3 unfold operation. In general, the mode-i unfold operation on a tensor $\mathcal{T}$ aligns all the mode-i vectors of $\mathcal{T}$ in an index order, producing a large matrix that has the same elements as in $\mathcal{T}$.

4.1. Decomposition without Missing Data

We first present the algorithm to solve the decomposition problem without missing data, as defined in Eq. (12). Our algorithm is a variant of HOSVD [4]. Specifically we iteratively update one U_i while fixing other $U_j, j \neq i$ till convergence. However, special care must be taken to satisfy the new constraints in Eq. (12).

We initialize all U_i with random orthonormal matrices that are also orthogonal to $\hat{\mathbf{1}}_{m_i}$. Since all $\begin{bmatrix} U_i & \hat{\mathbf{1}}_{m_i} \end{bmatrix}$ are orthonormal, updating U_1 while fixing other $U_i, i \neq 1$ leads to the following sub-problem of Eq. (12),

$$\begin{aligned} &\arg\min_{\mathcal{Q}, U_1} \left| \mathcal{T} - \mathcal{Q} \times_1 \begin{bmatrix} U_1 & \hat{\mathbf{1}}_{m_1} \end{bmatrix} \cdots \times_N \begin{bmatrix} U_N & \hat{\mathbf{1}}_{m_N} \end{bmatrix} \right|^2 \\ &= \arg\min_{\mathcal{Q}, U_1} \left| \mathcal{S} - \mathcal{Q} \times_1 \begin{bmatrix} U_1 & \hat{\mathbf{1}}_{m_1} \end{bmatrix} \right|^2 \end{aligned} \tag{14}$$

where $\mathcal{S} = \mathcal{T} \times_2 \begin{bmatrix} U_2 & \hat{\mathbf{1}}_{m_2} \end{bmatrix}^{\mathrm{T}} \cdots \times_N \begin{bmatrix} U_N & \hat{\mathbf{1}}_{m_N} \end{bmatrix}^{\mathrm{T}}$

The sub-problem becomes

$$\begin{aligned} &\min_{Q, U_1} \left| S - \begin{bmatrix} U_1 & \hat{\mathbf{1}}_{m_1} \end{bmatrix} Q \right|^2 \\ &\text{s.t.} \quad \mathbf{1}_{m_1}^{\mathrm{T}} U_1 = 0, \quad U_1^{\mathrm{T}} U_1 = I \end{aligned} \tag{15}$$

where

$$\begin{aligned} S &= \text{unfold}_1(\mathcal{S}) \\ Q &= \text{unfold}_1(\mathcal{Q}) \end{aligned} \tag{16}$$

In the above equation, the unfold operation of a tensor on mode-i (unfold$_i$) converts a tensor to a matrix, as illustrated in Figure 2.

To solve this sub-problem, we need to define a mean removal operation on matrices first. Let $\bar{\mathbf{s}}^{\mathrm{T}}$ be the row vector containing the mean values of each column of S (in Matlab notation, $\bar{\mathbf{s}}^{\mathrm{T}} = \text{mean}(S, 1)$), and define row-mean-removal operation RM() on matrix S as

$$\text{RM}(S) = S - \mathbf{1} \cdot \bar{\mathbf{s}}^{\mathrm{T}} \tag{17}$$

Under this definition, S can be separated as

$$S = \text{RM}(S) + \mathbf{1} \cdot \bar{\mathbf{s}}^{\mathrm{T}} \tag{18}$$

Similarly, after applying RM() on the second term in Eq. (15), it is separated as

$$\begin{bmatrix} U_1 & \hat{\mathbf{1}}_{m_1} \end{bmatrix} Q = U_1 Q_{\text{top}} + \hat{\mathbf{1}}_{m_1} Q_{\text{bottom}} \tag{19}$$

where $Q_{\text{top}} = Q(1{:}k_1, :)$ and $Q_{\text{bottom}} = Q(k_1+1, :)$ following the Matlab notation. Plug in Eq. (18) and Eq. (19), and the objective function Eq. (15) becomes

$$\left|\text{RM}(S) + \mathbf{1} \cdot \bar{\mathbf{s}}^{\text{T}} - U_1 Q_{\text{top}} - \hat{\mathbf{1}}_{m_1} Q_{\text{bottom}}\right|^2 \tag{20}$$

Note that the elements of each column vector $\mathbf{a}$ in both $\text{RM}(S)$ and $U_1 Q_{\text{top}}$ sum to zero, *i.e.*, $\mathbf{1}^{\text{T}}\mathbf{a} = 0$; so $\mathbf{a}$ is orthogonal to all the column vectors in $\mathbf{1}\cdot\bar{\mathbf{s}}^{\text{T}}$ and $\hat{\mathbf{1}}_{m_1} Q_{\text{bottom}}$. As a result, the objective function Eq. (20) can be split as

$$\left|\text{RM}(S) - U_1 Q_{\text{top}}\right|^2 + \left|\mathbf{1} \cdot \bar{\mathbf{s}}^{\text{T}} - \hat{\mathbf{1}}_{m_1} Q_{\text{bottom}}\right|^2 \tag{21}$$

U_1 depends only on the first term in Eq. (20), and can be solved by minimizing the following objective function.

$$\begin{aligned} &\min_{Q, U_1} \left|\text{RM}(S) - U_1 Q_{\text{top}}\right|^2 \\ &\text{s.t.} \quad \mathbf{1}_{m_1}^{\text{T}} U_1 = 0, \quad U_1^{\text{T}} U_1 = I \end{aligned} \tag{22}$$

This optimization problem can be solved by matrix SVD

$$\text{RM}(S) = UVW^{\text{T}} \tag{23}$$

and updating U_1 as $U_1 = U$. Note that the constraints on U_1 automatically hold:

- Since the elements of each column vectors in $\text{RM}(S)$ sum to zero, i.e., $\mathbf{1}^{\text{T}}\text{RM}(S) = 0$, U_1 must also satisfy $\mathbf{1}^{\text{T}} U_1 = 0$, provided the number of columns of U_1 is no larger than the rank of matrix $\text{RM}(S)$.
- SVD guarantees that $U_1^{\text{T}} U_1 = I$.

The same analysis applies for mode-2, mode-3, $\cdots$, mode-N. The algorithm for updating mode-i is summarized as Algorithm 1,

Algorithm 1 Function: Update-Mode(i)

$\mathcal{S} \leftarrow \mathcal{T}$
for $j \neq i$ **do**
  $\mathcal{S} \leftarrow \mathcal{S} \times_j \begin{bmatrix} U_j & \hat{\mathbf{1}}_{m_j} \end{bmatrix}^{\text{T}}$
end for
$[U_i, :, :] \leftarrow \text{SVD}(\text{RM}(\text{unfold}_i(\mathcal{S})))$

Updating $\{U_i\}$ iteratively, the objective function will keep decreasing and will eventually converge. Upon convergence, we can get $\mathcal{Q}$ by the following equation

$$\mathcal{Q} = \mathcal{T} \times_1 \begin{bmatrix} U_1 & \hat{\mathbf{1}}_{m_1} \end{bmatrix}^{\text{T}} \cdots \times_N \begin{bmatrix} U_N & \hat{\mathbf{1}}_{m_N} \end{bmatrix}^{\text{T}} \tag{24}$$

The complete work flow follows Algorithm 2. Since the objective function in Eq. (12) decreases after each iteration in Algorithm 1, it will finally converges to a local minimum.

Algorithm 2 Pseudo code describing the process of decomposing tensor $\mathcal{T}$ with the multi-affine model

Initialize $U_1, U_2, \cdots, U_N, \mathcal{Q}$
while not converge **do**
  for $i = 1 \cdots N$ **do**
    Update-Mode(i) (**Algorithm 1**)
  end for
end while
$\mathcal{Q} \leftarrow \mathcal{T} \times_1 \begin{bmatrix} U_1 & \hat{\mathbf{1}}_{m_1} \end{bmatrix}^{\text{T}} \cdots \times_N \begin{bmatrix} U_N & \hat{\mathbf{1}}_{m_N} \end{bmatrix}^{\text{T}}$

4.2. Decomposition with Missing Data

In the presence of missing data, we need to update the estimation of missing entries in each iteration as well. With all U_i fixed, the optimization problem we need to solve for the missing values becomes

$$\min_{\mathcal{Q}, \mathcal{T}_{\text{U}}} \left|\mathcal{T} - \mathcal{Q} \times_1 \begin{bmatrix} U_1 & \hat{\mathbf{1}}_{m_1} \end{bmatrix} \cdots \times_N \begin{bmatrix} U_N & \hat{\mathbf{1}}_{m_N} \end{bmatrix}\right|^2 \tag{25}$$

To solve for $\mathcal{T}$ and $\mathcal{Q}$ while fixing $\{U_i\}$, we put the elements in $\mathcal{T}$ and $\mathcal{Q}$ in column vectors $\mathbf{t}$ and $\mathbf{q}$, respectively. Since outer product operation $\times_i$ is a linear operation on $\mathcal{Q}$, $\mathcal{Q} \times_1 \begin{bmatrix} U_1 & \hat{\mathbf{1}}_{m_1} \end{bmatrix} \cdots \times_N \begin{bmatrix} U_N & \hat{\mathbf{1}}_{m_N} \end{bmatrix}$ can be viewed as a linear operation on $\mathbf{q}$, denoted by $A\mathbf{q}$. We can also rearrange the order of elements in $\mathbf{t}$ and A so that the entries corresponding to known elements come after missing elements in $\mathbf{t}$ and A. The optimization problem is then rewritten as Eq. (26)

$$\begin{aligned} &\arg\min_{\mathbf{q}, \mathbf{t}_{\text{U}}} \left| \begin{bmatrix} A_{\text{U}} \\ A_{\text{K}} \end{bmatrix} \mathbf{q} - \begin{bmatrix} \mathbf{t}_{\text{U}} \\ \mathbf{t}_{\text{K}} \end{bmatrix} \right|^2 \\ &= \arg\min_{\mathbf{q}, \mathbf{t}_{\text{U}}} \left|A_{\text{U}}\mathbf{q} - \mathbf{t}_{\text{U}}\right|^2 + \left|A_{\text{K}}\mathbf{q} - \mathbf{t}_{\text{K}}\right|^2 \end{aligned} \tag{26}$$

where subscript "U" stands for "unknown" and "K" stands for "known". By using least square, we can solve $\mathbf{q}$ as $\mathbf{q} = (A_{\text{K}}^{\text{T}} A_{\text{K}})^{-1} A_{\text{K}}^{\text{T}} \mathbf{t}_{\text{K}}$, and then it follows that $\mathbf{t}_{\text{U}} = A_{\text{U}} \cdot \mathbf{q}$.

The complete algorithm for decomposition with missing data is described in Algorithm 3.

Algorithm 3 Pseudo code describing the process of decomposing tensor $\mathcal{T}$ with missing data $\mathcal{T}_{\text{U}}$ with multi-affine model

Initialize $U_1, U_2, \cdots, U_N, \mathcal{Q}, \mathcal{T}_{\text{U}}$
while not converge **do**
  for $i = 1 \cdots N$ **do**
    Update-Mode(i) (**Algorithm 1**)
  end for
  $\mathcal{T}_{\text{U}} \leftarrow$ Minimizing Eq. (26) using least square
end while
$\mathcal{Q} \leftarrow \mathcal{T} \times_1 \begin{bmatrix} U_1 & \hat{\mathbf{1}}_{m_1} \end{bmatrix}^{\text{T}} \cdots \times_N \begin{bmatrix} U_N & \hat{\mathbf{1}}_{m_N} \end{bmatrix}^{\text{T}}$

5. Reconstructing New Mode Vectors

The multi-affine model we learned from a training data tensor can be used to recover missing data entries in a new mode vector sampled from the same latent function.

5.1. Optimization Problem

Suppose $\mathbf{v}$ is a mode-1 vector of the model we learned, namely

$$\mathbf{v} = \mathcal{Q} \times_1 \begin{bmatrix} U_1 & \mathbf{1}_{m_1} \end{bmatrix} \times_2 \begin{bmatrix} \mathbf{w}_2 \\ 1 \end{bmatrix}^{\mathrm{T}} \cdots \times_N \begin{bmatrix} \mathbf{w}_N \\ 1 \end{bmatrix}^{\mathrm{T}} \quad (27)$$

where $\mathbf{w}_i \in \mathbb{R}^{k_i}, i = 2, 3, \cdots, N$. Here we use $\mathbf{1}$ instead of $\hat{\mathbf{1}}_{m_i}$ to simplify notation since we can always perform matrix transformation on $\mathcal{Q}$ to find an equivalent solution that uses $\left\{ \begin{bmatrix} U_i & \mathbf{1} \end{bmatrix} \right\}$ instead of $\left\{ \begin{bmatrix} U_i & \hat{\mathbf{1}}_{m_i} \end{bmatrix} \right\}$.

Given a few observed data entries in $\mathbf{v}$, we seek to solve for the underlying parameters $\mathbf{w}_2, \mathbf{w}_3, \cdots$, that generate $\mathbf{v}$. Let $\mathbf{v}_{\mathrm{K}}$ be the vector of known entries of $\mathbf{v}$ (i.e. with all the unknown elements removed), and $U_{1\mathrm{K}}$ be the matrix of corresponding rows in U_1. The modified objective function is then

$$\min_{\{\mathbf{w}_i\}} \left| \mathbf{v}_{\mathrm{K}} - \mathcal{Q} \times_1 \begin{bmatrix} U_{1\mathrm{K}} & \mathbf{1} \end{bmatrix} \times_2 \begin{bmatrix} \mathbf{w}_2 \\ 1 \end{bmatrix}^{\mathrm{T}} \cdots \times_N \begin{bmatrix} \mathbf{w}_N \\ 1 \end{bmatrix}^{\mathrm{T}} \right|^2 \quad (28)$$

The optimization problem in Eq. (28) can be solved by alternately updating $\mathbf{w}_2, \mathbf{w}_3, \cdots, \mathbf{w}_N$. Updating a certain $\mathbf{w}_i$ with the other variables fixed is simply a least square problem. After solving the optimal parameters $\{\mathbf{w}_i\}$, the vector $\mathbf{v}$ can then be reconstructed by Eq. (27).

5.2. Regularization

If the number of known data entries in $\mathbf{v}$ is less than the sum of all $\dim(\mathbf{w}_i)$, the optimization problem in Eq. (28) is under-constrained. Furthermore, even the observed data entries may have noise; it is thus desirable to introduce a regularization that constrains $\{\mathbf{w}_i\}$ using prior knowledge.

Note that each mode-1 vector in a training data tensor has corresponding rows in U_i as its $\{\mathbf{w}_i\}$ parameter. Therefore, rows of U_i can be viewed as examples of $\mathbf{w}_i$ for new mode vectors. Due to the zero-mean and orthonormal constraints on $\begin{bmatrix} U_i & \hat{\mathbf{1}}_{m_i} \end{bmatrix}$, for each column k in U_i, we have

$$\hat{\mu} = \frac{1}{m_i} \sum_{j=1}^{m_i} U_i(j,k) = 0 \quad (29)$$

and

$$\hat{\sigma}^2 = \frac{1}{m_i} \sum_{j=1} (U_i(j,k) - 0)^2 = \frac{1}{m_i} \quad (30)$$

Therefore, we assume every entry of $\mathbf{w}_i$ follows $\mathcal{N}(0, 1/m_i)$, and add a regularization term for $\{\mathbf{w}_i(k)\}$ in the optimization problem Eq. (28) as below:

$$\min_{\{\mathbf{w}_i\}} \left| \mathbf{v}_{\mathrm{K}} - \mathcal{Q} \times_1 \begin{bmatrix} U_{1\mathrm{K}} & \mathbf{1} \end{bmatrix} \times_2 \begin{bmatrix} \mathbf{w}_2 \\ 1 \end{bmatrix}^{\mathrm{T}} \cdots \times_N \begin{bmatrix} \mathbf{w}_N \\ 1 \end{bmatrix}^{\mathrm{T}} \right|^2 + \lambda \sum_{i=2}^{N} \frac{1}{m_i} |\mathbf{w}_i|^2 \quad (31)$$

where λ is a constant that controls the weight of the regularization term. λ should be comparable to the variance of the noise of $\mathbf{v}_{\mathrm{K}}$.

Updating $\mathbf{w}_i$ with other parameter vectors fixed remains a least-square problem, so the alternating least square algorithm in Section 5.1 still works.

Note that this regularization can be used in multi-linear-model-based vector reconstruction in a similar way, which forms an optimization problem:

$$\min_{\{\mathbf{w}_i\}} \left| \mathbf{v}_{\mathrm{K}} - \mathcal{Q} \times_1 U_{1\mathrm{K}} \times_2 \mathbf{w}_2^{\mathrm{T}} \cdots \times_N \mathbf{w}_N^{\mathrm{T}} \right|^2 + \lambda \sum_{i=2}^{N} \frac{1}{m_i} |\mathbf{w}_i|^2 \quad (32)$$

However, the regularization term here is not reasonable and lacks statistical foundation. The regularization terms in Eq. (31) convey the belief that the vector $\mathbf{v}$ should not be very far from the mean of all the mode-1 vectors in $\mathcal{T}$. In the limiting case, as $\lambda \to \infty$, the solution of $\{\mathbf{w}_i\}$ will become 0, in which case the reconstructed $\mathbf{v}$ using Eq. (27) will become the mode-1 mean vector. In the multi-linear version this nice property does not exist. Letting $\lambda \to \infty$ leads to a 0-solution of $\{\mathbf{w}_i\}$, as well as a 0-solution of $\mathbf{v}$.

5.3. Extension on Reconstructing Mode Vectors

Similar to Eq. (27), an $m_1 \times m_2$ mode-1, 2 matrix $\mathcal{T}(:, :, i_3, \cdots, i_N)$ has a generative model as follows

$$\begin{aligned} \mathcal{T}(:, :, i_3, \cdots, i_N) = \mathcal{Q} &\times_1 \begin{bmatrix} U_1 & \mathbf{1}_{m_1} \end{bmatrix} \times_2 \begin{bmatrix} U_2 & \mathbf{1}_{m_2} \end{bmatrix} \\ &\times_3 \begin{bmatrix} U_3(i_3, :) & 1 \end{bmatrix} \times_4 \cdots \\ &\times_N \begin{bmatrix} U_N(i_N, :) & 1 \end{bmatrix} \end{aligned} \quad (33)$$

Reconstructing a new mode-1, 2 matrix M thus can be done by an optimization problem analogous to Eq. (31):

$$\min_{\{\mathbf{w}_i\}} \left| M_{\mathrm{K}} - \mathcal{Q}' \times_3 \begin{bmatrix} \mathbf{w}_3 \\ 1 \end{bmatrix}^{\mathrm{T}} \cdots \times_N \begin{bmatrix} \mathbf{w}_N \\ 1 \end{bmatrix}^{\mathrm{T}} \right|^2 + \lambda \sum_{i=3}^{N} \frac{1}{m_i} |\mathbf{w}_i|^2 \quad (34)$$

where $\mathcal{Q}' = \mathcal{Q} \times_1 \begin{bmatrix} U_{1\mathrm{K}} & \mathbf{1} \end{bmatrix} \times_2 \begin{bmatrix} U_{2\mathrm{K}} & \mathbf{1} \end{bmatrix}$, and the subscript "K" has the same meaning as in Eq. (28).

This extension can be generalized for reconstructing a higher order sub-tensor in a similar way, for example, reconstructing a mode-1, 2, 3 tensor is related to a generative

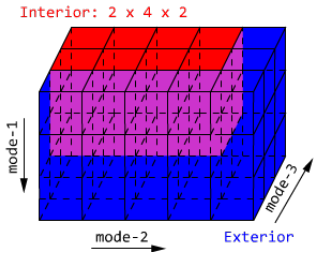


Figure 3: The figure demonstrates the exterior elements and the interior elements of a $3 \times 5 \times 3$ tensor. The exterior elements of a mode in a tensor is the set of elements whose subscripts of that mode are equal to the dimension of that mode. The interior sub-tensor excludes all exterior elements. **Best viewed electronically in color.**

model

$$\mathcal{R} = \mathcal{Q} \times_1 \begin{bmatrix} U_1 & \mathbf{1}_{m_1} \end{bmatrix} \times_2 \begin{bmatrix} U_2 & \mathbf{1}_{m_2} \end{bmatrix} \times_3 \begin{bmatrix} U_3 & \mathbf{1}_{m_3} \end{bmatrix} \times_4 \begin{bmatrix} \mathbf{w}_4 \\ 1 \end{bmatrix}^{\mathrm{T}} \cdots \times_N \begin{bmatrix} \mathbf{w}_N \\ 1 \end{bmatrix}^{\mathrm{T}} \tag{35}$$

6. Experiments

The experiments are conducted on two data sets to compare the performance of the multi-affine model and multi-linear model. The first one is synthetic data, and the second one is a set of landmarks describing the contours of faces extracted from the CMU Multi-PIE Face Database [2].

6.1. Synthetic Experiment

The synthetic experiment is designed to show that if our assumption is met, i.e. the translation offset cannot be ignored, the multi-affine model outperforms the multi-linear model in filling in the missing data in training data tensor and in reconstructing the new mode vectors.

Given the dimension parameters, we first generate a core tensor $\mathcal{Q}$ with every element as a random variable of $(0, 1)$ uniform distribution. Since only the exterior elements of $\mathcal{Q}$ (see Figure 3) contribute to the constant offsets, we amplify the exterior elements of each mode with a factor of 10 respectively. We then generate matrices $U_1, \cdots, U_N$ with every element drawn independently from $\mathcal{N}(0, 1)$. The data tensor is then constructed by

$$\mathcal{T} = \mathcal{Q} \times_1 \begin{bmatrix} U_1 & \mathbf{1} \end{bmatrix} \cdots \times_N \begin{bmatrix} U_N & \mathbf{1} \end{bmatrix} + \varepsilon \tag{36}$$

where ε is the Gaussian noise.

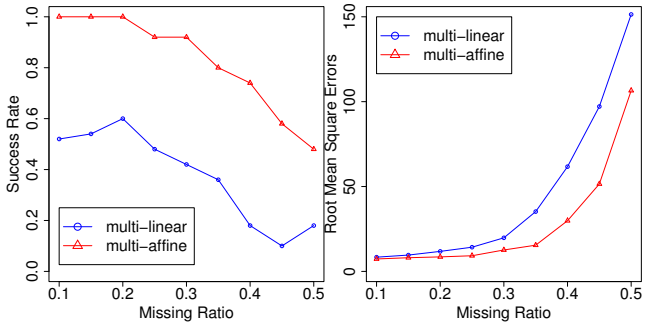
6.1.1 Data Tensor Recovery

We randomly remove a certain number of values from $\mathcal{T}$, and use both the multi-linear training algorithm and the multi-affine training algorithm to recover them.

Note that the alternating optimization algorithm doesn't guarantee a global minimum, so both methods may fail to fit to the ground truth model and thus produce a wrong data tensor. For each level of the missing data rate δ, we ran both methods on 50 randomly generated training data tensors. The results are shown in the Figure 4. The success rate indicates the frequency that the algorithm can recover the original data tensor. The residual error, measured on missing values, indicates the accuracy of missing value recovery on successful tests.

In the experiment, the data tensor $\mathcal{T}$ is of size $12 \times 10 \times 8 \times 6$, and the core tensor $\mathcal{Q}$ is of size $7 \times 6 \times 5 \times 3$. The noise on each element is a zero-mean Gaussian with 20.0 as variance.

For a multi-linear model, U_i has one more column which is not constrained to be $\mathbf{1}$. We believe the performance improvement of multi-affine model is due to the fact that it has fewer free variables.



(a) Success Rate (b) Estimation Error

Figure 4: Experiment results on recovering missing values in data tensor. 4a shows that multi-affine model succeeded more times than multi-linear model. 4b shows that multi-affine model training recovers missing values with smaller residual errors. **Best viewed electronically in color.**

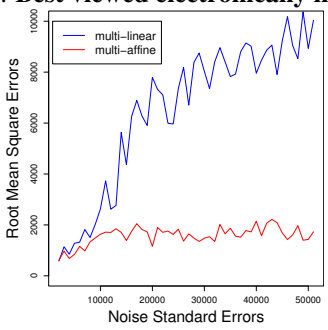
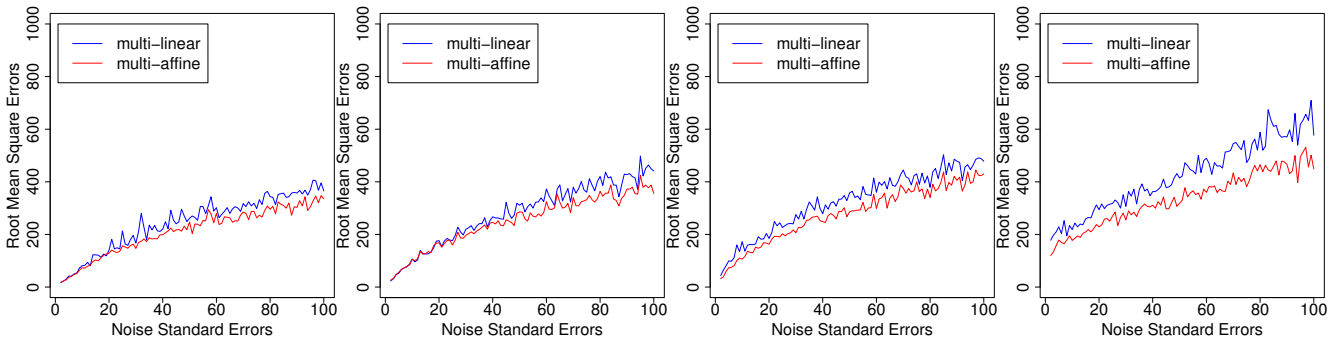


Figure 6: Same comparison as Figure 5 but with large range of noise level standard deviation. As the noise level increases dramatically, the residual error of both multi-affine and multi-linear models converges. The multi-affine model results in much smaller limit residual error. See Section 5.2 for explanation. **Best viewed electronically in color.**

6.1.2 Vector Reconstruction

In this experiment we randomly generate mode-1 vectors and make some of the entries missing. Then we use



(a) $\delta = 0.2$ (b) $\delta = 0.4$ (c) $\delta = 0.6$ (d) $\delta = 0.8$

Figure 5: Results of reconstructing missing data in new mode vectors using synthetic data. Residual errors are plotted against the standard deviation of noise with increasing missing data rate δ from left to right. As the noise level and missing data rate increase, the multi-affine model achieves noticeably smaller residual errors than multi-linear model.
Best viewed electronically in color.

both multi-linear and multi-affine models to reconstruct the mode-1 vectors.

The mode-1 vectors are generated as follows

$$\mathbf{v} = \mathcal{Q} \times_1 \begin{bmatrix} U_1 & \mathbf{1} \end{bmatrix} \times_1 \begin{bmatrix} \mathbf{w}_1 \\ 1 \end{bmatrix}^{\mathrm{T}} \cdots \times_N \begin{bmatrix} \mathbf{w}_N \\ 1 \end{bmatrix}^{\mathrm{T}} + \varepsilon \quad (37)$$

where elements of $\mathbf{w}_i$ are drawn from $\mathcal{N}(0, 1)$ (same as when we generate $\{U_i\}$ for the training data tensor). The noise term ε follows a zero-mean Gaussian distribution with a variance of σ^2, ranging from 0 to 100.

For each level of σ^2, and each level of missing data rate δ, we generated 100 mode-1 vectors and reconstructed them by solving the optimization problem Eq. (32) (multi-linear) and Eq. (31) (multi-affine), respectively. The weight of the regularization term λ is chosen to be σ^2. For each test we recorded the root mean square residual errors on missing values. The results are shown in Figure 5.

If the noise level keeps increasing and so does the regularization weight λ, the solution of $\mathbf{w}_i$ will converge to 0 in both methods. Therefore, the residual errors of both methods converge as shown in Figure 6. However in the multi-linear case, the converged solution corresponds to a 0 reconstruction vector while in the multi-affine case, the limit solution is the mode-1 mean vector, which is a much better estimation and achieves much lower RMS error.

6.2. Experiment on Face Contours

We extracted the contour landmarks from the faces of 48 people in Multi-PIE [2] as our training data. For each person we used 30 images with 5 poses and 6 expressions. The contour landmarks consisted of 68 2D vertices, so the data tensor $\mathcal{T}$ is an order-5 tensor of size $68 \times 2 \times 48 \times 6 \times 5$. The 5 modes express vertices, dimension (2D vertices), identities, poses and expressions, respectively.

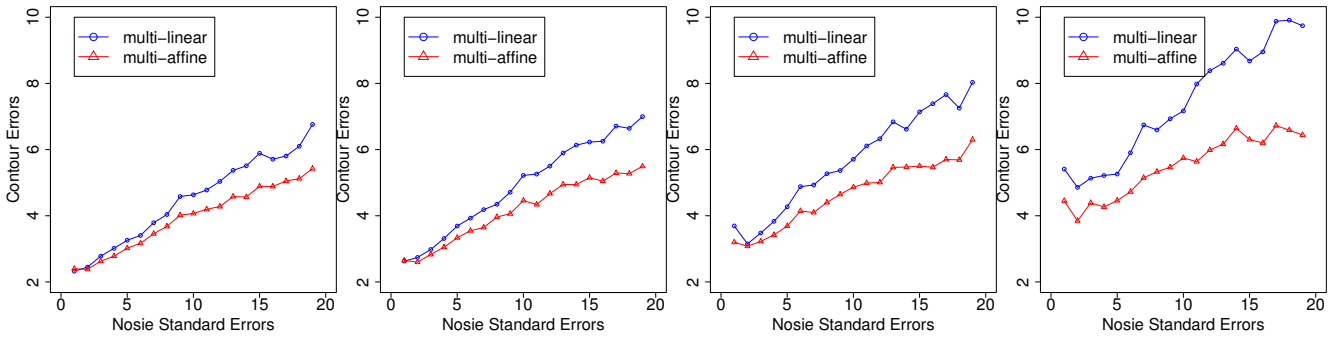
We first trained a multi-linear model and a multi-affine model out of this data tensor. Since the landmarks of some face photos are unknown, the models were trained with missing data. After the training, we used the model to recover missing landmarks of face images that were not in the training set.

Note that the landmarks of a single face image form a 68×2 matrix. Therefore, in our experiments, we reconstruct mode-1, 2 matrices (as discussed in Section 5.3), rather than new mode-1 vectors.

To evaluate our algorithm performance in the presence of noise, we also added a certain amount of random perturbation to the known landmark vertices. Figure 7 shows the performance comparison of both the multi-linear model and the multi-affine model on reconstructing the missing landmark locations, in terms of the root mean square errors. The errors are calculated by measuring the distances of estimated vertices from ground truth contours. Visual comparison of some results are shown in Figure 8.

7. Conclusion

In this paper, we have proposed a multi-affine model to handle arbitrary translation in different modes of tensor data. This translation invariance property is desirable but unavailable in conventional multi-linear models. We also have generalized HOSVD to compute the multi-affine model from an input tensor as well as to fill in its missing data. We have shown that the multi-affine model can be used to reconstruct new mode vectors with noisy and missing data entries. In particular, the regularization of the proposed multi-affine model is more sensible and effective than that of the multi-linear model for reconstructing new mode vectors. We have applied this multi-affine model on synthetic data analysis and on a face contour modeling problem. The comparison results show that the multi-affine model outperforms the multi-linear model in both filling in the missing values in model training and reconstruction of new mode vectors in model testing.



(a) $\delta = 48/68$ (b) $\delta = 53/68$ (c) $\delta = 58/68$ (d) $\delta = 63/68$

Figure 7: Results on reconstructing landmarks of face images that are not in the training set. Errors are plotted against the standard deviation of noise with increasing missing data rate δ from left to right. Errors are measured by the distances of estimated vertices from ground truth contours. As the noise level and missing data rate increase, the multi-affine model achieves noticeably smaller errors than the multi-linear model. **Best viewed electronically in color.**

(a) Ground Truth (b) Multi-linear (c) Multi-affine

Figure 8: Results on face contour reconstruction. From left to right, the three columns are the ground truth faces, the faces reconstructed by the multi-linear model and the faces reconstructed by the multi-affine model, respectively. In the figure, the red vertices are known, and the blue ones are unknown. Each input has only 5 known vertices, perturbed by a noise with standard deviation of 5 pixels. Note that the multi-affine reconstruction resembles the ground truth better, in terms of face size and shape.
Best viewed electronically in color.

8. Acknowledgement

This work is supported in part by NSF EFRI-0937847, NSF IIS-0845916, NSF IIS-0916441, a Sloan Research Fellowship, a Packard Fellowship for Science and Engineering, and a gift from Kodak Research Lab.

References

[1] V. M. Govindu. A tensor decomposition for geometric grouping and segmentation. *CVPR*, 2005. 2

[2] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker. Multi-pie. *IEEE International Conference on Automatic Face and Gesture Recognition*, 2008. 6, 7

[3] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 2008. 1

[4] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. On the best rank-1 and rank-(r1, r2, ... , rn) approximation of higher order tensors. *SIAM J. Matrix Analysis and Applications*, 2000. 1, 2, 3

[5] oshua B. Tenenbaum and W. T. Freeman. Separting style and content with bilinear models. *Neural Computation*, 2000. 2

[6] B. M. Smith, S. Zhu, and L. Zhang. Face image retrieval by shape manipulation. *CVPR*, 2011. 2

[7] M. A. O. Vasilescu and D. Terzopoulos. Tensortextures: Multilinear image-based rendering. *SIGGRAPH*, 2004. 2

[8] D. Vlasic, M. Brand, H. Pfister, and J. Popovic. Face transfer with multilinear models. *SIGGRAPHH*, 2005. 2

[9] M. A. O. Vsilescu and D. Terpoulos. Multilinear analysis of image ensembles: Tensorfaces. *ECCV*, 2002. 2

[10] H. Wang, Q. Wu, L. Shi, Y. Yu, and N. Ahuja. Out-of-core tensor approximation of multi-dimensional matrices of visual data. *ACM Trans. Graph.*, 2005. 2