

# Hierarchical Structuring of Data on Manifolds

Jun Li

Department of Computer Science  
Queen Mary, University of London  
London E1 4NS

junjy@dcs.qmul.ac.uk

Pengwei Hao

Department of Computer Science  
Queen Mary, University of London  
London E1 4NS

phao@dcs.qmul.ac.uk

## Abstract

*Manifold learning methods are promising data analysis tools. However, if we locate a new test sample on the manifold, we have to find its embedding by making use of the learned embedded representation of the training samples. This process often involves accessing considerable volume of data for large sample set. In this paper, an approach of selecting “landmark points” from the given samples is proposed for hierarchical structuring of data on manifolds. The selection is made such that if one use the Voronoi diagram generated by the landmark points in the ambient space to partition the embeded manifold, the topology of the manifold is preserved. The landmark points then are used to recursively construct a hierarchical structure of the data. Thus it can speed up queries in a manifold data set. It is a general framework that can fit any manifold learning algorithm as long as its result of an input can be predicted by the results of the neighbor inputs. Compared to the existing techniques of organizing data based on spatial partitioning, our method preserves the topology of the latent space of the data. Different from manifold learning algorithms that use landmark points to reduce complexity, our approach is designed for fast retrieval of samples. It may find its way in high dimensional data analysis such as indexing, clustering, and progressive compression. More importantly, it extends the manifold learning methods to applications in which they were previously considered to be not fast enough. Our algorithm is stable and fast, and its validity is proved mathematically.*

## 1. Introduction

Recently, many manifold learning algorithms have been designed for analyzing high dimensional data, which are samples of an intrinsically low dimensional manifold. However, most of the state-of-the-art manifold learning methods discover the embedding of the manifold rather than the explicit mapping. Thus although they demonstrate amazing

ability to find good low-dimensional parametrization for a set of manifold samples that are embedded in the high dimensional representation space, they are not able to give the parametrization for a new input sample. Generally, one needs to put the new sample and the training samples together and rerun the algorithm, which is often computationally expensive. Researchers have developed algorithms that can calculate the low dimensional parametrization for new samples with the help of the obtained parametrizations of its neighbors (2) (17). However, it is still cumbersome to find the nearest neighbors for every new sample, especially when there is a large number of samples, which is common in practice, because manifold learning needs dense sampling.

In this paper, the proposed approach is for hierarchically organizing data sampled from a manifold embedded in a high-dimensional space, so that they can be retrieved fast in response to a content-based query. This is particularly useful in the occasion that one needs to extend the resulting embedding from manifold learning algorithms.

Hierarchically organized data are helpful for the above data retrieval task, which is also of high interest in many research areas, such as data indexing, clustering, multimedia database management, and so on. The proposed method is natural, easy to implement and fast to run. Our method chooses a small subset of the samples as *landmark point*(LMP)s. Each sample is assigned to a nearby LMP. This can be done recursively to establish a hierarchical structure for the data.

Let  $\mathbf{x}$  denote a sample on manifold  $\mathbb{M}$ , and  $L_E(\mathbf{x})$  the closest LMP to  $\mathbf{x}$ . The LMP set is chosen such that the following condition is met:

**Condition 1.**  $\forall \mathbf{x} \in \mathbb{M}$ , connection between  $\mathbf{x}$  and  $L_E(\mathbf{x})$  is NOT a short circuit on  $\mathbb{M}$ .

This is to ensure that for an input test sample, it may find its neighborhood on manifold by finding its nearest LMP in Euclidean space. We formulate this condition in a section below.

Given a set of LMPs, their corresponding Voronoi dia-

gram partitions the ambient space. Intersecting with manifold  $\mathbb{M}$ , the Voronoi diagram also splits  $\mathbb{M}$ . We choose LMP such that each cell in the Voronoi diagram intersects  $\mathbb{M}$  in a local patch, i.e. it does not short-cut  $\mathbb{M}$ .

After reviewing related work in Section 2, we detail the method of finding LMPs and present the proof of its validity in Section 3. In Section 4, the problem of using LMPs to construct hierarchical data structure is discussed. In Section 5, we report some experimental results on widely used manifold learning data sets, a synthetic face data set (11) and a practical face database (8). Finally, we conclude the paper in Section 6.

## 2. Related Work

In the past few years, many manifold learning approaches emerged such as Isomap (18), local linear embedding (LLE) (16), Laplacian (1) and Hessian (6) eigenmaps, local tangent space alignment (LTSA) (23) and Riemannian manifold learning (14). Because of their potential to extract information from high dimensional raw data (7) (10), they have found their way to many applications (21), (3), (9). However, there are still many problems to be addressed before these approaches become practical tools for data analysis (20). One of the biggest barriers against manifold learning methods to be applied on practical tasks is that they do not yield explicit mapping between the latent space and the data space. Obviously, recomputing the embedding of a rich data set for a few test samples is computationally demanding. Some efforts have been made on this topic. The embedding generated by LLE can be used to embed new test points (22). Bengio et al. proposed a more general approach to extending the embedding found by Isomap, LLE and other eigenmaps for new samples (2), and they also provided the theoretical analysis. However, these extensions still rely on learned embedding of the training data.

There is rich literature in the field of data clustering and indexing. Those methods closely related to ours are a family of tree-based data structuring schemes, referred to as spatial accessing methods (SAM) (12) (19) (and references therein). Nevertheless, their objective is to organize the data in a balanced tree for fast retrieval. While we preserve the topology of the manifold, from which the observed data are sampled, our method is more suitable for dealing with data used by manifold learning algorithms. A figure below (Figure 2) shows a case where the query of neighbors for a newly incoming sample (the red triangle) does not return reasonable results using a traditional SAM.

The term ‘‘landmark point’’ in our paper is borrowed from techniques of manifold learning. However, they use LMPs to speed up or make it applicable on a large volume of data (4) (5) (13) (15), while we are concerned for neighborhood retrieval, our focuses are different.

## 3. Selection of Landmark Points

In this and the next section, we address the problem of structuring the data: Given observed samples, how to choose an appropriate set of landmark points from them. The proposed solution is able to be applied recursively in order to construct a hierarchical structure of the data. In the next section, we describe how to use the constructed structure to find neighbors on manifold for an incoming sample.

In the training (manifold learning) stage, the observed samples are the only information available about the unknown manifold. Therefore, LMPs are chosen from the sample set and we test our framework with those samples as well. In the following, we formulate Condition 1 given in Section 1, and give a method to choose LMPs to meet the condition.

First, we assume that we have had an initial set of LMPs. Then Condition 1 is examined by testing every sample on the manifold. If it is not met, a new LMP is added, and the procedure repeats.

### 3.1. Empirical Risk Detection

The symbols used in the following discussion are given as follows.  $\mathbf{X} := \{\mathbf{x}_i\}$ ,  $i = 1, \dots, N$ , is the set of the observed samples from a  $d$ -dimensional manifold  $\mathbb{M}$ , which is embedded in  $\mathbb{R}^D$ .  $\mathbf{S}_L \subset \mathbf{X}$  represents the current set of LMPs.  $d_E(\cdot, \cdot)$  stands for the Euclidean distance function in  $\mathbb{R}^D$ .  $d_M(\cdot, \cdot)$  measures the geodesic distance between two points on manifold. Given  $\mathbf{x} \in \mathbb{M}$ , denote the  $n$ -th nearest LMP to  $\mathbf{x}$  in the sense of Euclidean distance  $d_E$  as  $L_E^n(\mathbf{x}; \mathbf{S}_L)$  (‘‘E’’ for ‘‘Euclidean’’), and the one in the sense of geodesic distance on  $\mathbb{M}$  as  $L_M^n(\mathbf{x}; \mathbf{S}_L)$  (‘‘M’’ for ‘‘Manifold’’). In the following, we write  $L_E^1$  as  $L_E$  and  $L_M^1$  as  $L_M$ , and without writing  $\mathbf{S}_L$  explicitly if there is no ambiguity in context. We denote the set  $\{\mathbf{x} \in \mathbf{X} | L_E(\mathbf{x}) = \mathbf{s}\}$  as  $\mathbf{C}_s$ , or  $\mathbf{C}_{L_E(\mathbf{y})}$ , where  $\mathbf{y}$  is an arbitrary element in the set.

To be clear, we discuss using notions from the theory of Voronoi diagrams.  $\mathbf{S}_L$  partitions  $\mathbb{R}^D$  with its Voronoi diagram. Intersecting with the diagram,  $\mathbb{M}$  is also split, which is embedded in  $\mathbb{R}^D$ . We denote this Euclidean partition of the manifold as  $\mathcal{P}_E$ .

A *geodesic Voronoi diagram* partitions a manifold in a way analogous to a normal Voronoi diagram partitions an Euclidean space. The difference is that the distance between two points is defined as the length of the geodesic along the manifold, because all the *sites* (LMPs) are on the manifold, and the manifold is connected. They have a corresponding *geodesic Voronoi diagram* on the manifold. We denote the partition of the manifold with its geodesic Voronoi diagram as  $\mathcal{P}_M$ .

Generally speaking, one can test Condition 1 by verifying if  $\mathcal{P}_E$  is identical to  $\mathcal{P}_M$ , for  $\forall \mathbf{x}_t \in \mathbb{M}$ . However, it is not feasible in practice, because:

- The ambient space is of very high dimensions. Con-

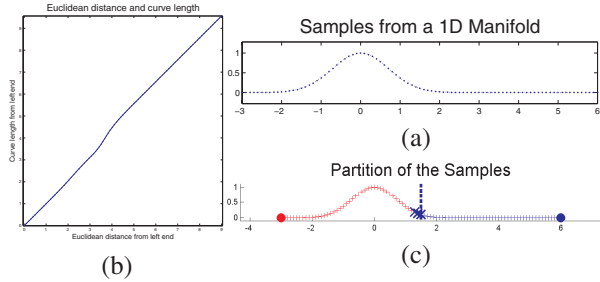
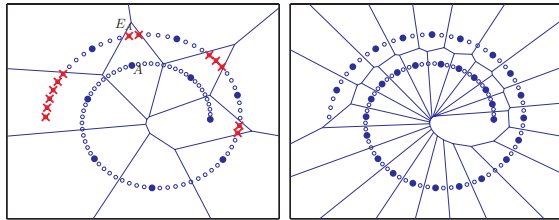


Figure 1. Border Error

- (a) samples from a 1D manifold; (b) distance metric fluctuation; (c) partition  $\mathcal{P}_E$  (dotted line),  $\mathcal{P}_M$  (red and blue colours) and border error (blue crosses)



(a) with TEPs (red crosses) (b) TEPs eliminated  
Figure 2. Partition of Manifold with LMPs (big dots)

structuring its Voronoi diagram is prohibitively expensive.

- The manifold is unknown. Thus it is impossible to test every point on it.
- It is also impossible to compute the geodesic between two points on the unknown manifold.

Therefore, we do not examine  $\mathcal{P}_E$  and  $\mathcal{P}_M$  everywhere on the manifold. Instead, we evaluate the *empirical risk* caused by  $\mathbf{S}_L$ . We take  $\mathcal{P}_E$  and  $\mathcal{P}_M$  on  $\mathbf{X}$  as labelling each  $\mathbf{x} \in \mathbf{X}$  with the “nearest” LMP. Moreover, in the case of  $\mathcal{P}_M$ , the geodesic length is replaced with the length of the shortest path in the neighborhood graph as in (18). In the following discussion, we do not distinguish the geodesic distance on the manifold and that defined by the shortest path. We keep using the notation  $L_M$  as well, where  $M$  denotes “manifold”.

### 3.2. Inconsistent points

“*Inconsistent points*” for  $\mathbf{S}_L$  refer to a set of points  $\{\mathbf{x} \in \mathbf{X} | L_E(\mathbf{x}; \mathbf{S}_L) \neq L_M(\mathbf{x}; \mathbf{S}_L)\}$ . If there are no inconsistent points, Condition 1 is met. Thus an inconsistent point indicates a possible failure to meet the condition. Careful study indicates that there are different kinds of inconsistent points, because the causes that make them inconsistent are different.

**Border error points** Consider a case that  $\mathcal{P}_E$  and  $\mathcal{P}_M$  are similar but not identical. Let  $\mathbf{C}_x = \{\mathbf{x}, \mathbf{x}_1, \dots, \mathbf{x}_{N_1}\}$

and  $\mathbf{C}_y = \{\mathbf{y}, \mathbf{y}_1, \dots, \mathbf{y}_{N_2}\}$  be two adjacent cells in  $\mathcal{P}_E$ , with  $\mathbf{x}$  and  $\mathbf{y}$  their respective LMPs.

For  $\mathbf{x}_0 \in \mathbf{C}_x$ , we have  $d_E(\mathbf{x}_0, \mathbf{y}) > d_E(\mathbf{x}_0, \mathbf{x})$ . However, because  $\mathbb{M}$  is nonlinear,  $d_M$  fluctuates from  $d_E$ , if  $\mathbf{x}_0$  is close to the border between  $\mathbf{C}_x$  and  $\mathbf{C}_y$ , it is possible that  $d_M(\mathbf{x}_0, \mathbf{y}) < d_M(\mathbf{x}_0, \mathbf{x})$ . In this case,  $\mathbf{x} = L_E(\mathbf{x}_0) \neq L_M(\mathbf{x}_0) = \mathbf{y}$ . It is the fluctuation of the distance metrics that makes  $\mathbf{x}_0$  an *inconsistent point*.

Figure 1 shows an example of this phenomenon. Figure 1(a) is a segment of a 1D manifold embedded in  $\mathbb{R}^2$ . Figure 1(b) shows the fluctuation between the two distance metrics by plotting curve length (vertical) against Euclidean distance, both being measured between samples on the curve and the left LMP (red dot in (c)). Figure 1(c) shows the *inconsistent points* in this case. Such inconsistent points do not affect greatly the structuring of the data. In our framework, they do not need special treatment.

**Topological error points** Another kind of inconsistent points are the *topological error points* (TEP). A TEP  $\mathbf{x}$  is faraway from  $L_E(\mathbf{x})$  on  $\mathbb{M}$ , which indicates topological inconsistency between  $\mathcal{P}_E$  and  $\mathcal{P}_M$ . For a structuring of  $\mathbb{M}$  with TEPs, there are some  $\mathbf{s} \in \mathbf{S}_L$ , where  $\mathbf{C}_s$  contains samples from distant areas of  $\mathbb{M}$ . If an input test sample arise from one of those areas, its neighbors may be erroneously found.

Figure 2 shows the topological error. When the Voronoi diagram of the LMPs are drawn, it splits the sample set. Topological error points and the mis-partitioning can be seen in the figure ( $A$  and  $E_A$ ).

To test whether  $\mathbf{x}$  is a TEP, it needs to be judged whether connection between  $\mathbf{x}$  and  $L_E(\mathbf{x})$  makes a short circuit on  $\mathbb{M}$ . However, it is often difficult to judge in practice, we should err on the safe side, if this happens.

Given a test sample  $\mathbf{x}$ , compare its nearest LMPs in the Euclidean space and those on the manifold. We find  $n$  such that  $L_E^1(\mathbf{x}) = L_M^n(\mathbf{x})$ . Suppose that  $\mathbb{M}$  has the intrinsic dimension  $d$ . For a “border error” inconsistent point  $\mathbf{x}$ , it is possible that  $\mathbf{x}$  lies in the vicinity of the point, which is equidistant from  $L_M^1(\mathbf{x}), \dots, L_M^{d+1}(\mathbf{x})$  as measured along the manifold. In this case,  $n \leq d + 1$ . Thus if  $n > d + 1$ ,  $\mathbf{x}$  is labelled as a TEP.

Figure 3 is plotted to demonstrate the validity of this detection. This figure shows TEP detection on 2000 samples of a swiss roll (18). There are 50 LMPs  $\{\mathbf{s}_i\}, i = 1, \dots, 50$  randomly selected from those samples.

Each subplot shows samples in a sample subset  $\mathbf{C}_{\mathbf{s}_i}, i = 1, \dots, 50$ . The curve in a subplot indicates the distances from the points in  $\mathbf{C}_{\mathbf{s}_i}$  to  $\mathbf{s}_i, d_M(\cdot, \mathbf{s}_i)$ , in ascending order. A red curve indicates that one or more points in its  $\mathbf{C}_{\mathbf{s}_i}$  is reported as TEP(s). Note that, our TEP-detection does not take geometric distance information. The sudden and significant increase of the value of  $d_M(\cdot, \mathbf{s}_i)$  coincides with the positive result of the TEP-detection, which shows that our criterion is effective to detect the empirical risk of breaking

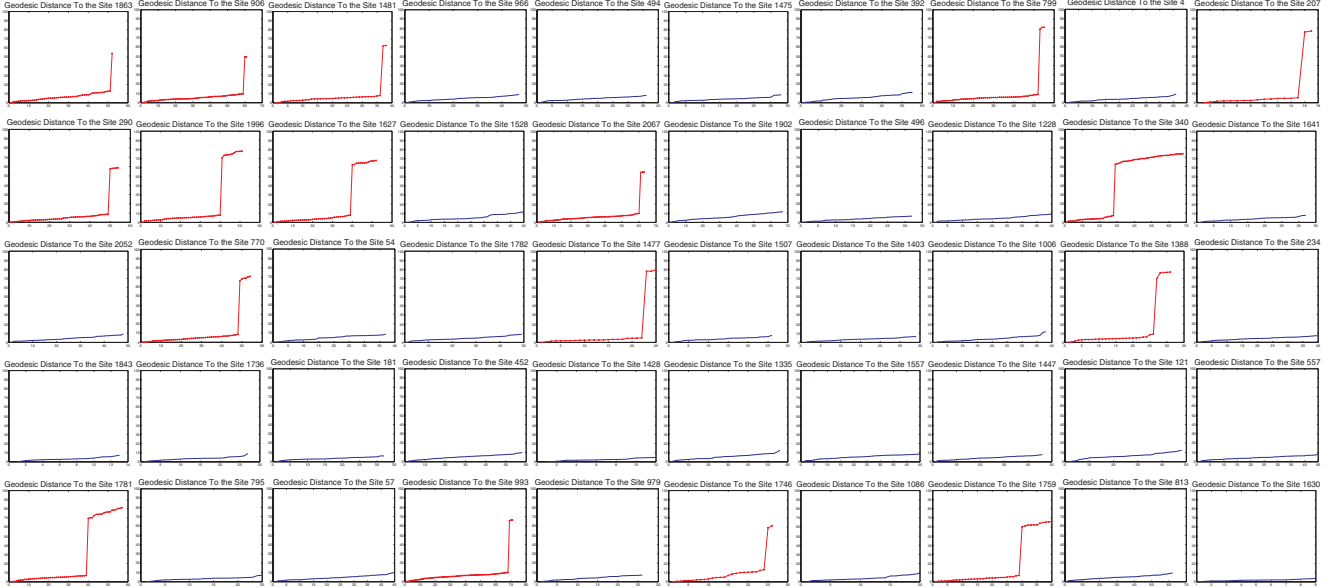


Figure 3. Geodesic distance from points in each cell to the site for a partition

manifold topology for a chosen set of LMPs.

### 3.3. Elimination of Topological Error Points

To eliminate the TEPs, we add new LMPs, because more LMPs give a finer partition. A simple solution is to choose a new LMP among those TEPs. Our solution is done in two steps. In the first step, we cluster TEPs according to their  $L_E$ . The second step is to choose a point in the biggest cluster as the new LMP by picking up the one that is closest to the Euclidean center of the cluster.

If  $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_{K_i}$  are detected to be TEPs, they have the same nearest LMP,  $s_Y = L_E(\mathbf{y}_1) = \dots = L_E(\mathbf{y}_{K_i})$ . Therefore, we choose one point from  $\mathbf{y}_1, \dots, \mathbf{y}_{K_i}$  as the new LMP. This splits the current  $\mathbf{C}_{s_Y}$ , and thus makes the partition finer. The computation is as follows.

Firstly, the center of those points is found

$$\bar{\mathbf{y}} = \frac{1}{K_i} \sum_{j=1}^{K_i} \mathbf{y}_j \quad (1)$$

Then the point  $\mathbf{y}_c$  that is nearest to  $\bar{\mathbf{y}}$  is found as the new LMP, such that,

$$d_E(\mathbf{y}_c, \bar{\mathbf{y}}) = \min_{j=1, \dots, K_i} d_E(\mathbf{y}_j, \bar{\mathbf{y}}) \quad (2)$$

At last, the LMP set is updated,

$$\mathbf{S}_L^k = \{\mathbf{y}_c\} \cup \mathbf{S}_L^{k-1}, k = 1, 2, \dots \quad (3)$$

### 3.4. Proof of Convergence

The algorithm converges definitely. Because an LMP can not be a TEP itself, there must be a trivial solution that

every point is an LMP. However, a more helpful LMP set is generated when the algorithm terminated before this trivial end. It is cumbersome to consider all possibilities of the training set of the manifold  $\mathbb{M}$  and the starting LMP set  $\mathbf{S}_L^0$  for the algorithm. We present a loose proof below explaining why the algorithm works in practice.

Let  $\mathbf{S}_L^k$  be the current LMP set. Denote the maximum radius of the cells in the corresponding Voronoi diagram as

$$D(\mathbf{S}_L^k; \mathbb{M}) = \max_{\mathbf{x} \in \mathbb{M}} d_E(\mathbf{x}, L_E(\mathbf{x}; \mathbf{S}_L^k)) \quad (4)$$

Because the manifold  $\mathbb{M}$  is constant, we denote the equation as  $D(\mathbf{S}_L^k)$ . Suppose there is a function that judges whether connecting two points on a manifold is topologically safe, i.e. connection between them does not cause short circuit on  $\mathbb{M}$ ,

$$f(\mathbf{x}_1, \mathbf{x}_2; \mathbb{M}) = \begin{cases} 0 & \text{if } (\mathbf{x}_1, \mathbf{x}_2) \text{ causes short circuit} \\ 1 & \text{if } (\mathbf{x}_1, \mathbf{x}_2) \text{ is topologically safe} \end{cases} \quad (5)$$

Let

$$d_{TS}(\mathbb{M}; f) = \max_{(\mathbf{x}_1, \mathbf{x}_2) \in \mathbf{E}_{TS}} d_E(\mathbf{x}_1, \mathbf{x}_2) \quad (6)$$

where  $d_{TS}$  is the ‘‘topological safe’’ connection set,

$$\mathbf{E}_{TS} := \{(\mathbf{x}_1, \mathbf{x}_2) | \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{M}, f(\mathbf{x}_1, \mathbf{x}_2; \mathbb{M}) = 1\} \quad (7)$$

For a manifold  $\mathbb{M}$  and an appropriately defined  $f$ ,  $d_{TS}$  is a positive constant, which means the ‘‘longest topologically safe’’ connection between an arbitrary pair of points on  $\mathbb{M}$ . In the  $k$ -th iteration, one adds a new LMP into  $\mathbf{S}_L^{k-1}$  to form  $\mathbf{S}_L^k$ , thus  $\mathbf{S}_L^0 \subset \mathbf{S}_L^1 \subset \dots$ . According to the definition in Eq(4), for  $\mathbf{S}_L^{k_1} \subset \mathbf{S}_L^{k_2}$ , we have  $D(\mathbf{S}_L^{k_1}) \geq D(\mathbf{S}_L^{k_2})$ . Therefore  $D(\mathbf{S}_L^0) \geq D(\mathbf{S}_L^1) \geq \dots$ , and when  $\mathbf{S}_L^k$  approaches

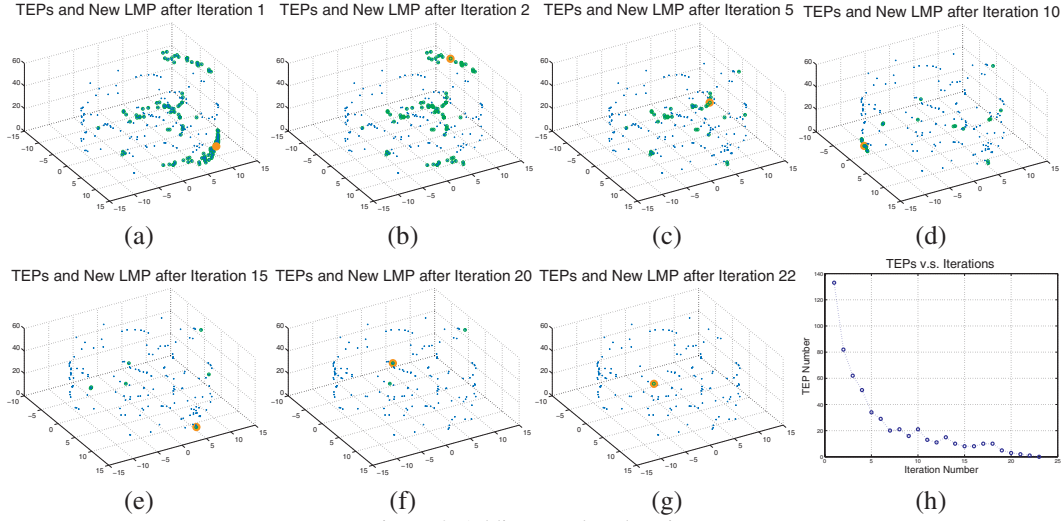


Figure 4. Adding Landmark Points

TEPs are green dots; new LMP is a bigger orange dot. (a) - (g) TEPs and the new LMPs after each iteration. (h) number of TEPs versus the iterations.

$\mathbf{X}$ ,  $D(\mathbf{S}_L^k)$  reaches 0. Thus when  $D(\mathbf{S}_L^k) < d_{TS}(\mathbb{M})$ , the algorithm exits with success. If we substitute a finite collection of samples of the manifold for  $\mathbb{M}$  itself, the statements above also hold. So our method works fine in practice.

### 3.5. Implementation and Complexity

In implementation, when a new LMP  $\mathbf{s}_{New}$  is acquired in the  $k$ -th iteration, the new set of LMP is  $\mathbf{S}_L^k = \{\mathbf{s}_{New}\} \cup \mathbf{S}_L^{k-1}$ . It is needed to compute  $L_E(\mathbf{x}; \mathbf{S}_L^k)$  and  $L_M^j(\mathbf{x}; \mathbf{S}_L^k)$ ,  $j = 1, \dots, d + 1$  for  $\mathbf{x} \in \mathbf{X}$ . They can be updated from  $L_E(\mathbf{x}; \mathbf{S}_L^{k-1})$  and  $L_M^j(\mathbf{x}; \mathbf{S}_L^{k-1})$  respectively.

To compute  $L_E(\mathbf{x}; \mathbf{S}_L^k)$ ,

- 1: **for**  $\mathbf{x} \in \mathbf{X}$  **do**
- 2:   **if**  $d_E(\mathbf{x}, \mathbf{s}_{New}) < d_E(\mathbf{x}, L_E(\mathbf{x}, \mathbf{S}_L^{k-1}))$  **then**
- 3:      $L_E(\mathbf{x}, \mathbf{S}_L^k) \leftarrow \mathbf{s}_{New}$
- 4:   **else**
- 5:      $L_E(\mathbf{x}, \mathbf{S}_L^k) \leftarrow L_E(\mathbf{x}, \mathbf{S}_L^{k-1})$
- 6:   **end if**
- 7: **end for**

This process has computational complexity of  $\mathcal{O}(N)$ , where  $N$  is the number of the samples in  $\mathbf{X}$ .

To compute  $L_M^j(\mathbf{x}; \mathbf{S}_L^k)$ ,  $j = 1, \dots, d + 1$ , let  $\Delta^{k-1} = \max_{j, \mathbf{x}} L_M^j(\mathbf{x}; \mathbf{S}_L^{k-1})$

- 1: Run Dijkstra algorithm to Compute  $d_M(\mathbf{x}, \mathbf{s}_{New})$
- 2: **for**  $\mathbf{x} \in \mathbf{X}$ , visited by Dijkstra algorithm **do**
- 3:   Update  $L_M^j(\mathbf{x})$
- 4:   **if**  $d_M(\mathbf{x}, \mathbf{s}_{New}) > \Delta^{k-1}$  **then**
- 5:     Terminate Dijkstra algorithm
- 6:   **end if**
- 7: **end for**
- 8: Update  $\Delta^k$  during the process

Computing  $d_M(\mathbf{x}, \mathbf{s}_{New})$  involves calculating the shortest path to  $\mathbf{s}_{New}$ . By Dijkstra algorithm with Fibonacci heap,

this takes  $\mathcal{O}(nN \log N)$  for  $\mathbf{X}$ , where  $n$  is the neighbors of a point on the neighborhood graph  $G$ . This is in practice  $\mathcal{O}(nN^- \log N^-)$ , where  $N^-$  is the number of samples whose shortest path to  $\mathbf{s}_{New}$  is calculated before the algorithm terminates. For uniformly sampled data and random  $\mathbf{S}_L^{k-1}$ ,  $N^-$  is proportional to  $N/N_L^{k-1}$ , where  $N_L^{k-1}$  is the number of LMPs in  $\mathbf{S}_L^{k-1}$ .

The initialization of  $L_E(\mathbf{x}; \mathbf{S}_L^0)$  takes  $\mathcal{O}(N_L^0 N)$ , and that of  $L_M^j(\mathbf{x}; \mathbf{S}_L^0)$ ,  $j = 1, \dots, d + 1$  takes  $\mathcal{O}(nN_L^0 N \log N)$ .

## 4. Hierarchical Data Structuring and Retrieval

With LMPs chosen for a data set, the next problem concerned is how to structure the data with them and to use the structuring for applications such as retrieval.

### 4.1. Retrieval

The solution to retrieval sets the objective for the structuring, and is straightforward. Given a test sample  $\mathbf{x}_t$ , firstly we find the nearest landmark by searching for the least Euclidean distance from the sample to the LMPs,  $L_E(\mathbf{x}_t)$ . Then we search the subset of data with  $L_E(\mathbf{x}_t)$ . Note that the geodesic distance measurement is not accessible in this case, thus  $L_M(\mathbf{x}_t)$  is not obtainable.

### 4.2. Structuring

The intuitive way to arrange the data with the LMPs is to establish association between individual sample  $\mathbf{x}_i$  with its nearest LMP  $L_E(\mathbf{x}_i)$ . The problem with this simple structuring is that: If a test sample  $\mathbf{x}_t$  is near the ‘‘border area’’ as described in Sec 3.2, it is possible that its neighborhood is only partially in the subset of data associated and  $L_E(\mathbf{x}_t)$ . In Figure 5(a), a cell illustrates the data subset with each LMP. The two test samples  $\mathbf{x}_A$  and  $\mathbf{x}_B$  are labelled as ‘‘A’’

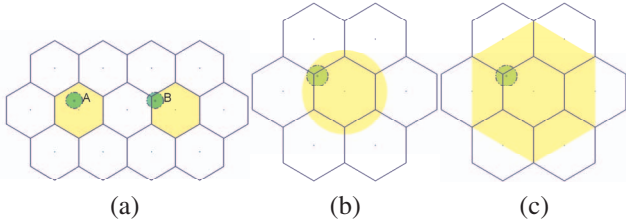


Figure 5. Association between Manifold Samples and LMPs (a) with only  $L_E(\mathbf{x})$ ; (b) ideal association; (c) association used



Figure 6. Manifold Samples, Partitions (colour), and LMPs (big dot)

and “B” in the figure.  $\mathbf{x}_A$  lies in the interior area of a subset, and thus its neighborhood is fully retrievable. But for the other sample  $\mathbf{x}_B$ , only some of its neighbor points can be found.

A solution to this problem is to “expand” the subset associated with each LMP, such that, for a sample that is near the border of the cell, it can find its neighbors as well. The expansion is done according to the size of the neighborhood that is possibly required. Figure 5(b) shows this solution. With the extra associated data, the neighborhood of  $\mathbf{x}_B$  can be found at the cost of slightly increased computational complexity. However, in the construction stage, it is not easy to decide whether a sample should be associated to an LMP such that the set of associated samples of the LMP is properly expanded.

We use a simpler solution in our implementation. Each sample  $\mathbf{x} \in \mathbf{X}$  is associated with its nearest LMPs on  $\mathbb{M}$ ,  $L_M^1(\mathbf{x}), \dots, L_M^k(\mathbf{x}), k = d + 1$ , where  $d$  is the dimension of  $\mathbb{M}$ . Figure 5(c) displays the association of samples and LMPs for this approach. With association expanded, the searching overhead is higher, but the implementation is straightforward.

## 5. Experiments

Data Set	Average Iterations	Max. Iterations
Swiss Roll	23.6	33
Swiss Hole	6.5	18
Helix	4.8	8
Faces	34.9	52

Table 1. Iteration Times on Data Sets

This section includes several experimental results to verify the proposed structuring algorithm. Three data sets of

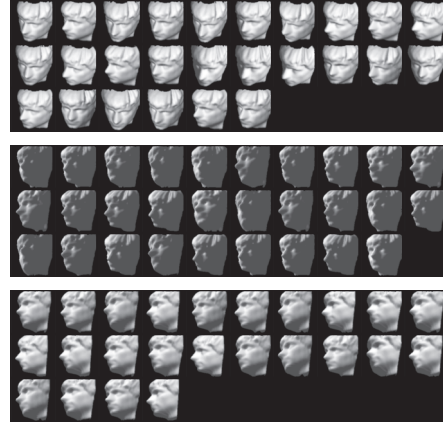


Figure 7. Face Model Images  
The first image in each group is the LMP.

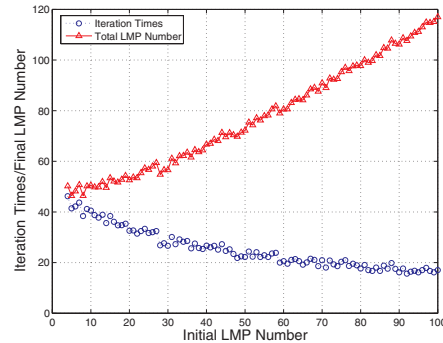


Figure 8. Iteration Times versus Initial Number of LMPs

3D point clouds and a set of synthetic face images are used for our experiments. The point clouds are samples from different manifolds embedded in the 3D Euclidean space. Each of them contains 2000 samples. They are displayed in Figure 6. The face images are from (11). Figure 7 shows some of the samples organized in the way that an LMP is displayed with its associated samples. The organization is generated by the proposed algorithm.

The first experiment is to test the effectiveness of the proposed algorithm. It is run on each data set for 100 times. At each time it is given a randomly selected set of initial LMPs; and the number of iterations is recorded. The number of initial LMPs is set to 50 for the manifolds of 3D space, and 20 for the face images. The maximum and the average number of iterations used during the 100 runnings for each data set are listed in Table 1. It shows that the algorithm does converge in practice with arbitrarily chosen initial LMPs. The iteration number loosely reflects the complexity of the manifold.

In the second experiment, the algorithm is tested under different initial LMP sets with various sizes. It is conducted on the Swiss Roll data set. The number of initial LMPs  $n$  is from 4 to 100. For each size  $n$ , 10 different initial sets of LMPs are chosen, denoted as  $S_{L_0}^{n,i}$ , where

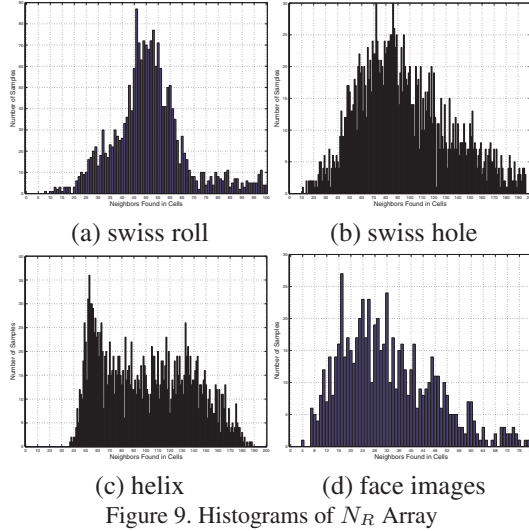


Figure 9. Histograms of  $N_R$  Array

$n = 4, \dots, 100$ ;  $i = 1, \dots, 10$ . The number of iterations is recorded as  $N_I(n, i)$ . The average iteration number for each  $n$ ,  $\bar{N}_I(n) = 1/10 \sum_{i=1}^{10} N_I(n, i)$  is displayed in Figure 8, as well as the total number of the LMPs generated,  $\bar{N}_I(n) + n$  (Note the total number of LMPs generated in one individual running is  $N_I(n, i) + n$ , for some  $i$ ). It can be observed that the more initial LMPs are provided, the less iterations are needed. However, the total number of the resulting LMPs increases, indicating finer partitions.

The following experiment consists of practical tests of neighborhood retrieval by searching the nearest samples using the obtained structure on each data set. Let  $\mathbf{P}(\mathbf{s})$  denote the training samples associated with LMP  $\mathbf{s}$  (different from  $\mathbf{C}_s$  in structuring stage), and  $\eta_k(\mathbf{x}; \mathbf{G})$  denote the  $k$ -th nearest point to  $\mathbf{x}$  in  $\mathbf{G} \subset \mathbb{X}$  (different from  $L_E(\cdot)$ , which refers to the nearest LMP), distance being measured in Euclidean space. The test is to evaluate the *empirical risk of retrieval* for a structuring of the data. At each time,  $\mathbf{x} \in \mathbf{X}$  is taken as the test sample.  $\eta_k(\mathbf{x}; \mathbf{X})$  and  $\eta_k(\mathbf{x}; \mathbf{P}(L_E(\mathbf{x})))$  are compared for  $k = 1, 2, \dots$ , until for some  $k$  they are different. Then  $k$  is recorded as  $N_R(\mathbf{x})$ . Figure 9 shows the histograms of  $\{N_R(\mathbf{x}) | \mathbf{x} \in \mathbf{X}\}$  for the data sets. The configuration in this experiment is the same as our first one.

In fact, because the structure has topology-keeping nature by construction, when  $\eta(\mathbf{x}; \mathbf{X}) \neq \eta(\mathbf{x}; \mathbf{P}(L_E(\mathbf{x})))$ , the neighbors retrieved with the structure are more reliable than those directly from the original data set. An experiment is done: For every  $\mathbf{x} \in \mathbf{X}$ ,  $\eta_k(\mathbf{x}; \mathbf{X})$  and  $\eta_k(\mathbf{x}; \mathbf{P}(L_E(\mathbf{x})))$  are found for some  $k$ . The *geodesic distances* are then computed from  $\mathbf{x}$  to both  $\eta_k(\mathbf{x}, \cdot)$ ,

$$\delta_k^{\mathbf{X}}(\mathbf{x}) = d_M(\mathbf{x}, \eta_k(\mathbf{x}; \mathbf{X})) \quad (8)$$

$$\delta_k^{\mathbf{P}(L_E(\mathbf{x}))}(\mathbf{x}) = d_M(\mathbf{x}, \eta_k(\mathbf{x}; \mathbf{P}(L_E(\mathbf{x})))) \quad (9)$$

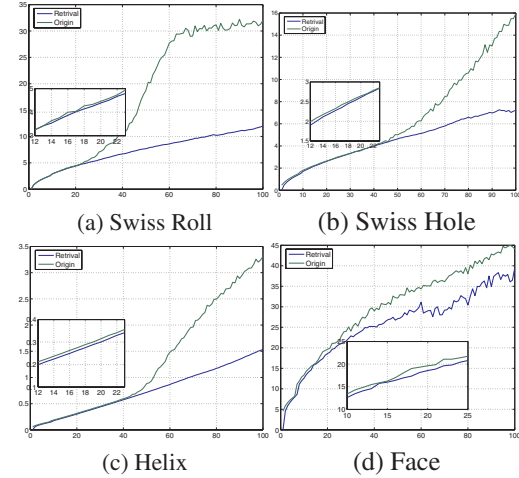


Figure 10. Geodesic Distances to Retrieved Neighbors

Then the averages are computed,

$$\bar{\delta}_k^{\mathbf{X}} = \frac{1}{N} \sum_{i=1}^N \delta_k^{\mathbf{X}}(\mathbf{x}_i) \quad (10)$$

$$\bar{\delta}_k^{\mathbf{P}} = \frac{1}{N} \sum_{i=1}^N \delta_k^{\mathbf{P}(L_E(\mathbf{x}_i))}(\mathbf{x}_i) \quad (11)$$

They are compared in Figure 10 for  $k = 1, \dots, 100$  for each data set. Note that, if  $\eta_k^{\mathbf{P}(L_E(\mathbf{x}))}$  is not defined for some  $\mathbf{x}$ , (when  $\mathbf{P}(L_E(\mathbf{x}))$  contains less than  $k$  elements), it is simply removed from the average in Eq(11).

The algorithm is also tested on practical data. We take images from YaleB face database (8), collapse them into image vectors, and construct a hierarchical structure using our algorithm. The resulting figure is shown in Figure 11. In that figure, (a) is drawn in a way that: the first image  $I_S$  is an LMP; the others are samples in  $\mathbf{C}_{I_S}$ . (b) shows the same LMP, with all samples associated to it. Note that some of the associated samples are far from the LMP by Euclidean distance, however, they share similar pose and illumination configurations, which are the major variant factors in the database. While (c) displays the parameter space of the data set, with LMPs drawn in dark blue and samples in (b) drawn in green.

An experiment of multilayer structuring is demonstrated in Figure 12. The algorithm takes a large data set of 50000 points sampled from the Swiss Roll. Two layers of LMPs are generated. First the algorithm yields 1102 LMPs for the original data set, being given 1000 randomly selected initial LMPs(Figure 12(a)). Taking the 1102 layer-1 LMPs as input, the algorithm generates 55 layer-2 LMPs for 50 layer-1 LMPs randomly selected as initial layer-2 LMPs. Figure 12(b) shows the LMPs, in which the bigger dots are layer-2 LMPs.

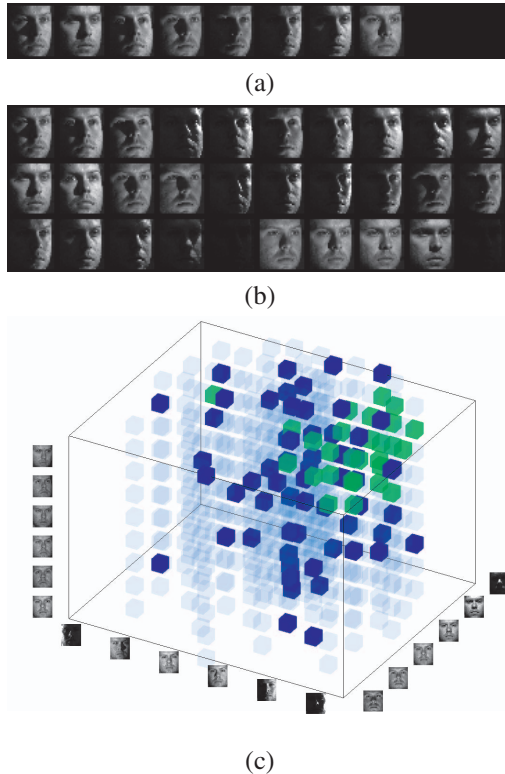


Figure 11. Structuring of YaleB Face Database  
 (a) LMP with samples in a Voronoi cell; (b) LMP with associated samples; (c) parameter space

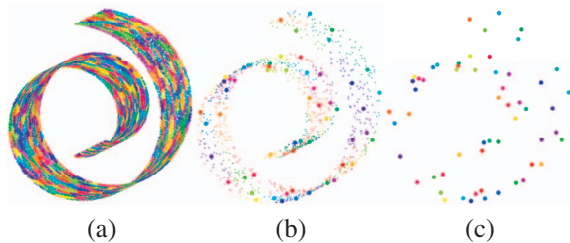


Figure 12. Hierarchical partition of 50000 points from swiss roll  
 (a) Data with 1102 layer-1 LMPs; (b) Layer-1 LMPs with 55 layer-2 LMPs; (c) Layer-2 LMPs

## 6. Conclusion

In this work, an algorithm to construct hierarchical representation of samples on manifold is proposed. It focuses on enabling searching for neighbors on manifold, which is a critical and time-consuming step in the current out-of-sample extensions of manifold learning algorithms. The generated LMPs can help locate a new nearby test sample on the manifold, so that its closest neighbors can be found. We have also proved that our algorithm terminates by producing a proper set of LMPs for a reasonably sampled data set on manifold. However, although every known sample is tested to guarantee that it can be located correctly with the LMPs, there still may be small unknown portions of the

manifold, on which samples may be mis-located. In future work, the test will be strengthened such that samples from everywhere on manifold as well as the vicinity of the manifold can be located correctly using the generated LMPs.

## References

- [1] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 2003.
- [2] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In *NIPS*, 2004.
- [3] H. Chang, D.-Y. Yeung, and Y. Xiong. Super-resolution through neighbor embedding. In *CVPR*, 2004.
- [4] V. de Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *NIPS*, 2002.
- [5] V. de Silva and J. B. Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, Stanford Mathematics, 2004.
- [6] D. L. Donoho and C. Grimes. Hessian eigenmaps: new locally linear embedding techniques for highdimensional data. In *PNAS*, 2003.
- [7] D. L. Donoho and C. Grimes. Image manifolds which are isometric to euclidean space. *J. Math. Imaging Vis.*, 2005.
- [8] A. Georghiadis, P. Belhumeur, and D. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(6):643–660, 2001.
- [9] X. He, S. Yan, Y. Hu, and P. Niyogi. Face recognition using laplacianfaces. *IEEE Trans. PAMI.*, 2005.
- [10] X. Huo and A. K. Smith. Performance analysis of a manifold learning algorithm in dimension reduction. Technical report, Statistics Group, Georgia Institute of Technology, 2006.
- [11] Synthetic face database. <http://isomap.stanford.edu/>.
- [12] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.
- [13] Ketprechasawat. Hierarchical landmark charting. Master's thesis, Brown University, 2006.
- [14] T. Lin, H. Zha, and S. U. Lee. Riemannian manifold learning for nonlinear dimensionality reduction. In *Proceedings of ECCV*, 2006.
- [15] J. Platt. Fastmap, metricmap, and landmark MDS are all nystrom algorithms. In *Proceedings of 10th International Workshop on Artificial Intelligence and Statistics*, 2005.
- [16] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 2000.
- [17] L. K. Saul and S. T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *J. Mach. Learn. Res.*, 2003.
- [18] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 2000.
- [19] P. van Oosterom. *Spatial Access Methods*, volume 1, chapter 2, pages 385–400. Wiley, 1999.
- [20] M.-C. Yeh, I.-H. Lee, G. Wu, Y. Wu, and E. Chang. Manifold learning, a promised land or work in progress? In *Proc. of IEEE Conf. on Multimedia and Expo*, 2005.
- [21] C. Zhang, J. Wang, N. Zhao, and D. Zhang. Reconstruction and analysis of multi-pose face images based on nonlinear dimensionality reduction. *Pattern Recognition*, 2004.
- [22] J. Zhang, H. Shen, and Z.-H. Zhou. Unified locally linear embedding and linear discriminant analysis algorithm (ul-lelda) for face recognition. In *SINOBIOMETRICS*, 2004.
- [23] Z. Zhang and H. Zha. Principal manifolds and nonlinear dimension reduction via local tangent space alignment. *SIAM J. Scientific Computing*, 2005.