Clustering and Dimensionality Reduction on Riemannian Manifolds

Alvina Goh René Vidal
Center for Imaging Science, Johns Hopkins University, Baltimore MD 21218, USA

Abstract

We propose a novel algorithm for clustering data sampled from multiple submanifolds of a Riemannian manifold. First, we learn a representation of the data using generalizations of local nonlinear dimensionality reduction algorithms from Euclidean to Riemannian spaces. Such generalizations exploit geometric properties of the Riemannian space, particularly its Riemannian metric. Then, assuming that the data points from different groups are separated, we show that the null space of a matrix built from the local representation gives the segmentation of the data. Our method is computationally simple and performs automatic segmentation without requiring user initialization. We present results on 2-D motion segmentation and diffusion tensor imaging segmentation.

1. Introduction

Segmentation is one of the most important problems in computer vision, thus it has been extensively studied. Its goal is to group the image data into clusters based upon image properties such as intensity, color, texture or motion.

Most existing segmentation algorithms proceed by associating a feature vector to each pixel in the image and then segmenting the data by clustering these feature vectors. When the structure of these features is simple enough, *central clustering* methods such as K-means or Expectation Maximization for a mixture of Gaussians [6] can be applied. More often, the features within each group are distributed in a subspace of the ambient space, hence *subspace clustering* techniques such as Generalized Principal Component Analysis (GPCA) [19] or EM for a mixture of probabilistic PCAs [18] need to be used. More generally, the features within each group can be distributed along a nonlinear submanifold of the ambient space. Here, one can use variations of the Euclidean distance to build a similarity matrix between pairs of features and then apply spectral clustering. Alternatively, one can extend nonlinear dimensionality reduction (NLDR) methods (often designed for one submanifold) to deal with multiple submanifolds. For instance, [15] combines Isomap [17] with EM, and [12, 8] combine LLE [14] with K-means.

Unfortunately, all these *manifold clustering* algorithms assume that the feature vectors are embedded in a Euclidean space and use (at least locally) the Euclidean metric or a variation of it to perform clustering. While this may be appropriate in some cases, there are several computer vision problems where it is more natural to consider features that live in a non-Euclidean space. For example, Grassmann manifolds and Lie groups are used for motion segmentation and multibody factorization [16], and symmetric positive semidefinite matrices are common in diffusion tensor imaging [1, 7, 11, 20] and structure tensor analysis [13].

The main contribution of this paper is the development of a novel framework for clustering data lying in different submanifolds of a Riemannian space. In particular, we consider three NLDR techniques, namely Laplacian Eigenmaps (LE) [2], Locally Linear Embedding (LLE) [14], and Hessian LLE (HLLE) [5], and show that they can be extended to deal with multiple submanifolds of a Riemannian space. At first sight one may think that this extension is straightforward, as it involves replacing the Euclidean metric by the Riemannian metric. This is indeed the case for LE, which relies only on the computation of the (geodesic) distances to the nearest neighbors of each data point. However, there are important challenges for other NLDR techniques. For example, LLE involves writing each data point as a linear combination of its neighbors. In the Euclidean case, this is simply a least-squares problem. In the Riemannian case, one needs to solve an interpolation problem on the manifold. How should the data points be interpolated? What cost function should be minimized? For HLLE, it involves computing the mean and a set of principal components from the neighborhood of each point. In the Euclidean case this can be done using PCA. In the Riemannian case one needs to find the mean and principal components on the manifold using e.g., Principal Geodesic Analysis (PGA) [7].

In addition, recall that our task is not only to apply NLDR to data on a manifold, but also to cluster data lying in multiple submanifolds. In this paper we show that when the different submanifolds are separated, LE, LLE and HLLE map all the points in one connected submanifold to a single point in the low-dimensional space. Hence, for separate submanifolds, LE, LLE and HLLE effectively reduce the manifold clustering problem to a standard central clustering problem. More specifically, we show that the segmentation

978-1-4244-2243-2/08/$25.00 ©2008 IEEE

of the data can be obtained from the null space of a matrix built from the local representation.

While we develop our manifold clustering framework for a generic Riemannian manifold, in applications the specific calculations vary depending on the explicit expressions for the Riemannian distance, geodesics, exponential and logarithm maps of the manifold. In this paper, we present applications to 2-D motion segmentation and diffusion tensor imaging segmentation that involve clustering on the space of symmetric semi-positive definite matrices. Our experiments show encouraging results on these challenging problems.

2. Preliminaries

Our algorithm for clustering submanifolds of a Riemannian space relies on basic concepts from Riemannian geometry and NLDR. For this purpose, in §2.1 we present a brief summary of the theory of Riemannian manifolds. We refer the reader to [4] for more details. In §2.2 we review three local NLDR algorithms, namely LLE, LE, and HLLE. While these may not be the best or the most recent NLDR algorithms, we have chosen them because they can be extended to Riemannian spaces, as we will show in §3.

2.1. Review of Riemannian Manifolds

A differentiable manifold $\mathcal{M}$ of dimension n is a topological space that is homeomorphic to the Euclidean space $\mathbb{R}^n$. The tangent space $T_{\boldsymbol{x}}\mathcal{M}$ at $\boldsymbol{x}$ is the vector space that contains the tangent vectors to all 1-D curves on $\mathcal{M}$ passing through $\boldsymbol{x}$. Fig. 1 shows an example of a two-dimensional manifold, a smooth surface living in $\mathbb{R}^3$. A *Riemannian metric* on a manifold $\mathcal{M}$ is a bilinear form which associates to each point $\boldsymbol{x} \in \mathcal{M}$, a differentiable varying inner product $\langle \cdot, \cdot \rangle_{\boldsymbol{x}}$ on the tangent space $T_{\boldsymbol{x}}\mathcal{M}$ at $\boldsymbol{x}$. The norm of a vector $\boldsymbol{v} \in T_{\boldsymbol{x}}\mathcal{M}$ is denoted by $\|\boldsymbol{v}\|_{\boldsymbol{x}}^2 = \langle \boldsymbol{v}, \boldsymbol{v} \rangle_{\boldsymbol{x}}$. The Riemannian distance between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ that lie on the manifold, $\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)$, is defined as the minimum length over all possible smooth curves on the manifold between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. The smooth curve with minimum length is known as the geodesic curve γ.

Given a tangent vector $\boldsymbol{v} \in T_{\boldsymbol{x}}\mathcal{M}$, locally there exists a unique geodesic $\gamma_{\boldsymbol{v}}(t)$ starting at $\boldsymbol{x}$ with initial velocity $\boldsymbol{v}$, and this geodesic has constant speed equal to $\|\boldsymbol{v}\|_{\boldsymbol{x}}$. The *exponential map*, $\exp_{\boldsymbol{x}} : T_{\boldsymbol{x}}\mathcal{M} \to \mathcal{M}$ maps a tangent vector $\boldsymbol{v}$ to the point on the manifold that is reached at time 1 by the geodesic $\gamma_{\boldsymbol{v}}(t)$. The inverse of $\exp_{\boldsymbol{x}}$ is known as the *logarithm map* and is denoted by $\log_{\boldsymbol{x}} : \mathcal{M} \to T_{\boldsymbol{x}}\mathcal{M}$. Now, if we have two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ on the manifold $\mathcal{M}$, the tangent vector to the geodesic curve from $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$ is defined as $\boldsymbol{v} = \overrightarrow{\boldsymbol{x}_i \boldsymbol{x}_j} = \log_{\boldsymbol{x}_i}(\boldsymbol{x}_j)$, and the exponential map takes $\boldsymbol{v}$ to the point $\boldsymbol{x}_j = \exp_{\boldsymbol{x}_i}(\log_{\boldsymbol{x}_i}(\boldsymbol{x}_j))$. In addition, $\gamma_{\boldsymbol{v}}(0) = \boldsymbol{x}_i$ and $\gamma_{\boldsymbol{v}}(1) = \boldsymbol{x}_j$. The Riemannian distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined as $\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\log_{\boldsymbol{x}_i}(\boldsymbol{x}_j)\|_{\boldsymbol{x}_i}$. Linear geodesic interpolation makes use of the exponential and logarithm maps and is given by $\widehat{\boldsymbol{x}} = \exp_{\boldsymbol{x}_i}(w\overrightarrow{\boldsymbol{x}_i \boldsymbol{x}_j})$, $w \in [0, 1]$. Finally, the Riemannian metric, the exponential map and the logarithm map depend on the point $\boldsymbol{x}$ under consideration, hence the subscripts reflecting this dependency.

We will briefly summarize how to calculate the mean and principal components of data points lying on a manifold [7]. The Karcher or intrinsic mean $\overline{\mathbf{x}}$ is defined as $\overline{\mathbf{x}} \doteq \arg\min_{\boldsymbol{x} \in \mathcal{M}} \sum_{i=1}^{n} \text{dist}(\boldsymbol{x}, \boldsymbol{x}_i)^2 = \arg\min_{\boldsymbol{x} \in \mathcal{M}} \sum_{i=1}^{n} \|\log_{\boldsymbol{x}}(\boldsymbol{x}_i)\|_{\boldsymbol{x}}^2$. Since $\overline{\mathbf{x}}$ is the solution to a minimization problem, there is no guarantee that $\overline{\mathbf{x}}$ exists or is unique. However, by assuming that the data lie in a small enough neighborhood of $\boldsymbol{x}$, we can show the existence and uniqueness of $\overline{\mathbf{x}}$. Algorithm 1 shows the computation of $\overline{\mathbf{x}}$. Calculating principal geodesic components on a Riemannian manifold involves projecting a point onto a geodesic curve, which is also defined as a minimization problem for which existence and uniqueness are not ensured [7]. Again, by making the assumptions that the data lie in a small neighborhood about the mean $\overline{\mathbf{x}}$, the projection can be shown to be unique. [7] also shows that finding principal geodesic components boils down to doing PCA in the tangent vectors $\log_{\overline{\mathbf{x}}}(\boldsymbol{x}_i) \in T_{\overline{\mathbf{x}}}\mathcal{M}$ about the mean $\overline{\mathbf{x}}$ and proposes Principal Geodesic Analysis (PGA) summarized in Algorithm 2.

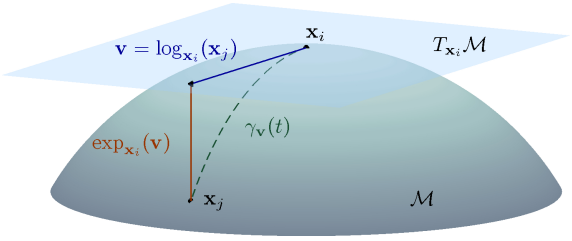


Figure 1. A two-dimensional manifold. The tangent plane at $\boldsymbol{x}_i$, together with the exp and log maps relating $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, are shown.

Algorithm 1 (Intrinsic Mean)

Given data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{M}$, a predefined threshold ϵ, the maximum number of iterations T,

1. Initialize $t = 1$, $\overline{\mathbf{x}}_1 = \boldsymbol{x}_i$ for a random i.
2. While $t \leq T$ or $\|\boldsymbol{v}\|_{\overline{\mathbf{x}}} \geq \epsilon$,
   (a) Compute tangent vector $\boldsymbol{v} = \frac{1}{n} \sum_{i=1}^{n} \log_{\overline{\mathbf{x}}_t}(\boldsymbol{x}_i)$.
   (b) Set $\overline{\mathbf{x}}_{t+1} = \exp_{\overline{\mathbf{x}}_t}(\boldsymbol{v})$

2.2. Review of Local Nonlinear Dimensionality Reduction Methods in Euclidean Spaces

Let $X = \{\boldsymbol{x}_i \in \mathbb{R}^D\}_{i=1}^n$ be a set of n data points sampled from a d-dimensional manifold embedded in $\mathbb{R}^D$, $d \ll D$. We assume that the n points are **k-connected**, i.e. for any two points $\boldsymbol{x}_i, \boldsymbol{x}_j \in X$ there is an ordered sequence of points in X having $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ as endpoints, such that any two consecutive points in the sequence have at least one

Algorithm 2 (Principal Geodesic Analysis)

Given data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{M}$,

1. Compute intrinsic mean $\overline{\mathbf{x}}$ as in Algorithm 1.
2. Calculate the tangent vectors $\boldsymbol{v}_i = \log_{\overline{\mathbf{x}}}(\boldsymbol{x}_i)$ about $\overline{\mathbf{x}}$.
3. Construct the covariance matrix $\text{cov}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{v}_i \boldsymbol{v}_i^\top$.
4. Perform eigenanalysis of the matrix $\text{cov}(\boldsymbol{x})$ to obtain the eigenvectors $\{\mathbf{u}_i\}_{i=1}^{d}$ giving the principal directions. $\{\mathbf{u}_i\}_{i=1}^{d}$ forms an orthonormal basis for $T_{\overline{\mathbf{x}}}\mathcal{M}$.

k-nearest neighbor in common. The goal of dimensionality reduction is to find a set of vectors $\{\mathbf{y}_i \in \mathbb{R}^d\}_{i=1}^n$, such that nearby points remain close and distant points remain far.

Locally Linear Embedding (LLE) [14] assumes that the local neighborhood of a point on the manifold can be well approximated by the affine subspace spanned by the k-nearest neighbors of the point and finds a low-dimensional embedding of the data based on these affine approximations. **Laplacian Eigenmaps (LE)** [2] are based on computing the low dimensional representation that best preserves *locality* instead of *local linearity* in LLE. **Hessian LLE (HLLE)** [5] bears substantial resemblance to LE, with the main difference being that the Laplacian matrix is replaced by the Hessian matrix. The three algorithms are summarized as follows. Notice that steps 1 and 4 are common.

Locally Linear Embedding (LLE) [14]

1. *Nearest neighbor search:* Find the k-nearest neighbors (kNN) of each $\boldsymbol{x}_i$ according to the Euclidean distance.
2. *Weight matrix*: Find a matrix of weights $W \in \mathbb{R}^{n\times n}$ whose entries W_{ij} minimize the reconstruction error

$$\varepsilon(W) = \sum_{i=1}^{n} \| \sum_{j=1}^{n} W_{ij}\boldsymbol{x}_j - \boldsymbol{x}_i\|^2 = \sum_{i=1}^{n} \text{dist}^2(\widehat{\boldsymbol{x}_i}, \boldsymbol{x}_i) \quad (1)$$

subject to the constraints (i) $W_{ij} = 0$ if $\boldsymbol{x}_j$ is not a k-nearest neighbor of $\boldsymbol{x}_i$ and (ii) $\sum_{j=1}^{n} W_{ij} = 1$. In (1), $\widehat{\boldsymbol{x}_i} = \boldsymbol{x}_i + \sum_{j=1}^{n} W_{ij}\overrightarrow{\boldsymbol{x}_i\boldsymbol{x}_j}$ is the linear interpolation of $\boldsymbol{x}_i$ and its kNN. Also, for each data point $\boldsymbol{x}_i$ the nonzero entries corresponding to the i-th row of W are given by

$$W_i = \frac{\mathbf{1}^\top C_i^{-1}}{\mathbf{1}^\top C_i^{-1}\mathbf{1}}, \quad (2)$$

where $C_i \in \mathbb{R}^{k\times k}$ is the local Gram matrix at $\boldsymbol{x}_i$, i.e. $C_i(j,l) = (\boldsymbol{x}_j - \boldsymbol{x}_i)\cdot(\boldsymbol{x}_l - \boldsymbol{x}_i)$, and $\mathbf{1} \in \mathbb{R}^k$ is the vector of all ones.

3. *Objective function*: Find vectors $\{\mathbf{y}_i \in \mathbb{R}^d\}_{i=1}^n$ that minimize the error

$$\phi(Y) = \sum_{i=1}^{n} \|\mathbf{y}_i - \sum_{j=1}^{n} W_{ij}\mathbf{y}_j\|^2 = \text{trace}(Y^\top M Y), \quad (3)$$

where $M = (I-W)^\top(I-W)$, $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_n]^\top \in \mathbb{R}^{n\times d}$, subject to the constraints (i) $\sum_{i=1}^{n}\mathbf{y}_i = \mathbf{0}$ and (ii) $\frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_i\mathbf{y}_i^\top = I$.

Laplacian Eigenmaps (LE) [2]

2. *Weight matrix*: Construct a matrix of weights $W \in \mathbb{R}^{n\times n}$

$$W_{ij} = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/\sigma^2) \quad (4)$$

subject to the constraint $W_{ij} = 0$ if $\boldsymbol{x}_j$ is not a k-nearest neighbor of $\boldsymbol{x}_i$. The entries of W, W_{ij}, measure the proximity between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

3. *Objective function*: Find vectors $\{\mathbf{y}_i \in \mathbb{R}^d\}_{i=1}^n$ that minimize the following objective function

$$\phi(Y) = \sum_{i,j} \frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2 W_{ij}}{\sqrt{D_{ii}D_{jj}}} = \text{trace}(Y^\top L Y) \quad (5)$$

where $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_n]^\top \in \mathbb{R}^{n\times d}$, D is diagonal with $D_{ii} = \sum_j W_{ij}$, and $L = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$.

Hessian LLE (HLLE) [5]

2. *Tangent coordinates*: For each data point $\boldsymbol{x}_i$, let $\{\boldsymbol{x}_{i,j}\}_{j=1}^k$ be its kNN. Form the D by D covariance matrix $\text{cov}(\boldsymbol{x}_i) = \frac{1}{k}\sum_{j=1}^{k}(\boldsymbol{x}_{i,j} - \bar{\boldsymbol{x}}_i)(\boldsymbol{x}_{i,j} - \bar{\boldsymbol{x}}_i)^\top$, where $\bar{\boldsymbol{x}}_i$ is the mean of the kNN. Perform an eigenanalysis of the matrix $\text{cov}(\boldsymbol{x}_i)$ to obtain the d eigenvectors $\{\mathbf{u}_q \in \mathbb{R}^D\}_{q=1}^d$. The tangent coordinates of the kNN are given by the d columns of the $k \times d$ matrix V given below, where $p = 1, \ldots, k$ and $q = 1, \ldots, d$

$$V_{pq} = (\boldsymbol{x}_{i,p} - \bar{\boldsymbol{x}}_i)^\top \mathbf{u}_q = \langle \boldsymbol{x}_{i,p} - \bar{\boldsymbol{x}}_i, \mathbf{u}_q \rangle. \quad (6)$$

3. *Objective function*: The embedding vectors are obtained as the null vectors of a matrix H that indicates the Hessian quadratic cost. While we refer the reader to [5] for details on the estimation of H, the basic principle is as follows. We first locally estimates a Hessian operator h^i at each point $\boldsymbol{x}_i$ in the manifold in a least squares sense. In particular, consider a smooth function $f : \mathcal{M} \to \mathbb{R}$. We evaluate the function at all kNN of a point $\boldsymbol{x}_i$ on the manifold and stack these entries into a vector $\mathbf{f}_i$. It can be shown that $h^i\mathbf{f}_i$ approximates the entries of the Hessian, whose (p,q)-th entry is given by $\frac{\partial^2 f}{\partial V_p \delta V_q}$. These local estimates are then used to obtain an empirical estimate of the (i,j)-th entry of H as

$$H_{i,j} = \sum_l \sum_r ((h^l)_{r,i}(h^l)_{r,j}). \quad (7)$$

4. *Sparse eigenvalue problem*: Let $\widetilde{M}$ be M in LLE, L in LE, and H in HLLE. Perform an eigenanalysis of $\widetilde{M}$ and obtain the $(d+1)$ eigenvectors associated with the smallest $(d+1)$ eigenvalues. The vector of all ones, $\mathbf{1} \in \mathbb{R}^n$, is an eigenvector of $\widetilde{M}$ associated with the eigenvalue 0. The d eigenvectors of the matrix $\widetilde{M}$ associated with its second to $(d+1)$-th smallest eigenvalues correspond to eigenvectors spanning a d-dimensional space which contains the embedding coordinates, i.e. the columns of Y. Since $\widetilde{M}$ is symmetric, one can choose the embedding vectors to be orthogonal to the vector $\mathbf{1}$.

3. Clustering on Riemannian Manifolds

We now have the necessary tools to develop the theory and present our algorithm for clustering data on Riemannian manifolds. We first proceed by extending existing NLDR algorithms to Riemannian manifolds. We then show that by making use of the mappings NLDR generate, the problem of manifold clustering reduces to a central clustering problem.

3.1. Extending NLDR to Riemannian Manifolds

The NLDR techniques presented in §2.2 are applicable only in the presence of *one* manifold with *unknown structure*. Furthermore, as the metric is unknown, every operation is approximated by the corresponding Euclidean operation. However, in computer vision problems, it is common that the spaces under consideration have well-studied geometries and closed-form formulae for Riemannian operations are available. The task at hand is to develop NLDR techniques for Riemannian manifolds in a way that takes into consideration the appropriate Riemannian structure.

Since the information about the local geometry of the manifold is essential only in the first two steps of each algorithm, modifications are made only to these two stages. The key issues are how to select the kNN and how to compute the matrix $\widetilde{M}$ representing the local geometry. The former is easy to deal with, while the latter is where our contribution lies. Given $\widetilde{M}$, calculating the low-dimensional representation remains the same as in the Euclidean case.

In what follows, we let $X = \{\boldsymbol{x}_i \in \mathbb{R}^D\}_{i=1}^n$ be a set of n data points sampled from a known Riemannian manifold.

3.1.1 Selection of the Riemannian kNN

The first step of any NLDR algorithm is the computation of the kNN associated with each data point. Instead of using the Euclidean distance, we define the kNN of $\boldsymbol{x}_i$ by incorporating the Riemannian distance. That is, *the kNN of $\boldsymbol{x}_i$ are the k data points $\boldsymbol{x}_j$ that minimize* $\|\log_{\boldsymbol{x}_i}(\boldsymbol{x}_j)\|_{\boldsymbol{x}_i}$.

3.1.2 Riemannian Calculation of $\widetilde{M}$ for LLE

The second step of LLE is to compute the matrix of weights $W \in \mathbb{R}^{n\times n}$. The two main questions are: what is the reconstruction cost? and how does one express a point as a linear combination of its neighbors? First of all, instead of minimizing the Euclidean error, we rewrite (1) to minimize the Riemannian reconstruction error,

$$\varepsilon_{Riem}(W) = \sum_{i=1}^{n} \left\| \log_{\boldsymbol{x}_i}(\widehat{\boldsymbol{x}}_{Riem,i}) \right\|_{\boldsymbol{x}_i}^2 \tag{8}$$

subject to $W_{ij} = 0$ if $\boldsymbol{x}_j$ is not a kNN of $\boldsymbol{x}_i$ and $\sum_j W_{ij} = 1$. $\widehat{\boldsymbol{x}}_{Riem,i}$, is the geodesic linear interpolation of $\boldsymbol{x}_i$ by its kNN

$$\widehat{\boldsymbol{x}}_{Riem,i} = \exp_{\boldsymbol{x}_i}\Big(\sum_{j=1}^{n} W_{ij} \log_{\boldsymbol{x}_i}(\boldsymbol{x}_j)\Big). \tag{9}$$

Since exp and log are inverse mappings, (8) becomes

$$\varepsilon_{Riem}(W) = \sum_{i=1}^{n} \Big\| \sum_{j=1}^{n} W_{ij} \log_{\boldsymbol{x}_i}(\boldsymbol{x}_j) \Big\|_{\boldsymbol{x}_i}^2. \tag{10}$$

Using similar manipulation as in the Euclidean case, the optimal weights are obtained as in (2), with the local Gram matrix $C_i \in \mathbb{R}^{k\times k}$ defined as

$$C_i(j,l) = \langle \log_{\boldsymbol{x}_i}(\boldsymbol{x}_j), \log_{\boldsymbol{x}_i}(\boldsymbol{x}_l) \rangle_{\boldsymbol{x}_i}. \tag{11}$$

$\widetilde{M}$ is then $(I - W)^\top (I - W)$.

3.1.3 Riemannian Calculation of $\widetilde{M}$ for LE

Here, instead of attempting to write each data point as a linear combination of its kNN, we find a matrix of weights $W \in \mathbb{R}^{n\times n}$ whose entries W_{ij} measure the proximity between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ as in (4). Therefore, modifying Laplacian eigenmaps for Riemannian manifolds is less involved than in the case of LLE. Instead of using $\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/\sigma^2)$ as in (4), we construct the weight matrix W using the Riemannian distance as

$$W_{ij} = \exp^{-\frac{\mathrm{dist}_{Riem}(\boldsymbol{x}_i,\boldsymbol{x}_j)^2}{\sigma^2}} = \exp^{-\frac{\|\log_{\boldsymbol{x}_i}(\boldsymbol{x}_j)\|_{\boldsymbol{x}_i}^2}{\sigma^2}} \tag{12}$$

subject to the constraint $W_{ij} = 0$ if $\boldsymbol{x}_j$ is not a k-nearest neighbor of $\boldsymbol{x}_i$. Now $\widetilde{M} = L = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ and D is a diagonal matrix, where $D_{ii} = \sum_j W_{ij}$, as before.

3.1.4 Riemannian Calculation of $\widetilde{M}$ for HLLE

The second step of HLLE is to compute the tangent coordinates for each $\boldsymbol{x}_i$ by applying Euclidean PCA to its neighbors. This makes the implicit approximation that these local points lie on a subspace. This assumption is no longer valid if $\boldsymbol{x}_i$ and its kNN lie on a Riemannian manifold. From §2.1, we know that in this case, calculating the principal geodesic components and the projection coordinates is not as simple as doing Euclidean PCA. There is a need to incorporate the correct Riemannian metric, mean and covariance matrix.

Again, let $\{\boldsymbol{x}_{i,j}\}_{j=1}^k$ denote the set of k-nearest neighbors of $\boldsymbol{x}_i$. First we calculate the intrinsic mean $\bar{\boldsymbol{x}}_i$ of the kNN (Algorithm 1). Next, we find the tangent vectors $\boldsymbol{v}_j = \log_{\bar{\boldsymbol{x}}_i}(\boldsymbol{x}_{i,j})$ about $\bar{\boldsymbol{x}}_i$ and the geodesic principal directions $\{\mathbf{u}_q\}_{q=1}^d$ using PGA (Algorithm 2). Since $\{\mathbf{u}_q \in \mathbb{R}^D\}_{q=1}^d$ is an orthonormal basis for $T_{\bar{\boldsymbol{x}}_i}\mathcal{M}$, we will rewrite the projection operator in (6) using the Riemannian metric. Thus the tangent coordinates of the kNN are given by the $k \times d$ matrix V, where

$$V_{pq} = \langle \log_{\bar{\boldsymbol{x}}_i}(\boldsymbol{x}_{i,p}), \mathbf{u}_q \rangle_{\bar{\boldsymbol{x}}_i},\ p = 1,..,k,\ q = 1,..,d. \tag{13}$$

Once the tangent coordinates are found, the estimation of the Hessian matrix $\widetilde{M}$ is the same as in the Euclidean case (7).

3.1.5 Calculation of the Embedding Coordinates

The last step of NLDR is to find a Euclidean low-dimensional representation of the data points. As this step is independent of the Riemannian structure, one can find the embedding coordinates as described in §2.2. That is, the embedding coordinates are the d eigenvectors of the matrix $\widetilde{M}$ associated with its second to $(d+1)$-th smallest eigenvalues and these correspond to eigenvectors spanning a d-dimensional space that the low-dimensional representation lies on.

Finally, we see that the modifications needed in order to account for the Riemannian structure of the data do not require significant additional computational complexity. With the exception of the calculation of the intrinsic mean, closed-form formulae are available for the remaining operations.

3.2. Local Riemannian Manifold Clustering

In this section, we extend NLDR algorithms for the purpose of clustering data lying in m submanifolds of a Riemannian space. We assume that the data is distributed in a k-disconnected union of m k-connected submanifolds of $\mathcal{M}$. We show that under this assumption, each of the m submanifolds will be mapped to a different point in $\mathbb{R}^m$.

With real data, the assumption of a k-disconnected union will be violated. Nevertheless it is reasonable to expect that instead of mapping data points on a manifold to a single point, the mapping will generate a collection of n points distributed around m cluster centers. While a similar result for Euclidean LLE has been proposed in [12, 8], we show a generalized result that is applicable to Riemannian LLE, Riemannian LE and Riemannian HLLE. Proposition 1 shows that in the case of a disconnected union of m k-connected submanifolds, the matrix $\widetilde{M}$ has at least m zero eigenvalues, whose eigenvectors give the clustering of the data.

Proposition 1 *Let $\{\boldsymbol{x}_i\}_{i=1}^n$ be a set of points drawn from a disconnected union of m k-connected d-dimensional submanifolds of a Riemannian manifold. Then, there exist m vectors $\{\mathbf{u}_j\}_{j=1}^m$ in the null space of $\widetilde{M}$ such that $\mathbf{u}_j$ corresponds to the j-th group of points, i.e. $\mathbf{u}_{ij} = 1$ if the i-th data point is in the j-th group, and $\mathbf{u}_{ij} = 0$ otherwise.*

Proof. The proof for all three algorithms is similar. Since the data can be partitioned into m k-connected groups, the matrix $\widetilde{M}$ is block-diagonal with m blocks, because if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong to different groups, then they cannot be kNN of each other, hence $\widetilde{M}_{ij} = 0$. We can then write $\widetilde{M} = \text{diag}(\widetilde{M}_j)$, where $\widetilde{M}_j \in \mathbb{R}^{n_j \times n_j}$ is the matrix for the j-th group, which contains n_j points. From the properties of the algorithms, we know that each one of the m blocks of $\widetilde{M}$, has the vector $\mathbf{1} \in \mathbb{R}^{n_j}$ in its null space. Therefore, there are m vectors $\{\mathbf{u}_j\}$ in $\ker(\widetilde{M})$, with each $\mathbf{u}_j$ taking the values 1 and 0, indicating the group membership. ∎

We see that there exists a mapping $g : \mathcal{M} \to \mathbb{R}^m$ that gives the membership of each point to the m submanifolds. This mapping is given by the rows of any basis for $\ker(\widetilde{M})$. However, notice that we do not necessarily obtain the set of membership vectors $\{\mathbf{u}_j\}$ when computing a basis for $\ker(\widetilde{M})$, but rather linear combinations of them, including the vector $\mathbf{1}$. In general, linear combinations of segmentation eigenvectors still contain the segmentation of the data. Hence, we can cluster the data into m groups by applying k-means to the columns of a matrix whose rows are the m eigenvectors in the null space of $\widetilde{M}$. Algorithm 3 summarizes our dimensionality reduction and clustering algorithm for m submanifolds of a Riemannian space.

Algorithm 3 (Unsupervised Clustering and Dimensionality Reduction on Riemannian Manifolds)

Given data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{M}$,

1. *Nearest neighbors:* Find the kNN of each data point $\boldsymbol{x}_i$ according to the Riemannian distance as in §3.1.1.
2. *Construction of $\widetilde{M}$:* For each NLDR algorithm, construct the appropriate $\widetilde{M}$ as described in §3.1.2-§3.1.4.
3. *Clustering:* Compute the m eigenvectors $\{\mathbf{u}_j\}_{j=1}^m$ of $\widetilde{M}$ associated with its m smallest eigenvalues and apply k-means to the rows of $[\mathbf{u}_1, \cdots, \mathbf{u}_m]$ to cluster the data into m different groups.
4. *Low-dimensional embedding:* Apply NLDR to each group to obtain a low-dimensional embedding for each submanifold.

4. Application and Experiments on SPSD(3)

In this section, we will present an application of the theory developed in §3 to the space of 3 by 3 symmetric positive semi-definite matrices SPSD(3). It is well-known [1, 7, 11, 20] that the traditional Euclidean distance is not the most appropriate metric for SPSD matrices as they lie on a Riemannian symmetric space. An example of such data is the well-known structure tensor found in direct 2-D motion segmentation from the image intensities *without* extracting features such as optical flow or point correspondences. Under the assumption that all surfaces are Lambertian, the optical flow (u, v) between two images of a sequence is related to the image partial derivatives $\nabla I = (I_x, I_y, I_t)$ by $I_x u + I_y v + I_t = 0 \Rightarrow \nabla I^\top (u, v, 1) = 0$, where (x, y) denotes pixel location and t denotes time. Premultiplying by ∇I gives an equation of the form $(\nabla I \nabla I^\top)(u, v, 1) = 0$, This system of linear equations involves the spatial-temporal structure tensor $(\nabla I \nabla I^\top)$. SPSD matrices also play an important role in Diffusion Tensor Imaging (DTI). DTI is a 3-D imaging technique that measures the diffusion of water molecules in living tissues. Water diffusion is represented mathematically with a symmetric positive semi-definite tensor field $\mathbf{T} : \mathbb{R}^3 \to \text{SPSD}(3) \subset \mathbb{R}^{3\times 3}$ that measures the diffusion in a direction $\mathbf{d} \in \mathbb{R}^3$ as $\mathbf{d}^\top \mathbf{T} \mathbf{d}$.

Our goal is to automatically segment a set of SPSD ma-

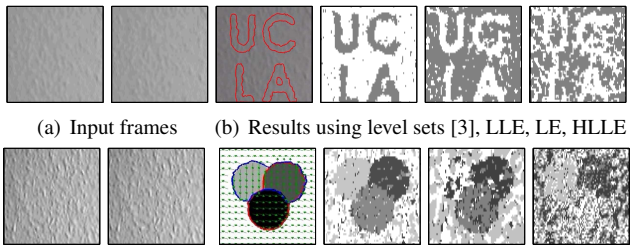
trices $\{\mathbf{T}_j \in \text{SPSD}(r)\}_{j=1}^n$ into different clusters, where different groups correspond to different 2-D motions in a video or to different fiber bundles in DTI. There are many possible metrics in SPSD(r) [1, 7, 9, 11, 20]. Each metric is derived from different geometrical, statistical or information-theoretic considerations. The question of which one is the best metric remains an active research area. In this paper, we will use the Riemannian metric proposed in [11] $\text{dist}_{Riem}(\mathbf{T}_i, \mathbf{T}_j) = \|\log(\mathbf{T}_i^{-\frac{1}{2}}\mathbf{T}_j\mathbf{T}_i^{-\frac{1}{2}})\|_F$, where $\|\cdot\|_F$ is the Frobenius norm and $\log(\cdot)$ is the matrix logarithm.

For this metric, the exponential map is defined as $\exp_{\mathbf{T}_i}(\mathbf{V}) = \mathbf{T}_i^{\frac{1}{2}}\exp(\mathbf{T}_i^{-\frac{1}{2}}\mathbf{V}\mathbf{T}_i^{-\frac{1}{2}})\mathbf{T}_i^{\frac{1}{2}}$, where $\exp(\cdot)$ is the matrix exponential and $\mathbf{V} \in T_{\mathbf{T}_i}\text{SPSD}(r)$. The logarithm map is $\overrightarrow{\mathbf{T}_i\mathbf{T}_j} = \log_{\mathbf{T}_i}(\mathbf{T}_j) = \mathbf{T}_i^{\frac{1}{2}}\log(\mathbf{T}_i^{-\frac{1}{2}}\mathbf{T}_j\mathbf{T}_i^{-\frac{1}{2}})\mathbf{T}_i^{\frac{1}{2}}$. The Gram matrix is $C_i(j,l) = \langle\log_{\mathbf{T}_i}(\mathbf{T}_j), \log_{\mathbf{T}_i}(\mathbf{T}_l)\rangle_{\boldsymbol{x}_i} = \text{trace}(\log(\mathbf{T}_i^{-\frac{1}{2}}\mathbf{T}_j\mathbf{T}_i^{-\frac{1}{2}})\log(\mathbf{T}_i^{-\frac{1}{2}}\mathbf{T}_l\mathbf{T}_i^{-\frac{1}{2}}))$. The geodesic linear interpolation $\widehat{\mathbf{T}}_{Riem,i}$ of $\{\mathbf{T}_j \in \text{SPSD}(r)\}_{j=1}^n$ about $\mathbf{T}_i$ with weights $W_{i1}, \ldots, W_{in}$ is given by $\mathbf{T}_i^{\frac{1}{2}}\exp\big(\sum_{j=1}^n W_{ij}\log(\mathbf{T}_i^{-\frac{1}{2}}\mathbf{T}_j\mathbf{T}_i^{-\frac{1}{2}})\big)\mathbf{T}_i^{\frac{1}{2}}$.

4.1. 2-D Motion Segmentation

We test our algorithm on 2-D motion segmentation from two consecutive frames of video sequences. The spatial-temporal structure tensor is $\mathbf{T} = K * (\nabla I \nabla I^\top)$, where $*$ is the convolution operator, K is a smoothing kernel (the Gaussian kernel is commonly used), and $\nabla I = (I_x, I_y, I_t)$ is the spatial-temporal image gradient. We use the same data set as in [3]. Fig. 2 shows two examples of moving patches of homogeneously textured wallpaper in which the different regions cannot be distinguished on the sole basis of appearance. This is because the input frames contain regions with the same intensity and texture with no clear edges or corners. Thus, all results are obtained exclusively from the motion information. Fig. 2(a) contains two regions, the text region with the word "UCLA" and the background. In Fig. 2(c), there are three overlapping circles, each with its own motion, and the background. As shown in Fig. 2(b) and 2(d), LLE yields the best results among all the algebraic methods, distinguishing the text "UCLA" and the three circles. As none of the NLDR methods incorporates a smoothness constraint (as done in level set methods), it is of no surprise that the level set method produces a cleaner segmentation. Nevertheless, it is immediate that our method provides a very good initialization for iterative techniques such as level sets.

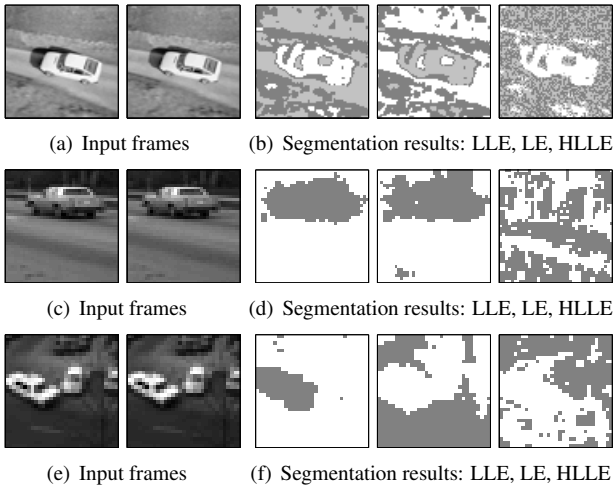
The next set of experiments is done on real video sequences. The first video involves a camera tracking a car going along a road, as shown in Fig. 3(a). There are three different motion groups found in the two consecutive frames. The first group contains mostly the pixels of the car, the second group contains the background pixels where the camera movement is apparent (e.g., edges and corners), and the last group contains the background pixels with the aperture



(a) Input frames (b) Results using level sets [3], LLE, LE, HLLE

(c) Input frames (d) Results using level sets [3], LLE, LE, HLLE

Figure 2. 2-D motion segmentation using the structure tensor.



(a) Input frames (b) Segmentation results: LLE, LE, HLLE

(c) Input frames (d) Segmentation results: LLE, LE, HLLE

(e) Input frames (f) Segmentation results: LLE, LE, HLLE

Figure 3. 2-D motion segmentation on real video sequences.

problem (e.g., middle of the road). The second video of a car is taken with a stationary camera, as shown in Fig. 3(c). There are two different motion groups in this case, the first group being the car and the second group being the background. The last video, shown in Fig. 3(e), is taken from the Hamburg Taxi sequence. In this dataset, the moving taxi forms the first group and the stationary background forms the second group. From Figs. 3(b), 3(d) and 3(f), it is clear that LLE is able to segment the different groups, LE gives a reasonable segmentation but suffers from artifacts, whereas the performance of HLLE performance is poor.

4.2. Diffusion Tensor Imaging Segmentation

We also test our proposed algorithm in the segmentation of the corpus callosum and the cingulum from real DTI data. The size of the entire DTI volume of the brain is $128 \times 128 \times 58$ voxels and the voxel size is $2 \times 2 \times 2$ mm. From the visualization of the tensor data, we know the approximate location of each cingulum bundle in the left and right hemispheres. Hence, we reduce the input volume to the algorithm by focusing in this location. In addition, we also mask out voxels with fractional anisotropy below a threshold of 0.2 in order to separate white matter from the rest of the brain. Also, tensors at adjacent voxels within a fiber bundle are similar (similar eigenvectors and eigenvalues), while tensors at distant voxels could be very different, even if they

lie in the same bundle. In order to account for this fact we choose the kNN $\{\mathbf{D}(x_j)\}_{j=1}^{k}$ of a tensor $\mathbf{D}(x_i)$ at x_i subject to $\|x_j - x_i\| \leq R$. We set the value of the spatial radius to $R = 10$ and the number of nearest neighbors to $k = 30$.

Fig. 4 shows the results of the left hemisphere. Fig. 4(a) shows the sagittal slices used and the ellipsoid visualization of the tensors. The corpus callosum is the bundle with the red tensors pointing out of the plane and resembles the letter 'C'. The cingulum, which is significantly smaller, is the bundle left to the corpus callosum with the green tensors oriented vertically. We see that the corpus callosum and the cingulum are clustered around different centers. Fig. 4(b) shows the results of LLE. The corpus callosum forms a distinct cluster (in red). Fig. 4(c) shows the results of LE. Even though it appears that the cingulum forms a distinct group, the corpus callosum is merged into the same group as the tensors in the background. HLLE (not shown) failed to produce any reasonable segmentation of the fiber bundles.

As our algorithm does not incorporate any smoothness constraint, our segmentation is noisier compared to energy minimization methods such as in [10]. However, for the segmentation of the cingulum bundle in [10], a significant effort was required to manually remove voxels in the corpus callosum before running their respective algorithms. Our algorithm, on the other hand, is automatic. Hence, an immediate use for our method is that the output could be used as an automatic initialization for such algorithms.

5. Conclusion

We have proposed a novel algorithm for clustering data sampled from multiple submanifolds of a Riemannian manifold with known structure. In motion segmentation and DTI segmentation, HLLE performs badly and this is due to the poor estimation of the Hessian matrix. HLLE depends critically on the prior knowledge of the intrinsic dimension of the submanifolds, which is unknown beforehand. Results on the space of symmetric positive definite matrices are encouraging. In motion segmentation, our algorithm deals directly with the image intensities and provides a very good initialization for iterative methods. Results on DTI show that it is possible to segment the data into different fiber bundles.

Acknowledgements

Work supported by startup funds from JHU, by grants NSF CAREER IIS-0447739, NSF EHS-0509101, NIH RO1 HL082729, and ONR N00014-05-10836, and by contract JHU APL-934652.

References

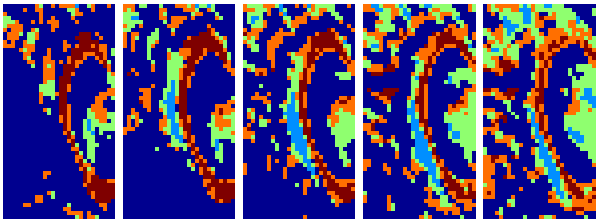
[1] V. Arsigny et al. Log-Euclidean metrics for fast and simple calculus on diffusion tensors. *Mag. Reson. Med.*, 56:411–421, 2006.

[2] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, pg 585–591, 2002.

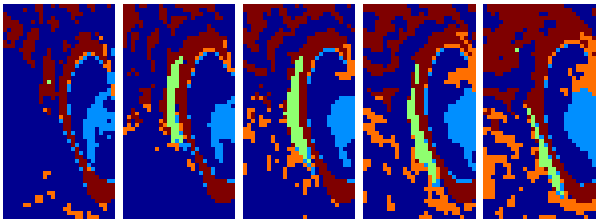
[3] D. Cremers et al. Motion competition: A variational framework for piecewise parametric motion segmentation. *IJCV*, 62:249–265, 2005.

[4] M. P. do Carmo. *Riemannian Geometry*. Birkhäuser, 1992.

(a) Ellipsoid visualization of the tensors in five sagittal slices



(b) Segmentation results of the five sagittal slices using LLE



(c) Segmentation results of the five sagittal slices using LE

Figure 4. Segmenting the corpus callosum and the cingulum. The first row shows the visualization of the data in five sagittal slices and the tensor at each voxel is represented by an ellipsoid. The second row shows the clustering result given by LLE. The third row shows the clustering result given by LE.

[5] D. Donoho et al. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *PNAS*, 100:5591–5596, 2003.

[6] R. Duda et al. *Pattern Classification*. Wiley, 2000.

[7] P. T. Fletcher et al. Riemannian geometry for the statistical analysis of diffusion tensor data. *Signal Processing*, 87(2):250–262, 2007.

[8] A. Goh and R. Vidal. Segmenting motions of different types by unsupervised manifold clustering. In *CVPR*, 2007.

[9] G. Kindlmann et al. Geodesic-loxodromes for diffusion tensor interpolation and difference measurement. In *MICCAI*, pg 1–9, 2007.

[10] J. Melonakos et al. Finsler tractography for white matter connectivity analysis of the cingulum bundle. In *MICCAI*, pg 36–43, 2007.

[11] X. Pennec, P. Fillard, and N. Ayache. A Riemannian framework for tensor computing. *IJCV*, 66(1):41–46, 2006.

[12] M. Polito and P. Perona. Grouping and dimensionality reduction by locally linear embedding. In *NIPS*, pg 1255-1262, 2002.

[13] Y. Rathi et al. Segmenting images on the tensor manifold. In *CVPR*, 2007.

[14] S. Roweis and L. Saul. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *JMLR*, 4:119–155, 2003.

[15] R. Souvenir and R. Pless. Manifold clustering. In *ICCV*, 2005.

[16] R. Subbarao and P. Meer. Nonlinear mean shift for clustering over analytic manifolds. In *CVPR*, pg 1168–1175, 2006.

[17] J. B. Tenenbaum et al. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[18] M. Tipping et al. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482, 1999.

[19] R. Vidal, Y. Ma, and S. Sastry. Generalized Principal Component Analysis (GPCA). *PAMI*, 27(12):1–15, 2005.

[20] Z. Wang et al. An affine invariant tensor dissimilarity measure and its applications to tensor-valued image segmentation. In *CVPR*, 2004.