Image Clustering with Tensor Representation*

Xiaofei He[1] Deng Cai[2] Haifeng Liu[3] Jiawei Han[2]

[1] Department of Computer Science, University of Chicago, Chicago, IL 60637
xiaofei@cs.uchicago.edu
[2] Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61081
{dengcai2,hanj}@cs.uiuc.edu
[3] Department of Computer Science, University of Toronto, Toronto, Canada
hfliu@cs.toronto.edu

ABSTRACT

We consider the problem of image representation and clustering. Traditionally, an $n_1 \times n_2$ image is represented by a vector in the Euclidean space $\mathbb{R}^{n_1 \times n_2}$. Some learning algorithms are then applied to these vectors in such a high dimensional space for dimensionality reduction, classification, and clustering. However, an image is intrinsically a matrix, or the second order tensor. The vector representation of the images ignores the spatial relationships between the pixels in an image. In this paper, we introduce a tensor framework for image analysis. We represent the images as points in the tensor space $\mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$ which is a tensor product of two vector spaces. Based on the tensor representation, we propose a novel image representation and clustering algorithm which explicitly considers the manifold structure of the tensor space. By preserving the local structure of the data manifold, we can obtain a tensor subspace which is optimal for data representation in the sense of local isometry. We call it **TensorImage** approach. Traditional clustering algorithm such as k-means is then applied in the tensor subspace. Our algorithm shares many of the data representation and clustering properties of other techniques such as Locality Preserving Projections, Laplacian Eigenmaps, and spectral clustering, yet our algorithm is much more computationally efficient. Experimental results show the efficiency and effectiveness of our algorithm.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Search and Retrieval—*clustering*; I.4.m [**image processing and computer vision**]: miscellaneous—*image clustering*

General Terms

algorithm, performance, theory, measurement, experimentation

Keywords

image clustering, image representation, dimensionality reduction, tensor, locality preserving projection, graph, manifold, subspace learning

* The work was supported in part by the U.S. National Science Foundation NSF IIS-02-09199/IIS-03-08215. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

MM'05, November 6–11, 2005, Singapore.

1. INTRODUCTION

Due to the rapid growth of the number of digital images, there is an increasing demand for effective image management tools. Image clustering and categorization is a means for high-level description of image content [5], [9], [20]. It maps the images in the database into different classes (clusters) such that the images in the same class share some common semantics. The generated classes provide a concise summarization and visualization of the image content that can be used for different tasks related to image management. Some typical applications include the implementation of efficient Content-Based Image Retrieval [2] and browsing [15] systems and the design of user-friendly interface [14].

In this paper, we consider the problem of image representation and clustering. Our framework of analysis is based on the algebra of tensors and the geometry of manifolds. Traditionally, an image (face image in particular) of size $n_1 \times n_2$ is considered as a point of the vector space $\mathbb{R}^{n_1 \times n_2}$. An image vector is formed by concatenating all the column vectors (or row vectors) of the image matrix together. The necessity of such "matrix-to-vector" conversion is due to the fact that most of the current learning algorithms can only be applied to the vectorial data. Different from traditional algorithms for image representation and clustering, we consider an image as a matrix, or the second order tensor. An $n_1 \times n_2$ image is naturally represented as a second order tensor in the tensor space $\mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$ which is a tensor product of two vector spaces.

Generally, the representation space of the images is of high dimensionality. For example, a typical face image is of size 32×32, resulting in a 1024-dimensional vector, or a 32×32-dimensional matrix. Following the intuition that naturally occurring image data may be generated by structured systems with possibly much fewer degrees of freedom than the ambient dimension would suggest, various researchers have considered the case when the data lives on or close to a submanifold of the ambient space. One hopes then to

estimate geometrical and topological properties of the submanifold from random points lying on this unknown submanifold. The typical manifold learning algorithms include Principal Component Analysis (PCA, [4]), Linear Discriminant Analysis (LDA, [4]), Locality Preserving Projections (LPP, [7]), ISOMAP [18], Locally Linear Embedding (LLE, [16]) and Laplacian Eigenmaps [1]. However, all of these algorithms can only be applied to vectorial data rather than data in matrix form. In this paper, we develop a novel algorithm for learning a lower dimensional representation of the image manifold embedded in the ambient tensor space. We call this algorithm **TensorImage**. Our algorithm explicitly takes into account the manifold structure which is modeled by a nearest neighbor graph. The basis functions of the *optimal* tensor subspace in the sense of local isometry is obtained by approximating the eigenfunctions of the Laplace Beltrami operator on the manifold.

Once we obtain a compact representation of the image, clustering can be then performed in the lower dimensional tensor subspace. Some typical clustering algorithms include k-means, Gaussian Mixture Model, and spectral clustering [13], [17]. Spectral clustering have attracted considerable attention in recent years. It can be thought of as a combination of spectral dimensionality reduction [1] and traditional clustering algorithms like k-means. In this work, we apply k-means in the tensor subspace for clustering. The TensorImage approach may be interpreted as a search for good spectral partitioning functions by restricting our search to *linear* functions (i.e. projections) alone. The objective function of TensorImage is actually a min-cut like graph partitioning criteria. In this sense, our method can be thought of as a linear spectral clustering. The relationship between spectral clustering and dimensionality reduction can be found in [1].

There are a few aspects of the developments in this paper that are worth emphasizing:

1. Different from most of the existing algorithms for image management that treat images as vectors, our TensorImage approach treats images as matrices. TensorImage provides a more natural representation for images.

2. By using the algebra of tensors, our algorithm is much more computationally efficient than traditional dimensionality reduction algorithms such as PCA, LDA, and LPP. Note that, in this paper our primary interest is focused on images which can be considered as the second order tensors. However, our algorithm can be easily extended to higher order tensors. For example, videos can be naturally considered as the third order tensors in that the time is the third dimension.

3. Our results may also be of interest to researchers in computer graphics who have considered the question of modeling the Bidirectional Texture Function (BTF) whose observational data is of six dimensions (i.e. sixth order tensor), two variables for surface location, two variables for view direction and two variables for illumination direction [10]. Researchers in computer vision, pattern recognition, molecular biology, information retrieval, and other areas where large amount of higher order tensor (rather than vector) based data are available may find some use of the algorithm and analysis of this paper.

The rest of the paper is organized as follows: Section 2 provides a brief description of the algebra of tensors. The *TensorImage* approach for image analysis (representation and clustering) is described in Section 3. In Section 4, we give a theoretical justification of TensorImage and its connections to LPP and Laplacian Eigenmap. The experimental results on image database are presented in Section 5. Finally, we provide some concluding remarks and suggestions for future work in Section 6.

2. PRELIMINARIES

In this section, we provide a brief overview of the algebra of tensors. For a detailed treatment please see [8].

2.1 The Algebra of Tensors

A tensor with order k is a real-valued multilinear function on k vector spaces:

$$T : \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_k} \to \mathbb{R}$$

The number k is called the *order* of T. A multilinear function is linear as a function of each variable separately. The set of all k-tensors on $\mathbb{R}^{n_i}, i = 1, \cdots, k$, denoted by $\mathcal{T}^k$, is a vector space under the usual operations of pointwise addition and scalar multiplication:

$$(aT)(\mathbf{a}_1, \cdots, \mathbf{a}_k) = a\left(T(\mathbf{a}_1, \cdots, \mathbf{a}_k)\right),$$

$$(T + T')(\mathbf{a}_1, \cdots, \mathbf{a}_k) = T(\mathbf{a}_1, \cdots, \mathbf{a}_k) + T'(\mathbf{a}_1, \cdots, \mathbf{a}_k)$$

where $\mathbf{a}_i \in \mathbb{R}^{n_i}$. Given two tensors $S \in \mathcal{T}^k$ and $T \in \mathcal{T}^l$, define a map:

$$S \otimes T : \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_{k+l}} \to \mathbb{R}$$

by

$$S \otimes T(\mathbf{a}_1, \cdots, \mathbf{a}_{k+l}) = S(\mathbf{a}_1, \cdots, \mathbf{a}_k) T(\mathbf{a}_{k+1}, \cdots, \mathbf{a}_{k+l})$$

It is immediate from the multilinearity of S and T that $S \otimes T$ depends linearly on each argument $\mathbf{a}_i$ separately, so it is a $(k+l)$-*tensor*, called the tensor product of S and T.

For the first order tensors, they are simply the covectors on $\mathbb{R}^{n_1}$. That is, $\mathcal{T}^1 = \mathcal{R}^{n_1}$, where $\mathcal{R}^{n_1}$ is the dual space of $\mathbb{R}^{n_1}$. The second order tensor space is a product of two first order tensor spaces, i.e. $\mathcal{T}^2 = \mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$. Let $\mathbf{e}_1, \cdots, \mathbf{e}_{n_1}$ be the standard basis of $\mathbb{R}^{n_1}$, and $\varepsilon_1, \cdots, \varepsilon_{n_1}$ be the dual basis of $\mathcal{R}^{n_1}$ which is formed by coordinate functions with respect to the basis of $\mathbb{R}^{n_1}$. Likewise, let $\widetilde{\mathbf{e}}_1, \cdots, \widetilde{\mathbf{e}}_{n_2}$ be a basis of $\mathbb{R}^{n_2}$, and $\widetilde{\varepsilon}_1, \cdots, \widetilde{\varepsilon}_{n_2}$ be the dual basis of $\mathcal{R}^{n_2}$. We have,

$$\varepsilon_i(\mathbf{e}_j) = \delta_{ij} \text{ and } \widetilde{\varepsilon}_i(\widetilde{\mathbf{e}}_j) = \delta_{ij}$$

where δ_{ij} is the kronecker delta function. Thus, $\{\varepsilon_i \otimes \widetilde{\varepsilon}_j\}$ $(1 \le i \le n_1,\ 1 \le j \le n_2)$ forms a basis of $\mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$. For any 2-tensor T, we can write it as:

$$T = \sum_{\substack{1 \le i \le n_1 \\ 1 \le j \le n_2}} T_{ij} \varepsilon_i \otimes \widetilde{\varepsilon}_j$$

This shows that every 2-tensor in $\mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$ uniquely corresponds to a $n_1 \times n_2$ matrix. Given two vectors $\mathbf{a} =$

$\sum_{k=1}^{n_1} a_k \mathbf{e}_k \in \mathbb{R}^{n_1}$ and $\mathbf{b} = \sum_{l=1}^{n_2} b_l \widetilde{\mathbf{e}}_l \in \mathbb{R}^{n_2}$, we have

$$\begin{aligned} T(\mathbf{a}, \mathbf{b}) &= \sum_{ij} T_{ij} \varepsilon_i \otimes \widetilde{\varepsilon}_j (\sum_{k=1}^{n_1} a_k \mathbf{e}_k, \sum_{l=1}^{n_2} b_l \widetilde{\mathbf{e}}_l) \\ &= \sum_{ij} T_{ij} \varepsilon_i (\sum_{k=1}^{n_1} a_k \mathbf{e}_k) \widetilde{\varepsilon}_j (\sum_{l=1}^{n_2} b_l \widetilde{\mathbf{e}}_l) \\ &= \sum_{ij} T_{ij} a_i b_j \\ &= \mathbf{a}^T T \mathbf{b} \end{aligned}$$

Note that, in this paper our primary interest is focused on the second order tensors. However, the algebra presented here and the algorithm presented in the next section can also be applied to higher order tensors.

3. TENSORIMAGE: IMAGE ANALYSIS WITH TENSOR REPRESENTATION

In this section, we introduce a tensor framework for image analysis (representation and clustering). The following section is based on the tensor algebra and the standard spectral graph theory [3]. We begin with a brief description of the coordinate transformation in the tensor space.

3.1 Coordinate System Transformation

Let $\{\mathbf{u}_k\}_{k=1}^{n_1}$ be the new orthonormal basis of $\mathcal{R}^{n_1}$ and $\{\mathbf{v}_l\}_{l=1}^{n_2}$ be the new orthonormal basis of $\mathcal{R}^{n_2}$. We have

$$\mathbf{u}_k = (u_{k1}, \cdots, u_{kn_1})^T = \sum_i u_{ki} \varepsilon_i$$

and

$$\mathbf{v}_l = (v_{l1}, \cdots, v_{ln_2})^T = \sum_j v_{lj} \widetilde{\varepsilon}_j$$

It is easy to show that:

$$\varepsilon_i = \sum_{k=1}^{n_1} u_{ki} \mathbf{u}_k$$

$$\widetilde{\varepsilon}_j = \sum_{l=1}^{n_2} v_{lj} \mathbf{v}_l$$

Thus, an image tensor can be computed using the new basis as follows:

$$\begin{aligned} T &= \sum_{ij} T_{ij} \varepsilon_i \otimes \widetilde{\varepsilon}_j \\ &= \sum_{ij} T_{ij} \left(\sum_{k=1}^{n_1} u_{ki} \mathbf{u}_k \right) \otimes \left(\sum_{l=1}^{n_2} v_{lj} \mathbf{v}_l \right) \\ &= \sum_{kl} \left(\sum_{ij} T_{ij} u_{ki} v_{lj} \right) \mathbf{u}_k \otimes \mathbf{v}_l \\ &= \sum_{kl} \left(\mathbf{u}_k^T T \mathbf{v}_l \right) \mathbf{u}_k \otimes \mathbf{v}_l \end{aligned} \quad (1)$$

Eqn. 1 shows that $\{\mathbf{u}_i \otimes \mathbf{v}_j\}$ forms a new basis of the tensor space $\mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2}$. Thus, the tensor T becomes $U^T T V$ with respect to the new basis, where $U = (\mathbf{u}_1, \cdots, \mathbf{u}_{n_1})$ and $V = (\mathbf{v}_1, \cdots, \mathbf{v}_{n_2})$. Specifically, the projection of T on the basis $\mathbf{u}_i \otimes \mathbf{v}_j$ can be computed as their inner product:

$$< T, \mathbf{u}_i \otimes \mathbf{v}_j > = < T, \mathbf{u}_i \mathbf{v}_j^T > = \mathbf{u}_i^T T \mathbf{v}_j$$

In this paper, we consider an image as a 2-tensor. For the basis tensors $\mathbf{u}_i \otimes \mathbf{v}_j$, or $\mathbf{u}_i \mathbf{v}_j^T$ in matrix form, they can also be considered as images. We call them *TensorImages*. Therefore, for any image, it can be represented as a linear combination of the *TensorImages*.

3.2 The Objective Function for Data Clustering and Dimensionality Reduction

Recently, there has been considerable interest in spectrally based techniques to data clustering [13], [17], [20] and dimensionality reduction [1], [7] in high dimensional spaces. Spectral clustering has very close tie to spectral dimensionality reduction. In fact, spectral clustering can be thought of as a combination of spectral dimensionality reduction and traditional clustering algorithms such as k-means. In this subsection, we describe a novel objective function for data reduction and clustering with tensor representation.

As we described previously, the images are probably sampled from a low dimensional submanifold embedded in the ambient space. One hopes then to estimate geometrical and topological properties of the submanifold from random points ("scattered data") lying on this unknown submanifold. Particularly, We consider the problem of finding a linear subspace approximation to the submanifold in the sense of local isometry.

Let X denote the second order image tensor, or matrix. Given m data points $\mathcal{X} = \{X_1, \cdots, X_m\}$ sampled from the face submanifold $\mathcal{M} \in \mathcal{R}^{n_1} \otimes \mathcal{R}^{n_1}$, one can build a nearest neighbor graph $\mathcal{G}$ to model the local geometrical structure of $\mathcal{M}$. Let S be the weight matrix of $\mathcal{G}$. A possible definition of S is as follows:

$$S_{ij} = \begin{cases} 1, & \text{if } X_i \text{ is among the } p \text{ nearest neighbors of } X_j, \text{ or } X_j \text{ is among the } p \text{ nearest neighbors of } X_i; \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

There are also other definitions of S, such as using heat kernel. Please see [1] for details.

Let U and V be the transformation matrices. A reasonable transformation respecting the graph structure can be obtained by solving the following objective functions:

$$\min_{U,V} \sum_{ij} \|U^T X_i V - U^T X_j V\|^2 S_{ij} \quad (3)$$

The objective function incurs a heavy penalty if neighboring points X_i and X_j are mapped far apart. Therefore, minimizing it is an attempt to ensure that if X_i and X_j are "close" then $U^T X_i V$ and $U^T X_j V$ are "close" as well.

Our objective function in (3) can also be interpreted from the perspective of *min-cut* graph partitioning. Without loss of generality, we consider the problem of dividing the graph $\mathcal{G}$ into two disjoint subgraphs, $\mathcal{G}_1$ and $\mathcal{G}_2$, such that $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$ and $\mathcal{G}_1 \cap \mathcal{G}_2 = \emptyset$. The min-cut criteria can be stated as follows:

$$\min_{\mathcal{G}_1, \mathcal{G}_2} \sum_{i \in \mathcal{G}_1} \sum_{j \in \mathcal{G}_2} S_{ij} \quad (4)$$

Now, for each node (data point) X_i, we assign it a label y_i to indicate whether it is in $\mathcal{G}_1$ or $\mathcal{G}_2$, $y_i \in \{-1, 1\}$. If X_i and X_j belong to the same subgraph, $y_i - y_j = 0$; otherwise, $|y_i - y_j| = 2$. Thus, the min-cut graph partitioning problem can be reduced to finding the optimal vector $\mathbf{y}(= (y_1, \cdots, y_m)^T)$

which minimizes the following objective function:

$$\min_{\mathbf{y}, y_i \in \{-1,1\}} \sum_{ij} (y_i - y_j)^2 S_{ij} \tag{5}$$

Since y_i can only be 1 or -1, finding the optimal $\mathbf{y}$ is time consuming. In order to reduce the computational complexity, one may resort to spectral relaxation method to approximate the optimal solution. Specifically, we can relax the constraint that $y_i \in \{-1, 1\}$, by allowing y_i to be any real number. Further, if we restrict the map from X_i to y_i to be linear, i.e. $y_i = \mathbf{u}^T X_i \mathbf{v}$, we get the objective function (3). Please see [12], [6] for the details about spectral graph partitioning.

3.3 Derivation

Let $Y_i = U^T X_i V$. Let D be a diagonal matrix, $D_{ii} = \sum_j S_{ij}$. Since $\|A\|^2 = tr(AA^T)$, we see that:

$$\begin{aligned}
& \frac{1}{2} \sum_{ij} \|U^T X_i V - U^T X_j V\|^2 S_{ij} \\
= \; & \frac{1}{2} \sum_{ij} tr\left((Y_i - Y_j)(Y_i - Y_j)^T \right) S_{ij} \\
= \; & \frac{1}{2} \sum_{ij} tr\left(Y_i Y_i^T + Y_j Y_j^T - Y_i Y_j^T - Y_j Y_i^T \right) S_{ij} \\
= \; & tr\left(\sum_i D_{ii} Y_i Y_i^T - \sum_{ij} S_{ij} Y_i Y_j^T \right) \\
= \; & tr\left(\sum_i D_{ii} U^T X_i V V^T X_i^T U - \sum_{ij} S_{ij} U^T X_i V V^T X_j^T U \right) \\
= \; & tr\left(U^T \left(\sum_i D_{ii} X_i V V^T X_i^T - \sum_{ij} S_{ij} X_i V V^T X_j^T \right) U \right) \\
\doteq \; & tr\left(U^T (D_V - S_V) U \right)
\end{aligned}$$

where $D_V = \sum_i D_{ii} X_i V V^T X_i^T$ and $S_V = \sum_{ij} S_{ij} X_i V V^T X_j^T$. "$tr$" denotes the *trace* operator. Similarly, $\|A\|^2 = trace(A^T A)$, so we also have

$$\begin{aligned}
& \frac{1}{2} \sum_{ij} \|U^T X_i V - U^T X_j V\|^2 S_{ij} \\
= \; & \frac{1}{2} \sum_{ij} tr\left((Y_i - Y_j)^T (Y_i - Y_j) \right) S_{ij} \\
= \; & \frac{1}{2} \sum_{ij} tr\left(Y_i^T Y_i + Y_j^T Y_j - Y_i^T Y_j - Y_j^T Y_i \right) S_{ij} \\
= \; & tr\left(\sum_i D_{ii} Y_i^T Y_i - \sum_{ij} S_{ij} Y_i^T Y_j \right) \\
= \; & tr\left(V^T \left(\sum_i D_{ii} X_i^T U U^T X_i - \sum_{ij} S_{ij} X_i^T U U^T X_j \right) V \right) \\
\doteq \; & tr\left(V^T (D_U - S_U) V \right)
\end{aligned}$$

where $D_U = \sum_i D_{ii} X_i^T U U^T X_i$ and $S_U = \sum_{ij} S_{ij} X_i^T U U^T X_j$. Therefore, we should simultaneously minimize $tr\left(U^T (D_V - S_V) U \right)$ and $tr\left(V^T (D_U - S_U) V \right)$.

In addition to preserving the graph structure, we also aim at maximizing the global variance on the manifold. Recall that the variance of a random variable x can be written as follows:

$$var(x) = \int_{\mathcal{M}} (x - \mu)^2 dP(x)$$

$$\mu = \int_{\mathcal{M}} x dP(x)$$

where $\mathcal{M}$ is the data manifold, μ is the expected value of x and dP is the probability measure on the manifold. By spectral graph theory [3], dP can be discretely estimated by the diagonal matrix $D(D_{ii} = \sum_j S_{ij})$ on the sample points. Let $Y = U^T X V$ denote the random variable in the tensor subspace and suppose the data points have a zero mean. Thus, the *weighted* variance can be estimated as follows:

$$\begin{aligned}
var(Y) &= \sum_i \|Y_i\|^2 D_{ii} \\
&= \sum_i tr(Y_i^T Y_i) D_{ii} \\
&= \sum_i tr\left(V^T X_i^T U U^T X_i V \right) D_{ii} \\
&= tr\left(V^T \left(\sum_i D_{ii} X_i^T U U^T X_i \right) V \right) \\
&= tr\left(V^T D_U V \right)
\end{aligned}$$

Similarly, $\|Y_i\|^2 = tr(Y_i Y_i^T)$, so we also have:

$$\begin{aligned}
var(Y) &= \sum_i tr(Y_i Y_i^T) D_{ii} \\
&= tr\left(U^T \left(\sum_i D_{ii} X_i V V^T X_i^T \right) U \right) \\
&= tr\left(U^T D_V U \right)
\end{aligned}$$

Finally, we get the following optimization problems:

$$\min_{U,V} \frac{tr\left(U^T (D_V - S_V) U \right)}{tr\left(U^T D_V U \right)} \tag{6}$$

$$\min_{U,V} \frac{tr\left(V^T (D_U - S_U) V \right)}{tr\left(V^T D_U V \right)} \tag{7}$$

The above two minimization problems (6) and (7) depends on each other, and hence can not be solved independently. In the following, we describe a simple yet effective computational method to solve these two optimization problems.

It is easy to see that the optimal U should be the generalized eigenvectors of $(D_V - S_V, D_V)$ and the optimal V should be the generalized eigenvectors of $(D_U - S_U, D_U)$. However, it is difficult to compute the optimal U and V simultaneously since the matrices D_V, S_V, D_U, S_U are not fixed. In this paper, we compute U and V iteratively as follows. We first fix U, then V can be computed by solving the following generalized eigenvector problem:

$$(D_U - S_U)\mathbf{v} = \lambda D_U \mathbf{v} \tag{8}$$

Once V is obtained, U can be updated by solving the following generalized eigenvector problem:

$$(D_V - S_V)\mathbf{u} = \lambda D_V \mathbf{u} \tag{9}$$

Thus, the optimal U and V can be obtained by iteratively computing the generalized eigenvectors of (8) and (9). In our experiments, U is initially set to be the identity matrix.

3.4 Image Clustering in the Tensor Subspace

Once we project the images into a tensor subspace, clustering can be performed in such a lower dimensional space. We adopt k-means clustering algorithm for its simplicity in this paper. Note that, for the images of size $n_1 \times n_2$, our TensorImage algorithm only needs to compute the eigen-decomposition of matrices of size $n_1 \times n_1$ or $n_2 \times n_2$. For traditional vector based subspace learning algorithms such PCA, LDA, and LPP [7], they need to compute the eigen-decomposition of matrices of size $n \times n$, where $n = n_1 \times n_2$. Clearly, our algorithm is much more computationally efficient and can deal with images with large size.

4. THEORETICAL ANALYSIS

In this section, we provide a theoretical analysis of our algorithm. We show that TensorImages are actually linear approximations to the eigenfunctions of the Laplace Beltrami operator on the manifold.

4.1 Optimal Tensor Embedding

Given a face manifold $\mathcal{M}$, let $f : \mathcal{M} \subset \mathcal{R}^{n_1} \otimes \mathcal{R}^{n_2} \rightarrow \mathcal{R}^{l_1} \otimes \mathcal{R}^{l_2}$ be a smooth map. Belkin and Niyogi [1] showed that the optimal map preserving locality can be obtained by solving the following optimization problem on the manifold:

$$\min_{\|f\|_{\mathcal{L}^2(\mathcal{M})}=1} \int_{\mathcal{M}} \|\nabla f\|^2 \tag{10}$$

By Stokes' theorem, we have:

$$\int_{\mathcal{M}} \|\nabla f\|^2 = \int_{\mathcal{M}} < \mathcal{L}f, f >$$

where $\mathcal{L}$ is the Laplace Beltrami operator, i.e. $\mathcal{L}f = -\text{div}\nabla(f)$. Therefore, the optimal f to the objective function (10) has to be an eigenfunction of $\mathcal{L}$. In general the manifold $\mathcal{M}$ is unknown, thus there is no way to solve the optimization problem. However, given some data points $\mathcal{X} = \{X_1, \cdots, X_m\}$ sampled from $\mathcal{M}$, one may find a discrete approximation to $\mathcal{L}$. By spectral graph theory [3], the Laplacian matrix L for finite graph is analogous to $\mathcal{L}$ on compact Riemannian manifold. Therefore, the integral can be discretely approximated as follows:

$$\int_{\mathcal{M}} < (\mathcal{L}f)(x), f(x) > \approx \sum_{ij} L_{ij} < f(X_i), f(X_j) > \tag{11}$$

$$\|f\|^2 = \int_{\mathcal{M}} < f(x), f(x) > \approx \sum_{i} < f(X_i), f(X_i) > D_{ii} \tag{12}$$

where L is the Laplacian matrix, $L = D - S$ (D and S are defined in Section 3). D_{ii} models the local density of X_i. If we assume f to be a linear map, i.e. $f(X) = U^T X V$, we have:

$$\begin{aligned}
&\int_{\mathcal{M}} < (\mathcal{L}f)(x), f(x) > \\
\approx\ & \sum_{ij} L_{ij} < U^T X_i V, U^T X_j V > \\
=\ & \sum_{ij} L_{ij} \cdot tr(U^T X_i V V^T X_j^T U) \\
=\ & tr\left(U^T \left(\sum_{ij} L_{ij} X_i V V^T X_j^T\right) U\right) \\
=\ & tr\left(V^T \left(\sum_{ij} L_{ij} X_i^T U U^T X_j\right) V\right)
\end{aligned}$$

and,

$$\begin{aligned}
\|f\|^2 =\ & \int_{\mathcal{M}} < f(x), f(x) > dx \\
\approx\ & \sum_{i} D_{ii} < U^T X_i V, U^T X_i V > \\
=\ & \sum_{i} D_{ii} \cdot tr\left(U^T X_i V V^T X_i^T U\right) \\
=\ & tr\left(U^T \left(\sum_{i} D_{ii} X_i V V^T X_i^T\right) U\right) \\
=\ & tr\left(V^T \left(\sum_{i} D_{ii} X_i^T U U^T X_i\right) V\right)
\end{aligned}$$

This ultimately leads to the optimization problems as in Eqn. (6) and (7). The derivation reflects the intrinsic geometrical structure of the manifold.

4.2 Connections between LPP, Laplacian Eigenmaps and TensorImage

In this section, we provide a theoretical analysis about the connections between several spectral techniques for data representation and clustering. Specifically, we consider the following three algorithms: Locality Preserving Projection (LPP, [7]), Laplacian Eigenmap (LE, [1]), and TensorImage. All of these algorithms are based on a graph model induced from the data points. We assume that they share the same graph model in the following.

Recall that Laplacian Eigenmap tries to find an optimal *nonlinear* embedding of the data points which preserves the graph structure. Given m data points $\{\mathbf{x}_1, \cdots, \mathbf{x}_m\} \in \mathbb{R}^n$. $\mathbf{x}_i$ is the *vector* representation of the i-th image. Let y_i denote the one-dimensional representation of $\mathbf{x}_i$ and $\mathbf{y} = (y_1, \cdots, y_m)$. The embedding of Laplacian Eigenmap can be obtained by solving the following generalized eigenvector problem:

$$L\mathbf{y} = \lambda D\mathbf{y} \tag{13}$$

where L is the Laplacian matrix of the graph. Please see [1] for details. For LPP, the embedding is linear, i.e. $y_i = \mathbf{w}^T \mathbf{x}_i$. Thus, $\mathbf{y} = \mathbf{X}^T \mathbf{w}$, where $\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_m) \in \mathbb{R}^n$. LPP can be obtained by solving the following eigen-problem:

$$\mathbf{X}L\mathbf{X}^T\mathbf{w} = \lambda \mathbf{X}D\mathbf{X}^T\mathbf{w} \tag{14}$$

The rank of $\mathbf{X}$ is no greater than $\min(m, n)$. Thus, if $n > m$, we can reduce the image space into an m dimensional subspace without losing any information by using Singular

Value Decomposition (SVD). Correspondingly, the data matrix $\mathbf{X}$ in such a subspace becomes a square matrix. We have the following proposition:

PROPOSITION 4.1. *If $\boldsymbol{X}$ is a full rank square matrix, then LPP and Laplacian Eigenmap have the same result.*

PROOF. Since $\mathbf{y} = \mathbf{X}^T\mathbf{w}$, Eqn. (14) can be rewritten as follows:

$$\mathbf{X}L\mathbf{y} = \lambda\mathbf{X}D\mathbf{y}$$

Since $\mathbf{X}$ is a full rank square matrix, we get the following equation:

$$L\mathbf{y} = \lambda D\mathbf{y}$$

which is just the eigenvalue problem of Laplacian Eigenmaps. □

From the above proposition we see that, if the dimensionality of the image space is larger than the number of sample images and the sample images are linearly independent, LPP and Laplacian Eigenmap will have the same embedding result.

While Laplacian Eigenmap is *nonlinear* and LPP is *linear*, our TensorImage algorithm is *multilinear*, i.e. $y_i = \mathbf{u}^T X_i \mathbf{v}$ where X_i is the *matrix* representation of the i-th image. In fact, TensorImage can be thought of as a special case of LPP with the following constraint:

$$w_{n_2(i-1)+j} = u_i v_j \tag{15}$$

For $n_1 \times n_2$ images, the projection vector $\mathbf{w}$ is $n_1 \times n_2$-dimensional, so there are $n_1 \times n_2$ parameters for LPP. For TensorImage, there are only $n_1 + n_2$ parameters. Therefore, TensorImage is much more computationally tractable and especially suitable for small sample issues.

5. EXPERIMENTAL RESULTS

In this section, we evaluate our algorithm on a standard image databases. We begin with a description of the data preparation.

5.1 Data Preparation

The database used in our experiment is the PIE (Pose, Illumination, and Experience) database from CMU. It contains 68 subjects with 41,368 face images as a whole. The face images were captured under varying pose, illumination and expression. We fixed the pose and expression. Thus, for each subject, we got 22 images under different lighting conditions. Figure 2 shows some sample images for a certain subject. Preprocessing to locate the faces was applied. Original images were normalized (in scale and orientation) such that the two eyes were aligned at the same position. Then, the facial areas were cropped into the final images for matching. The size of each cropped image in all the experiments is 32×32 pixels, with 256 gray levels per pixel. No further preprocessing is done. For traditional learning algorithms (PCA and LPP), the image is represented as a 1024-dimensional vector. For TensorImage, the image is represented as a (32×32)-dimensional matrix, or the second order tensor.

5.2 2-D Visualization of Image Set

As we described previously, PCA, LPP and TensorImage are different dimensionality reduction algorithms. In this subsection, we use them to project the images into a 2-dimensional subspace for visualization. We randomly selected 5 classes for this test. Figure 1 shows the 2D embedding results. In this example, we did not show the embedding result of Laplacian Eigenmaps [1] because it gives the same result as LPP, please see Proposition 4.1 for details. For TensorImage method, we can project the images into either $\mathcal{R}^1 \otimes \mathcal{R}^2$ or $\mathcal{R}^2 \otimes \mathcal{R}^1$, both of which are 2-dimensional spaces. $\mathcal{R}^1 \otimes \mathcal{R}^2$ is formed by the projection $\mathbf{u}_1^T X[\mathbf{v}_1, \mathbf{v}_2]$ and $\mathcal{R}^2 \otimes \mathcal{R}^1$ is formed by the projection $[\mathbf{u}_1, \mathbf{u}_2]^T X \mathbf{v}_1$. As can be seen, PCA performs the worst. It fails to distinguish the different classes, and the five classes are mixed together. TensorImage performed marginally better than LPP. For LPP, we can see that there are two classes mixed together. Clearly, these two classes will be grouped together when clustering is performed. This illustrative example shows that TensorImage can have more discriminating power than PCA and LPP.

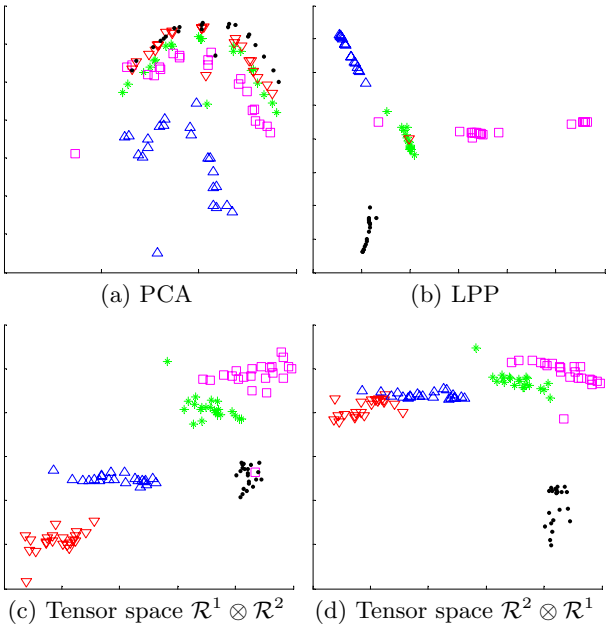

Figure 1: 2D visualization of image set

5.3 Evaluation Metrics of Clustering

We compared the following five algorithms for image clustering:

- baseline: k-means in the original space (K-means)
- our algorithm: TensorImage+k-means (TensorImage)
- PCA+k-means (PCA)
- LPP+k-means (LPP, [20])
- Normalized Cut (NCut, [17])

Note that, TensorImage, PCA, and LPP are all linear algorithms. Normalized cut (NCut, [17]) is nonlinear. NCut can be thought of as a combination of Laplacian Eigenmaps and k-means. We tested these algorithms on several cases. For each case, $k(= 5, 10, 30, 68)$ classes were randomly selected

Figure 2: Sample face images from CMU PIE database. For each subject, there are 22 face images under different lighting conditions and fixed pose (C27) and expression.

Table 1: Clustering performance comparisons on CMU PIE database

k	Accuracy (%)				
	Kmeans	PCA	LPP	NCut	TensorImage
5	49.3	51.3	96.6	96.6	**99.95**
10	39.7	40.8	86.1	86.1	**92.95**
30	34.9	35.4	77.8	77.8	**84.32**
68	33.6	34.4	74.5	73.5	**82.23**
k	Mutual Information (%)				
	Kmeans	PCA	LPP	NCut	TensorImage
5	46.7	47.8	97.0	97.0	**99.88**
10	50.1	51.1	92.9	92.9	**96.95**
30	56.1	56.9	90.9	90.9	**94.95**
68	62.6	63.9	91.3	90.6	**95.20**

from the data corpus. The data points and the cluster number k are provided to the clustering algorithms. The clustering result is evaluated by comparing the obtained label of each data point with that provided by the data corpus. Two metrics, the accuracy (AC) and the normalized mutual information metric ($\overline{MI}$) are used to measure the clustering performance [19]. Given a data point $\mathbf{x}_i$, let r_i and s_i be the obtained cluster label and the label provided by the data corpus, respectively. The AC is defined as follows:

$$AC = \frac{\sum_{i=1}^{n} \delta(s_i, map(r_i))}{n} \tag{16}$$

where n is the total number of data points and $\delta(x, y)$ is the delta function that equals one if $x = y$ and equals zero otherwise, and map(r_i) is the permutation mapping function that maps each cluster label r_i to the equivalent label from the data corpus. The best mapping can be found by using the Kuhn-Munkres algorithm [11].

Given two sets of data clusters C, C', their mutual information metric $MI(C, C')$ is defined as:

$$MI(C, C') = \sum_{c_i \in C, c'_j \in C'} p(c_i, c'_j) \cdot log_2 \frac{p(c_i, c'_j)}{p(c_i) \cdot p(c'_j)} \tag{17}$$

where $p(c_i)$ and $p(c'_j)$ are the probabilities that a data point arbitrarily selected from the corpus belongs to the clusters c_i and c'_j, respectively, and $p(c_i, c'_j)$ is the joint probability that

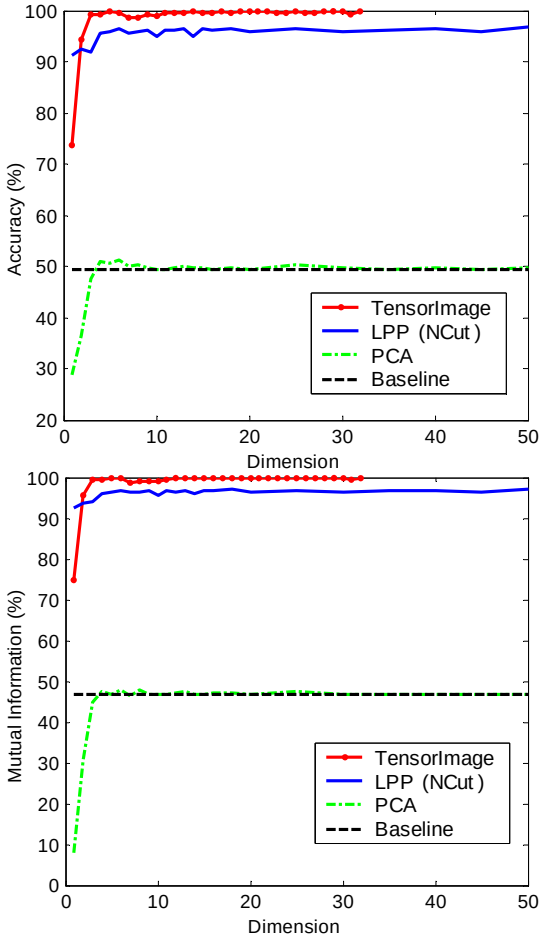

Figure 3: Clustering results on 5 randomly chosen classes

the arbitrarily selected data point belongs to the clusters c_i as well as c'_j at the same time. In our experiments, we use the normalized mutual information $\overline{MI}$ as follows:

$$\overline{MI}(C, C') = \frac{MI(C, C')}{\max(H(C), H(C'))} \tag{18}$$

where $H(C)$ and $H(C')$ are the entropies of C and C', respectively. It is easy to check that $\overline{MI}(C, C')$ ranges from 0 to 1. $\overline{MI} = 1$ if the two sets of clusters are identical, and $\overline{MI} = 0$ if the two sets are independent.

5.4 Clustering Results

The evaluations were conducted with different numbers of clusters. For each given class number k, k classes were randomly selected from the database. This process were repeated 50 times, and the average performance was computed. For each single test (given k classes of face images), we applied the above five methods. For each method, the k-means step was repeated 10 times with different initializations and the best result was recorded. For LPP, PCA, NCut, and TensorImage, they all need to estimate the dimensionality of the subspace. In general, their performance varies with the dimensionality of the subspace. For LPP, PCA and NCut, the images were projected into $\mathbb{R}^d (d < 1024)$. For TensorImage, the images were projected

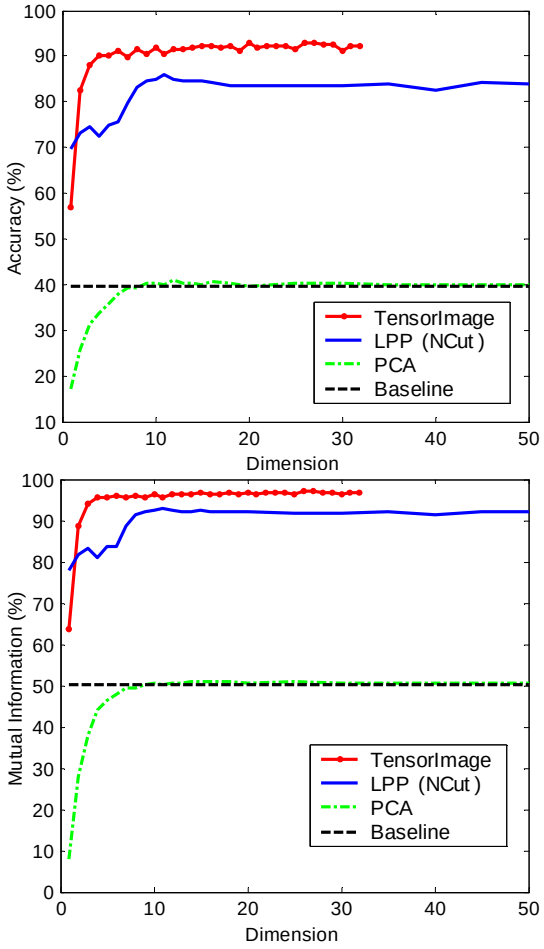

Figure 4: Clustering results on 10 randomly chosen classes

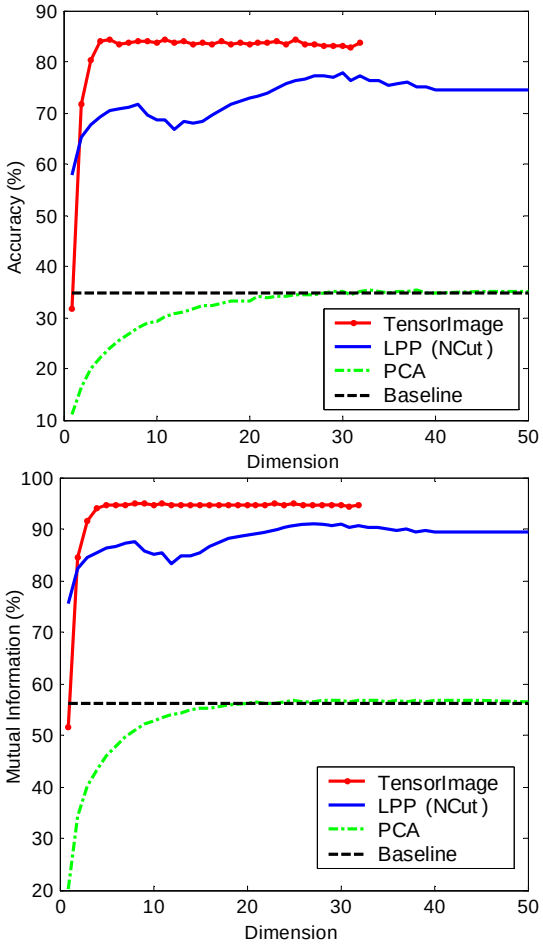

Figure 5: Clustering results on 30 randomly chosen classes

into $\mathcal{R}^d \otimes \mathcal{R}^d (1 \leq d < 32)$. Notice that, for TensorImage, the images can be actually projected into $\mathcal{R}^{d_1} \otimes \mathcal{R}^{d_2} (1 \leq d_1, d_2 < 32)$. In our experiment, we set $d_1 = d_2$ for the sake of simplicity. Figure 3-6 show the clustering performance of these algorithms as a function of the dimensionality of the subspace (d). Table 1 shows the best performance obtained by each algorithm. As can be seen, our clustering algorithm consistently outperformed PCA and LPP based clustering algorithms. Note that, when the number of classes (k) is 5, 10, or 30, the total number of images ($22 \times k$) is less than the number of features (1024). Therefore, NCut has the same result as LPP as suggested by Proposition 4.1. Moreover, the performance of PCA based clustering algorithm is almost the same as that of baseline. This shows that PCA fails to discover the intrinsic class structure of the image database. Finally, it can be seen that LPP, NCut and TensorImage achieved their best results at very low dimensions. This indicates that dimensionality reduction is a necessary preprocessing step for data clustering and classification.

6. CONCLUSIONS AND FUTURE WORK

In this paper we introduced a tensor framework for image representation and clustering. In particular, we considered the case that the images are sampled from a low dimensional submanifold embedded in a higher dimensional space. The algorithm and analysis presented in the paper are fundamentally based on the algebra of tensors and the geometry of manifolds.

Different from traditional learning algorithms such as PCA and LPP which consider an image as a vector, our algorithm considers an image as a matrix, or the second order tensor. By preserving the local structure of the image manifold, we obtained a tensor subspace for image representation. We call it *TensorImage* approach. By spectral relaxation, We have showed that the objective function of our algorithm is an attempt to find the optimal graph partitioning function based on the min-cut criteria while restricting the function to be linear. Furthermore, we have showed that TensorImages are actually linear approximations to the eigenfunctions of the Laplace Beltrami operator on the manifold. This indicates that the local structure of the image manifold can be preserved in the tensor subspace spanned by the TensorImages. Clustering experiments on PIE database show the efficiency and effectiveness of our algorithm.

There are several interesting problems that we are going to explore in the future work:

1. TensorImage is a linear method. Therefore, if the image manifold is highly nonlinear, it may fail to discover

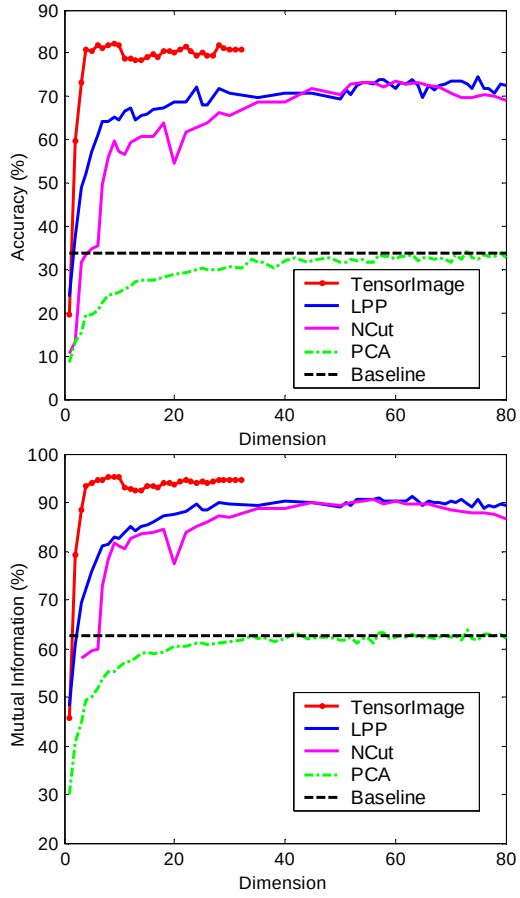

Figure 6: Clustering results on 68 randomly chosen classes

the intrinsic geometrical structures. It remains unclear how to generalize our algorithm to nonlinear case.

2. TensorImage is unsupervised. The local geometrical structure of the image manifold is modeled by a nearest neighbor graph. However, the graph constructed in this paper (Eqn. 2) may not be optimal in the sense of discrimination. When the label information is available, one may construct a graph according to the label information. For example, we can put an edge between two images if they belong to the same class.

3. The algorithm and analysis presented in this paper is primarily focused on images. But it can be easily extended to higher order tensors. For example, video can be thought of as the third order tensor. We expect that tensor based techniques can provide better representation for video.

7. REFERENCES

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, 2001.

[2] Y. Chen, J. Z. Wang, and R. Krovetz. Content-based image retrieval by clustering. In *Proceedings of the 5th ACM SIGMM International Workshop on Multimedia Information Retrieval*, pages 193–200, 11 2003.

[3] F. R. K. Chung. *Spectral Graph Theory*, volume 92 of *Regional Conference Series in Mathematics*. 1997.

[4] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, Hoboken, NJ, 2nd edition, 2000.

[5] S. Gordon, H. Greenspan, and J. Goldberger. Applying the information bottleneck principle to unsupervised clustering of discrete and continuous image representations. In *IEEE International Conference on Computer Vision*, Nice, France, 2003.

[6] S. Guattery and G. L. Miller. Graph embeddings and laplacian eigenvalues. *SIAM Journal on Matrix Analysis and Applications*, 21(3):703–723, 2000.

[7] X. He and P. Niyogi. Locality preserving projections. *Advances in Neural Information Processing Systems 16*, 2003.

[8] J. M. Lee. *Introduction to Smooth Manifolds*. Springer-Verlag New York, 2002.

[9] J. Lim, J. Ho, M.-H. Yang, K.-C. Lee, and D. Kriegman. Image clustering with metric, local linear structure and affinity symmetry. In *Proceedings of the European Conference on Computer Vision*, 2004.

[10] X. Liu, Y. Yu, and H.-Y. Shum. Synthesizing bidirectional texture functions for real-world surfaces. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 20:97–106, 2001.

[11] L. Lovasz and M. Plummer. *Matching Theory*. Akadémiai Kiadó, North Holland, Budapest, 1986.

[12] B. Mohar. Some applications of laplace eigenvalues of graphs. In *In G. Hahn and G. Sabidussi, editors, Graph Symmetry: Algebraic Methods and Applications*, 1997.

[13] A. Y. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, 2001.

[14] J.-M. Odobez, D. Gatica-Perez, and M. Guillemot. Spectral structuring of home videos. In *Proceedings of the 2nd International Conference on Image and Video Retrieval*, July 2003.

[15] J. C. Platt, M. Czerwinski, and B. A. Field. Phototoc: Automatic clustering for browsing personal photographs. In *Fourth IEEE Pacific Rim Conference on Multimedia*, Singapore, 2003.

[16] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[17] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[18] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[19] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of ACM SIGIR Conference on Information Retrieval*, 2003.

[20] X. Zheng, D. Cai, X. He, W.-Y. Ma, and X. Lin. Locality preserving clustering for image database. In *Proceedings of the ACM Conference on Multimedia*, October 2004.