An Explicit Nonlinear Mapping for Manifold Learning

Hong Qiao, *Senior Member, IEEE*, Peng Zhang, *Member, IEEE*, Di Wang, and Bo Zhang, *Member, IEEE*

***Abstract*—Manifold learning is a hot research topic in the field of computer science and has many applications in the real world. A main drawback of manifold learning methods is, however, that there are no explicit mappings from the input data manifold to the output embedding. This prohibits the application of manifold learning methods in many practical problems such as classification and target detection. Previously, in order to provide explicit mappings for manifold learning methods, many methods have been proposed to get an approximate explicit representation mapping with the assumption that there exists a linear projection between the high-dimensional data samples and their low-dimensional embedding. However, this linearity assumption may be too restrictive. In this paper, an explicit nonlinear mapping is proposed for manifold learning, based on the assumption that there exists a polynomial mapping between the high-dimensional data samples and their low-dimensional representations. As far as we know, this is the first time that an explicit nonlinear mapping for manifold learning is given. In particular, we apply this to the method of locally linear embedding and derive an explicit nonlinear manifold learning algorithm, which is named neighborhood preserving polynomial embedding. Experimental results on both synthetic and real-world data show that the proposed mapping is much more effective in preserving the local neighborhood information and the nonlinear geometry of the high-dimensional data samples than previous work.**

***Index Terms*—Data mining, machine learning, manifold learning, nonlinear dimensionality reduction (NDR).**

I. Introduction

MANIFOLD learning has drawn great interests since it was first proposed in 2000 (see [1], [2], and [4]) as a promising nonlinear dimensionality reduction (NDR) method for high-dimensional data manifolds. Its basic assumption is that high-dimensional input data samples lie on or close to a low-dimensional smooth manifold embedded in the ambient Euclidean space. For example, by rotating the camera around the same object with a fixed radius, images of the object can be viewed as a 1-D curve embedded in a high-dimensional Euclidean space, whose dimension is equal to the number of pixels in the image. With the manifold assumption, manifold learning methods aim to extract the intrinsic degrees of freedom underlying the input high-dimensional data samples, by preserving local or global geometric characteristics of the manifold from which data samples are drawn. In recent years, various manifold learning algorithms have been proposed, such as locally linear embedding (LLE) [2], [3], isometric mapping (ISOMAP) [4], [5], locally multidimensional scaling [6], maximum variance unfolding (MVU) [7], local tangent space alignment [8], Laplacian eigenmap (LE) [9], Riemannian manifold learning (RML) [10], diffusion maps [12], and Hessian eigenmap [13]. They have achieved great success in finding meaningful low-dimensional embeddings for high-dimensional data manifolds. Meanwhile, manifold learning also has many important applications in real-world problems, such as human motion detection [15], face recognition [16], classification and compressed expression of hyperspectral imageries [17], dynamic shape and appearance classification [18], and visual tracking [19]–[21].

However, a main drawback of the manifold learning methods is that they learn the low-dimensional representations of the high-dimensional input data samples implicitly. No explicit mapping relationship from the input data manifold to the output embedding can be obtained after the training process. Therefore, in order to obtain the low-dimensional representations of new coming samples, the learning procedure, containing all previous samples and new samples as inputs, has to be repeatedly implemented. It is obvious that such a strategy is extremely time consuming for sequentially arrived data, which greatly limits the application of manifold learning methods to many practical problems, such as classification, detection, and visual tracking.

In order to address the issue of lacking explicit mappings, many linear projection-based methods have been proposed for manifold learning by assuming that there exists a linear projection between the high-dimensional input data samples and their low-dimensional representations, such as locality preserving projections (LPP) [22], [23], neighborhood preserving embedding (NPE) [24], neighborhood preserving projections (NPP) [25], orthogonal locality preserving projections (OLPP) [26], orthogonal neighborhood preserving projections (ONPP) [27], and graph embedding [28]. Although these methods have

Manuscript received February 21, 2011; revised December 24, 2011; accepted April 15, 2012. Date of publication June 18, 2012; date of current version January 11, 2013. This work was supported in part by the National Natural Science Foundation of China under Grant 61033011, Grant 60725310, and Grant 11131006. This paper was recommended by Associate Editor N. Chawla.

H. Qiao is with the Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: hong.qiao@ia.ac.cn).

P. Zhang is with the Data Center, National Disaster Reduction Center of China, Beijing 100124, China (e-mail: zhangpeng@ndrcc.gov.cn).

D. Wang is with the Graduate School and the Institute of Applied Mathematics, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China (e-mail: wangdi@amss.ac.cn).

B. Zhang is with the State Key Lab of Scientific and Engineering Computing and the Institute of Applied Mathematics, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China (e-mail: b.zhang@amt.ac.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2012.2198916

achieved their success in many problems, the linearity assumption may still be too restrictive.

On the other hand, several kernel embedding methods have been also proposed to give nonlinear but implicit mappings for manifold learning (see, e.g., [29]–[32]). These methods use the kernel trick to reformulate the manifold learning methods as kernel learning problems and then utilize the existing kernel extrapolation (KE) techniques to find the location of new data samples in the low-dimensional space. The mappings provided by the kernel-based methods are nonlinear and implicit. In addition, these mappings are computed within a subset of the feature space rather than the whole feature space itself. Furthermore, these mappings are given in terms of the kernel and the training data samples explicitly; hence, their computational complexity would be extremely high for very large data sets.

The main purpose of this paper is to propose an explicit nonlinear mapping for manifold learning, which, compared with linear projection methods (such as LPP and ONPP) and kernel-based nonlinear mapping methods [such as kernel LPP (KLPP) and kernel ONPP (KONPP)], gives more accurate embedding and out-of-sample extension results and, meanwhile, is very fast in locating new coming samples. Hence, this explicit nonlinear mapping can be expected to be used for practical large-scale online learning problems such as online classification, detection, and visual tracking.

In this paper, based on the assumption that there exists a polynomial mapping between high- and low-dimensional representations, such an explicit nonlinear mapping is proposed for manifold learning for the first time. The proposed mapping has the following main features.

1) The mapping is explicit; hence, it is straightforward to locate any new data samples in the low-dimensional space. This is different from the traditional manifold learning methods such as LLE, LE, and ISOMAP [4], in which the mapping is implicit, and it is not clear how new data samples can be embedded in the low-dimensional space. Compared with mappings provided by the kernel embedding methods, the proposed mapping is not expressed in terms of the training data samples or the specific kernels and, therefore, is very fast and efficient in finding the low-dimensional representations of new data samples even for very large data sets. A detailed comparison between the proposed method and the kernel embedding methods is given in Section V.
2) The mapping is nonlinear. In contrast to the linear projection-based methods that find a linear projection mapping from the input high-dimensional samples to their low-dimensional representations, the proposed mapping provides a nonlinear polynomial mapping from the input space to the reduced space. Clearly, it is more reasonable to use a polynomial mapping to handle data samples lying on nonlinear manifolds. Meanwhile, our analysis and experiments show that the proposed mapping is of similar computational complexity with the linear projection-based methods.

Combining this explicit nonlinear mapping with existing manifold learning methods (e.g., LLE, LE, and Isomap) can give explicit manifold learning algorithms. In this paper, we concentrate on the LLE manifold learning method and propose an explicit nonlinear manifold learning algorithm called neighborhood preserving polynomial embedding (NPPE) algorithm. Experiments on both synthetic and real-world data have been conducted to illustrate the validity and effectiveness of the proposed mapping.

TABLE I
MAIN NOTATIONS

$\mathbb{R}^n$	n-dimensional Euclidean space where input samples lie
$\mathbb{R}^m$	m-dimensional Euclidean space, $m < n$, where the low-dimensional embedding lie
x_i	$x_i = (x_i^1, \cdots, x_i^n)^T$, the i-th input sample in $\mathbb{R}^n$, $i = 1, 2, \ldots, N$
$\mathcal{X}$	$\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$, the set of input samples
X	$X = [x_1\ x_2\ \cdots\ x_N]$, $n \times N$ matrix of input samples
y_i	$y_i = (y_i^1, \cdots, y_i^m)^T$, low-dimensional representation of x_i obtained by manifold learning, $i = 1, 2, \ldots, N$
$\mathcal{Y}$	$\mathcal{Y} = \{y_1, y_2, \ldots, y_N\}$, the set of low-dimensional representations
Y	$Y = [y_1\ y_2\ \cdots\ y_N]$, $m \times N$ matrix of low-dimensional representations
I_m	Identity matrix of size m
$\lVert \cdot \rVert_2$	L_2-norm where $\lVert v \rVert_2 = \sqrt{\sum_{k=1}^m (v^k)^2}$ for $v \in \mathbb{R}^m$

The remaining part of this paper is organized as follows. Section II gives a brief review of the existing mapping methods for manifold learning, including those based on linear projections and kernel-based nonlinear mappings. Details of the explicit nonlinear mapping for manifold learning are presented in Section III, whereas the NPPE algorithm is given in Section IV. In Section V, the difference between the proposed method and the kernel embedding methods is discussed in detail. In Section VI, experiments are conducted on both synthetic and real-world data sets to demonstrate the validity of the proposed algorithm. Conclusion and future work are given in Section VII.

II. Related Works

In this section, we briefly review existing mapping methods for manifold learning, including those based on linear projections and out-of-sample nonlinear extensions for learned manifolds.

For convenience of presentation, the main notations used in this paper are summarized in Table I. Throughout this paper, all data samples are in the form of column vectors. The superscript of a data vector is the index of its component.

A. Linear Projections for Manifold Learning

Manifold learning methods based on linear projections assume that there exists a linear projection that maps high-dimensional samples into a low-dimensional space, that is

$$y_i = U^T x_i, \quad \text{where } U \in \mathbb{R}^{n\times m} \tag{1}$$

where x_i is a high-dimensional sample, and y_i is its low-dimensional representation.

1) LPP: LPP [22], [23] provides a linear mapping for LE, by applying (1) into the training procedure of LE. The LE method aims to train a set of low-dimensional representations $\mathcal{Y}$ that can best preserve the adjacency relationship among high-dimensional inputs $\mathcal{X}$. This property is achieved by solving the following optimization problem:

$$\min \quad \sum_{i,j=1}^{N} W_{ij}\|y_i - y_j\|_2 \tag{2}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} D_i y_i y_i^T = I_m \tag{3}$$

where the penalty weights W_{ij} are given by the heat kernel $W_{ij} = \exp(-\|x_i - x_j\|_2^2/t)$ and $D_i = \sum_j W_{ij}$.

In LPP, (1) is applied to (2) and (3). By a straightforward algebraic calculation, (2) and (3) are transformed into

$$\begin{array}{ll} \min & \mathrm{Tr}(U^T XLX^T U) \\ \text{s.t.} & U^T XDX^T U = I_m \end{array} \tag{4}$$

where $W = (W_{ij})$, $L = D - W$ and D is the diagonal matrix whose (i, i)th entry is D_i. This optimization problem leads to a generalized eigenvalue problem, i.e.,

$$XLX^T u_i = \lambda_i XDX^T u_i$$

and the optimal solutions $u_1, u_2, \ldots, u_m$ are the eigenvectors corresponding to the m smallest eigenvalues.

Once $\{u_i\}_{i=1}^n$ are computed, the linear projection matrix provided by LPP is given by $U = [u_1\ u_2\ \cdots\ u_m]$. For any new data sample x from $\mathbb{R}^n$, LPP finds its low-dimensional representation y simply by $y = U^T x$.

2) NPP and NPE: The linear projection mapping for LLE is independently provided by NPE [24] and NPP [25]. Similar to LPP, NPE and NPP apply the linear projection assumption (1) to the training process of LLE and reformulate the optimization problem in LLE as to compute the linear projection matrix.

During the training procedure of LLE, a set of linear reconstruction weights $\{W_{ij}\}_{i,j=1}^N$, which best preserves local topology, is first computed.

Then, LLE aims to preserve $\{W_{ij}\}_{i,j=1}^N$ from $\mathcal{X}$ to $\mathcal{Y}$. This is achieved by solving the following optimization problem:

$$\min \quad \sum_{i=1}^{N} \left\| y_i - \sum_{j=1}^{N} W_{ij} y_j \right\|_2^2 \tag{5}$$

$$\text{s.t.} \quad \frac{1}{N}\sum_{i=1}^{N} y_i y_i^T = I_m. \tag{6}$$

In NPE and NPP, the linear projection assumption (1) is used in the above optimization problem; hence, (5) and (6) become

$$\begin{array}{ll} \min & \mathrm{Tr}(U^T XMX^T U) \\ \text{s.t.} & U^T XX^T U = I_m \end{array} \tag{7}$$

where $M = (I_N - W)^T (I_N - W)$ with $W = (W_{ij})$. The optimal solutions $u_1, u_2, \ldots, u_m$ are the eigenvectors of the following generalized eigenvalue problem corresponding to the m smallest eigenvalues:

$$XMX^T u_i = \lambda_i XX^T u_i.$$

After finding the linear projection matrix $U = [u_1 u_2 \cdots u_m]$, any new data sample x from the high-dimensional space $\mathbb{R}^n$ can be easily mapped into the lower dimensional space $\mathbb{R}^m$ by $y = U^T x$.

3) OLPP and ONPP: OLPP [26] and ONPP [27] are the same as LPP and NPE (or NPP), respectively, except that the linear projection matrix provided by LPP and NPE (or NPP) is restricted to be orthogonal. This is achieved by replacing the constraints (4) and (7) with $U^T U = I_m$. Then, the optimization problems in OLPP and ONPP become

$$\begin{array}{ll} \text{OLPP:} & U_{\text{OLPP}} = \underset{U^T U = I_m}{\arg\min}\ \mathrm{Tr}(U^T XLX^T U) \\ \text{ONPP:} & U_{\text{ONPP}} = \underset{U^T U = I_m}{\arg\min}\ \mathrm{Tr}(U^T XMX^T U). \end{array}$$

Unlike in the cases of LPP and NPE (or NPP), these two optimization problems lead to eigenvalue problems that are much easier to numerically solve than a generalized eigenvalue problem. The column vectors of U_{OLPP} are given by the eigenvectors of XLX^T corresponding to the m smallest eigenvalues. The same result holds for U_{ONPP} by replacing XLX^T with XMX^T. The reader is referred to [26] and [27] for details of these two algorithms.

B. Out-of-Sample Nonlinear Extensions for Manifold Learning

Besides linear projections for manifold learning, several out-of-sample nonlinear extensions are also proposed for manifold learning in order to get low-dimensional representations of unseen data samples from the learned manifold.

An important kind of methods is the kernel embedding methods. They are based on kernel functions and extrapolation techniques. Bengio *et al.* [29], [33] proposed a unified framework for extending LLE, ISOMAP, and LE, in which these methods are seen as learning eigenfunctions of operators defined from data-dependent kernels. The data-dependent kernels are implicitly defined by LLE, ISOMAP, and LE and are used together with the Nyström formula [35] to extrapolate the embedding of a manifold learned from finite training samples to new coming samples for LLE, ISOMAP, and LE (see [29] and [33]). Chin and Suter [32] investigated the equivalence between MVU and kernel principal component analysis [36], by which extending MVU to new samples is reduced to extending a kernel matrix. In their work [32], the kernel matrix is generated from an unknown kernel eigenfunction, which is approximated using Gaussian basis functions. A framework was proposed in [30] for efficient KE, which is based on a matrix approximation theorem and an extension of the representer theorem. Under this framework, LLE was reformulated, and the issue of extending LLE to new data samples was addressed in [30].

The kernel versions of the LPP and ONPP methods (KLPP and KONPP) were proposed in [22] and [27], respectively. The common idea is to first map data samples into a feature space and then use the kernel trick to reformulate the optimization problem in the feature space. Finally, the optimization problem in the feature space is solved within a subset of the whole feature space (since the feature mapping is unknown) to obtain a nonlinear mapping from the original high-dimensional manifold into the low-dimensional embedding; hence, the low-dimensional representations of new coming samples can be obtained by the nonlinear mapping. It should be remarked that the nonlinear mapping obtained is given in terms of the kernel and the training data samples; hence, it is very computationally expensive to find the corresponding point in the learned embedding for a new coming data sample if the data set is very large.

Apart from kernel-based methods, a natural out-of-sample extension method for LLE was suggested in [3]. For a new coming data sample, first, its k-nearest neighbors (k-NNs) in the training data set are found. Then, its linear reconstruction weights with respect to the neighbors are computed using the standard process in the LLE method. Finally, its low-dimensional representation is given by the linear combination of the low-dimensional representations of the neighbors with respect to the computed weights. Landmark ISOMAP (L-ISOMAP) [5] is proposed to overcome the computational inefficiency of ISOMAP and can be used to embed new coming data samples. In L-ISOMAP, a portion of the training data samples is first selected as landmark points in the training process. When a new sample arrives, its nearest path lengths to the landmark points are approximated, based on which the low-dimensional embedding is computed through a distance-based triangulation.

Besides, the out-of-sample extension issue is also addressed by the group of incremental manifold learning methods. In the RML method [10], the manifold learning problem is formulated as constructing coordinate charts in Riemannian geometry. The low-dimensional embedding of all training data can be incrementally obtained by preserving the distances and angles in local neighborhoods. Han *et al.* [11] proposed the incremental alignment manifold learning (IAML) algorithm, which first incrementally searches the neighborhood patches and then aligns the low-dimensional embedding of data patch by patch. These incremental manifold learning methods, in which the low-dimensional embedding of all training data is iteratively obtained through solving local optimization problems, have a good performance for out-of-sample extension on very large data sets, although they are time consuming.

The difference between the proposed NPPE method and the above kind of incremental manifold learning methods is as follows. These incremental manifold learning approaches learn the low-dimensional embedding by decomposing the global dimensionality reduction process of all data into local dimensionality reduction problems step by step. By such a strategy, out-of-sample extension to sequentially arrived new data is achieved. However, NPPE uses the whole training data to find a nonlinear and explicit mapping so that the learned mapping can optimally preserve the locally linear reconstruction weights. On the other hand, NPPE obtains an explicit fixed embedding mapping (which is not given in terms of the training data) after training; hence, NPPE is very fast in embedding new data samples on the learned low-dimensional manifold and may be expected to be used for practical online learning problems such as online classification, detection, and visual tracking.

III. Explicit Nonlinear Mappings for Manifold Learning

Here, we propose an explicit nonlinear mapping for manifold learning, based on the assumption that there exists a polynomial mapping between high- and low-dimensional representations. Here, by saying "explicit," we mean that the mapping has a closed-form expression and is independent of the training samples. Precisely, we assume that the kth component y_i^k of y_i is a polynomial of degree p in x_i in the following manner:

$$y_i^k = \sum_{\substack{l_1,l_2,\ldots,l_n \geq 0 \\ 1 \leq l_1+l_2+\cdots+l_n \leq p}} v_k^{\mathbf{l}} \left(x_i^1\right)^{l_1} \left(x_i^2\right)^{l_2} \cdots \left(x_i^n\right)^{l_n} \tag{8}$$

where $l_1, l_2, \ldots, l_n$ are all integers. The superscript $\mathbf{l}$ stands for the n-tuple indexing array $(l_1, l_2, \ldots, l_n)$, and v_k is the vector of polynomial coefficients, which is defined by

$$v_k = \begin{pmatrix} v_k^{\mathbf{l}}|_{l_1=p,l_2=0,\ldots,l_n=0} \\ v_k^{\mathbf{l}}|_{l_1=p-1,l_2=1,\ldots,l_n=0} \\ \vdots \\ v_k^{\mathbf{l}}|_{l_1=1,l_2=0,\ldots,l_n=0} \\ \vdots \\ v_k^{\mathbf{l}}|_{l_1=0,l_2=0,\ldots,l_n=1} \end{pmatrix}.$$

By assuming the polynomial mapping relationship, we aim to find a polynomial approximation to the unknown mapping from the high-dimensional data samples into their low-dimensional embedding space. Compared with the linear projection assumption previously used, a polynomial mapping provides high-order approximation to the unknown nonlinear mapping and, therefore, is more accurate for data samples lying on nonlinear manifolds.

Recently, it has been proven in [28] that most manifold learning methods, including LLE, LE, and ISOMAP, can be cast into the framework of spectral embedding. Under this framework, finding the low-dimensional embedding representations of the high-dimensional data samples is reduced to solving the following optimization problem:

$$\min_{y_i} \quad \frac{1}{2}\sum_{i,j=1}^{N} W_{ij}\|y_i - y_j\|_2^2 \tag{9}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} D_i y_i y_i^T = I_m \tag{10}$$

where W_{ij} $(i, j = 1, 2, \ldots, N)$ are symmetrical and positive weights that can be defined by using the input data samples and $D_i = \sum_{j=1}^N W_{ij}$.

Applying the polynomial assumption (8) to the above general model of manifold learning gives a general manifold learning algorithm with an explicit nonlinear mapping. For $x_i = (x_i^1, x_i^2, \ldots, x_i^n)^T \in \mathbb{R}^n$, denote $(x_i^1)^{l_1}(x_i^2)^{l_2}\cdots(x_i^n)^{l_n}$ by $x_i^{\mathbf{l}}$ and substitute (8) into (9). Then, the objective function becomes

$$\frac{1}{2}\sum_{i,j} W_{ij} \left\| \begin{pmatrix} \sum_{\mathbf{l}} v_1^{\mathbf{l}} x_i^{\mathbf{l}} \\ \vdots \\ \sum_{\mathbf{l}} v_m^{\mathbf{l}} x_i^{\mathbf{l}} \end{pmatrix} - \begin{pmatrix} \sum_{\mathbf{l}} v_1^{\mathbf{l}} x_j^{\mathbf{l}} \\ \vdots \\ \sum_{\mathbf{l}} v_m^{\mathbf{l}} x_j^{\mathbf{l}} \end{pmatrix} \right\|_2^2$$
$$= \frac{1}{2}\sum_{i,j} W_{ij} \sum_k \left(\left(\sum_{\mathbf{l}} v_k^{\mathbf{l}} x_i^{\mathbf{l}}\right) - \left(\sum_{\mathbf{l}} v_k^{\mathbf{l}} x_j^{\mathbf{l}}\right) \right)^2$$
$$= \sum_{i,j} W_{ij} \sum_k \left(\left(\sum_{\mathbf{l}} v_k^{\mathbf{l}} x_i^{\mathbf{l}}\right)^2 - \left(\sum_{\mathbf{l}} v_k^{\mathbf{l}} x_i^{\mathbf{l}}\right)\left(\sum_{\mathbf{l}} v_k^{\mathbf{l}} x_j^{\mathbf{l}}\right) \right)$$

$$
\begin{aligned}
&= \sum_k \left(\sum_i \left(\sum_{\mathbf{l}} v_k^{\mathbf{l}} x_i^{\mathbf{l}} \right) \left(\sum_j W_{ij} \right) \left(\sum_{\mathbf{l}} v_k^{\mathbf{l}} x_i^{\mathbf{l}} \right) \right) \\
&\quad - \sum_k \left(\sum_{i,j} \left(\sum_{\mathbf{l}} v_k^{\mathbf{l}} x_i^{\mathbf{l}} \right) W_{ij} \left(\sum_{\mathbf{l}} v_k^{\mathbf{l}} x_j^{\mathbf{l}} \right) \right) \\
&= \sum_k \left(\sum_i \left(\sum_{\mathbf{l}} v_k^{\mathbf{l}} x_i^{\mathbf{l}} \right) D_i \left(\sum_{\mathbf{l}} v_k^{\mathbf{l}} x_i^{\mathbf{l}} \right) \right) \\
&\quad - \sum_k \left(\sum_{i,j} \left(\sum_{\mathbf{l}} v_k^{\mathbf{l}} x_i^{\mathbf{l}} \right) W_{ij} \left(\sum_{\mathbf{l}} v_k^{\mathbf{l}} x_j^{\mathbf{l}} \right) \right). \qquad (11)
\end{aligned}
$$

Substitute (8) into (10), so the constraint is transformed into

$$
\sum_i D_i \begin{pmatrix} \sum_{\mathbf{l}} v_1^{\mathbf{l}} x_i^{\mathbf{l}} \\ \vdots \\ \sum_{\mathbf{l}} v_m^{\mathbf{l}} x_i^{\mathbf{l}} \end{pmatrix} \left(\sum_{\mathbf{l}} v_1^{\mathbf{l}} x_i^{\mathbf{l}} \cdots \sum_{\mathbf{l}} v_m^{\mathbf{l}} x_i^{\mathbf{l}} \right) = I_m.
$$

This is equivalent to

$$
\sum_i D_i \left(\sum_{\mathbf{l}} v_j^{\mathbf{l}} x_i^{\mathbf{l}} \right) \left(\sum_{\mathbf{l}} v_k^{\mathbf{l}} x_i^{\mathbf{l}} \right) = \delta_{jk} \qquad (12)
$$

where $\delta_{jk} = 1$ for $j = k$ and $\delta_{jk} = 0$ otherwise.

In order to simplify (11) and (12), given an input sample x_i, we define $X_p^{(i)}$ by

$$
X_p^{(i)} = \begin{pmatrix} \overbrace{x_i \otimes x_i \otimes \cdots \otimes x_i}^{p} \\ \vdots \\ x_i \otimes x_i \\ x_i \end{pmatrix} \qquad (13)
$$

where $\otimes$ stands for the Kronecker product defined on matrices. For two matrices $A = (a_{ij})$ and B, $A \otimes B$ is a block matrix whose (i, j)th block is $a_{ij}B$. Then, $\sum_{\mathbf{l}} v_k^{\mathbf{l}} x_i^{\mathbf{l}} = v_k^T X_p^{(i)}$; hence, (11) and (12) are reduced, respectively, to

$$
\begin{aligned}
\min_{v_k} \quad & \sum_k v_k^T \left\{ \sum_i X_p^{(i)} D_i \left(X_p^{(i)} \right)^T \right. \\
& \left. \qquad - \sum_{ij} X_p^{(i)} W_{ij} \left(X_p^{(j)} \right)^T \right\} v_k \qquad (14) \\
\text{s.t.} \quad & v_j^T \left\{ \sum_i X_p^{(i)} D_i \left(X_p^{(i)} \right)^T \right\} v_k = \delta_{jk}. \qquad (15)
\end{aligned}
$$

By writing $X_p = [X_p^{(1)}\ X_p^{(2)}\ \cdots\ X_p^{(N)}]$, (14) and (15) can be further simplified to

$$
\begin{aligned}
\min_{v_k} \quad & \sum_k v_k^T X_p (D - W) X_p v_k \qquad (16) \\
\text{s.t.} \quad & v_j^T X_p D X_p v_k = \delta_{jk} \qquad (17)
\end{aligned}
$$

where $W = (W_{ij})$, and D is a diagonal matrix whose ith diagonal entry is D_i.

By the Rayleigh–Ritz theorem [37], the optimal solutions v_k, $k = 1, 2, \ldots, m$, are the eigenvectors of the following generalized eigenvalue problem corresponding to the m smallest eigenvalues:

$$
X_p (D - W) X_p^T v_i = \lambda X_p D X_p^T v_i, \; v_i^T X_p D X_p^T v_j = \delta_{ij}. \qquad (18)
$$

Once v_k, $k = 1, 2, \ldots, m$, are computed, the explicit nonlinear mapping from the high-dimensional data samples to the low-dimensional embedding space $\mathbb{R}^m$ can be given as

$$
y = \begin{pmatrix} \sum_{\mathbf{l}} v_1^{\mathbf{l}} (x^1)^{l_1} (x^2)^{l_2} \cdots (x^n)^{l_n} \\ \vdots \\ \sum_{\mathbf{l}} v_m^{\mathbf{l}} (x^1)^{l_1} (x^2)^{l_2} \cdots (x^n)^{l_n} \end{pmatrix} \qquad (19)
$$

where x is a high-dimensional data sample, and y is its low-dimensional representation. For a new coming sample x_t, its location in the low-dimensional embedding manifold can be simply obtained by

$$
y_t = \left(v_1^T X_p^{(t)}, v_2^T X_p^{(t)}, \ldots, v_m^T X_p^{(t)} \right)^T \qquad (20)
$$

where $X_p^{(t)}$ is defined in the same way as in (13).

In the next section, we will make use of a similar method as in LLE to define the weights W_{ij}, $i, j = 1, 2, \ldots, N$, so that the geometry of the neighborhood of each data point can be captured.

IV. NPPE

Here, we propose a new manifold learning algorithm with an explicit nonlinear mapping, named NPPE, which is obtained by defining the weights W_{ij}, $i, j = 1, 2, \ldots, N$, in a way similar to the LLE method and combining them with the explicit nonlinear mapping as in the preceding Section III.

A. NPPE

Consider a data set $\{x_1, x_2, \ldots, x_N\}$ from the high-dimensional space $\mathbb{R}^n$. NPPE starts with finding a set of linear reconstruction weights that can best reconstruct each data sample x_i by its k-NNs. This step is identical with that of LLE [2], [3]. The weights R_{ij}, $i, j = 1, 2, \ldots, N$, which are defined to be nonzero only if x_j is among the k-NNs of x_i, are computed by solving the following optimization problem:

$$
R_{ij} = \mathop{\arg\min}_{\sum_{j=1}^N R_{ij} = 1} \sum_{i=1}^N \left\| x_i - \sum_{j=1}^N R_{ij} x_j \right\|_2^2.
$$

The weights R_{ij} represent the linear coefficients for reconstructing the sample x_i from its neighbors $\{x_j\}$. The weight matrix, i.e., $R = (R_{ij})$, has a closed-form solution given by

$$
r_i = \frac{G^{-1} e}{e^T G^{-1} e} \qquad (21)
$$

where r_i is a column vector formed by the k nonzero entries in the ith row of R, and e is a column vector of all ones. The (j, l)th entry of the $k \times k$ matrix G is $(x_j - x_i)^T (x_l - x_i)$, where x_j and x_l are among the k-NNs of x_i.

NPPE aims to preserve the reconstruction weights R_{ij} from the high-dimensional input data samples to their low-dimensional representations under the polynomial mapping assumption. This is achieved by solving the following optimization problem:

$$\mathcal{Y} = \underset{\sum_{i=1}^{N} y_i y_i^T = I_m}{\arg\min} \sum_{i=1}^{N} \left\| y_i - \sum_{j=1}^{N} R_{ij} y_j \right\|_2^2 \tag{22}$$

where each y_i satisfies (8).

By a simple algebraic calculation, it can be shown that (22) is equivalent to (9) and (10) with

$$W_{ij} = R_{ij} + R_{ji} - \sum_{k=1}^{N} R_{ik} R_{kj}, \quad \text{and } D_i = 1. \tag{23}$$

By the result in Section III, the explicit nonlinear mapping can be obtained by solving (18), and the low-dimensional representations $\mathcal{Y}$ of $\mathcal{X}$ can be computed by applying (19) to $\mathcal{X}$. For a new coming sample x_t, its low-dimensional representation can be simply given by (20).

B. SNPPE

As shown in Section IV-C, in the training stage of the NPPE algorithm, as the polynomial degree p increases, the overall computational complexity of the NPPE algorithm exponentially increases with p, which would be extremely time consuming when the data dimension is very high. To overcome this difficulty, we simplify NPPE by removing the crosswise items. This is achieved by replacing the Kronecker product in (13) with the Hadamard product, i.e.,

$$X_p^{(i)} = \begin{pmatrix} \overbrace{x_i \odot x_i \odot \cdots \odot x_i}^{p} \\ \vdots \\ x_i \odot x_i \\ x_i \end{pmatrix} \tag{24}$$

where $\odot$ refers to entrywise matrix multiplication.

As shown in Section IV-C, the simplified NPPE (SNPPE) algorithm has only quadratic running time complexity $O(kNn^2p^2)$ in the training stage and linear running time complexity $O(mnp)$ in locating new samples, where p is the polynomial degree, n is the original data dimension, k stands for the number of nearest neighbors, N is the number of training samples, and m is the dimension of the learned manifold. On the other hand, as shown in Fig. 3 for a `SwissRoll` data set in Section VI below, the SNPPE and NPPE algorithms have a similar and good performance in locating new samples, which are out of the range of the training data. Note that the NPPE algorithm does not work well for high-dimensional real data since it is extremely time consuming due to the high dimensionality of the data, as discussed above; hence, we did not compare the performance of the SNPPE and NPPE algorithms for real data. However, compared with other methods, the SNPPE algorithm has a much better performance for three real data sets (`lleface`, `usps − 0`, and `PIE`), as shown in Figs. 4–6 below.

TABLE II
COMPUTATIONAL COMPLEXITY OF SNPPE, LINEAR METHODS, KERNEL METHODS, AND LLEOOS ON COMPUTING THE LOW-DIMENSIONAL REPRESENTATION OF A NEW COMING SAMPLE, WHERE p IS THE POLYNOMIAL DEGREE FOR SNPPE, AND k STANDS FOR THE NUMBER OF NEAREST NEIGHBORS

Methods	Complexity
SNPPE	$O(mpn)$
Linear	$O(mn)$
Kernel	$O(nN)$
LLEoos	$O((N+k^2)n + N\log(N))$

The NPPE and SNPPE algorithms are summarized in Algorithm 1.

Algorithm 1: The NPPE/SNPPE Algorithm

```
Input: Data matrix $X$, the number $k$ of nearest neighbors,
       and the polynomial degree $p$.
Output: Vectors of polynomial coefficients $v_i, i = 1, 2, \ldots, m$.
1 Compute $R_{ij}$ by (21).
2 Compute $W$ and $D$ by (23).
3 Generate $X_p$
      NPPE:Generate $X_p$ according to (13).
      SNPPE:Generate $X_p$ according to (24).
4 Solve the generalized eigenvalue problem (18) to get $v_i$,
  $i = 1, 2, \ldots, m$.
```

C. Computational Complexity of NPPE and SNPPE

In the training stage of NPPE, the computational complexity of generating X_p is $O(N\sum_{i=2}^{p} n^i)$. Computing $X_pWX_p^T$ and $X_pDX_p^T$ takes $O(kN(\sum_{i=1}^{p} n^i)^2)$ and $O(N(\sum_{i=1}^{p} n^i)^2)$ operations, respectively, since there are only k nonzero entries in each column of W, and D is a diagonal matrix. The computational complexity of the final eigendecomposition is $O(m(\sum_{i=1}^{p} n^i)^2)$, which is the most time-consuming step.

In the procedure of locating new samples with NPPE, generating $X_p^{(t)}$ takes $O(\sum_{i=2}^{p} n^i)$ operations, and computing y_t takes $O(m\sum_{i=1}^{p} n^i)$ operations.

In the training stage of SNPPE, the computational complexity of generating X_p becomes $O(Npn)$, whereas the computational complexity of computing $X_pWX_p^T$ and $X_pDX_p^T$ becomes $O(kNn^2p^2)$ and $O(Nn^2p^2)$, respectively. The computational complexity of the final eigendecomposition and computing y_t is $O(mn^2p^2)$ and $O(mnp)$, respectively. Therefore, in the training stage, the overall computational complexity of the SNPPE algorithm is $O(kNn^2p^2)$.

Finally, in the stage of locating new coming samples in the low-dimensional embedding, the computational complexity of the SNPPE, linear methods, kernel methods, and the natural out-of-sample method for LLE (denoted by LLEoos for short) [3] is summarized in Table II. The computational complexity of different kernel methods varies. Here, we only state the computational complexity of the common step of computing the inner products and the projections in the feature space for the Gaussian and polynomial kernels.

V. Comparison With the Kernel Embedding Methods

Here, we give a detailed comparison between the NPPE method and the kernel embedding methods.

A. Kernel Embedding Methods

The standard idea behind the kernel embedding methods can be summarized as follows. First, the original data samples $\{x_i\}$ are mapped into an unknown Hilbert space $\mathcal{H}$ through a nonlinear feature mapping $\psi : \mathbb{R}^n \to \mathcal{H}$, and the kth component of y_i can be written in terms of $\psi(x_i)$ as

$$y_i^k = w_k^T \psi(x_i)$$

for some vectors w_k, $k = 1, \ldots, m$. Then, w_k, $k = 1, 2, \ldots, m$ are computed by solving the following optimization problem:

$$\begin{aligned} \min_{w_i} \quad & \sum_{i=1}^{m} w_i^T \psi(X)(D-W)\psi^T(X) w_i \\ \text{s.t.} \quad & w_i^T \psi(X) D \psi^T(X) w_j = \delta_{ij} \end{aligned} \tag{25}$$

where $\psi(X) = [\psi(x_1)\ \psi(x_2)\ \cdots\ \psi(x_N)]$.

However, since both $\mathcal{H}$ and ψ are unknown and $\mathcal{H}$ may be infinite-dimensional, the optimization problem (25) cannot be directly solved in $\mathcal{H}$. The common tradeoff taken by the kernel embedding methods is to constrain $\{w_i\}_{i=1}^m$ to lie within the range of $\psi(X)$, that is, there exists $\alpha_i \in \mathbb{R}^N$ such that $w_i = \psi(X)\alpha_i$. Then, (25) is reformulated as

$$\begin{aligned} \min_{\alpha_i} \quad & \sum_{i=1}^{m} \alpha_i^T K(D-W) K \alpha_i \\ \text{s.t.} \quad & \alpha_i^T K D K \alpha_j = \delta_{ij} \end{aligned} \tag{26}$$

where $K_{ij} = \langle \psi(x_i), \psi(x_j) \rangle = \kappa(x_i, x_j)$, and $\kappa(\cdot,\cdot) : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ is a kernel function. The solution α_i of the problem (26) is the eigenvector of the following generalized eigenvalue problem corresponding to the ith smallest eigenvalue:

$$K(D-W)K\alpha_i = \lambda_i K D K \alpha_i, \quad \alpha_i^T K D K \alpha_j = \delta_{ij}.$$

Finally, for a new coming data sample x_t, its low-dimensional representation y_t is given by

$$\begin{aligned} y_t &= \left[w_1^T \psi(x_t)\, w_2^T \psi(x_t) \cdots w_m^T \psi(x_t) \right]^T \\ &= \left[\alpha_1^T \psi^T(X)\psi(x_t)\ \alpha_2^T \psi^T(X)\psi(x_t) \right. \\ &\qquad \left. \cdots \alpha_m^T \psi^T(X)\psi(x_t) \right]^T \\ &= \begin{pmatrix} \alpha_1^T \\ \vdots \\ \alpha_m^T \end{pmatrix} \begin{pmatrix} \kappa(x_1, x_t) \\ \vdots \\ \kappa(x_N, x_t) \end{pmatrix}. \end{aligned}$$

B. Feature Mapping Perspective of SNPPE

For any data vector $x \in \mathbb{R}^n$ define the mapping $\phi : \mathbb{R}^n \to \mathbb{R}^{pn}$ as

$$\phi(x) = \begin{pmatrix} \overbrace{x \odot x \odot \cdots \odot x}^{p} \\ \vdots \\ x \odot x \\ x \end{pmatrix}.$$

TABLE III
ABBREVIATIONS FOR DATA SETS AND METHODS IN SECTION VI

Notation	Description
`rsw1k`	`SwissRoll` with 1000 randomly generated data
`rsw10k`	`SwissRoll` with 10000 randomly generated data
KE	Kernel Extrapolation [30]
LLEoos	LLE with out-of-sample extension [3]
g-KONPP	Kernel ONPP with Gaussian kernel [27]
p-KONPP	Kernel ONPP with polynomial kernel [27]

Then, in the SNPPE algorithm, the process of generating $X_p^{(i)}$ by (24) can be seen as mapping x_i by ϕ into a finite-dimensional feature space $\mathcal{F} \subset \mathbb{R}^{pn}$. In fact, ϕ can be viewed as a feature mapping from the original data space into the finite-dimensional feature space $\mathcal{F}$. It reformulates x as a higher dimensional vector being composed of monomials of x's components.

With the above notation, the polynomial assumption (8) can be rewritten as

$$y_i^k = v_k^T \phi(x_i)$$

where $v_k \in \mathbb{R}^{pn}$ with $k = 1, \ldots, m$, and the optimization problem (16)–(17) becomes

$$\begin{aligned} \min_{v_i} \quad & \sum_{i=1}^{m} v_i^T \phi(X)(D-W)\phi^T(X) v_i \\ \text{s.t.} \quad & v_i^T \phi(X) D \phi^T(X) v_j = \delta_{ij} \end{aligned} \tag{27}$$

where $\phi(X) = [\phi(x_1)\, \phi(x_2)\, \cdots\, \phi(x_N)]$ is a $(pn) \times N$ matrix. After finding the solution v_i $(i = 1, 2, \ldots, m)$ of the problem (27), for a new coming data sample x_t, its low-dimensional representation y_t is simply given by

$$y_t = \left[v_1^T \phi(x_t)\, v_2^T \phi(x_t)\ \cdots\ v_m^T \phi(x_t) \right]^T.$$

C. Discussion

From the above two subsections, the difference between the SNPPE algorithm and the kernel embedding methods can be summarized as follows.

- In the SNPPE algorithm, the feature mapping ϕ is an explicitly defined polynomial of degree p, whereas in the kernel embedding methods, the feature mapping ψ is unknown.
- In the SNPPE algorithm, the feature space $\mathcal{F}$ is a finite-dimensional subspace of $\mathbb{R}^{pn}$, whereas in the kernel embedding methods, the feature space $\mathcal{H}$ is an unknown Hilbert space induced by the kernel κ and may be infinite-dimensional.
- In the SNPPE algorithm, the optimization problem (27) is directly solved in the feature space; hence, the local geometric property of the data manifold can be optimally preserved in the feature space $\mathcal{F}$. However, in the kernel embedding methods, the optimization problem (25) is solved within the range of $\psi(X)$, which is only a subset of the feature space. Therefore, the geometric characteristics of the data manifold may not be optimally preserved in the feature space $\mathcal{H}$.

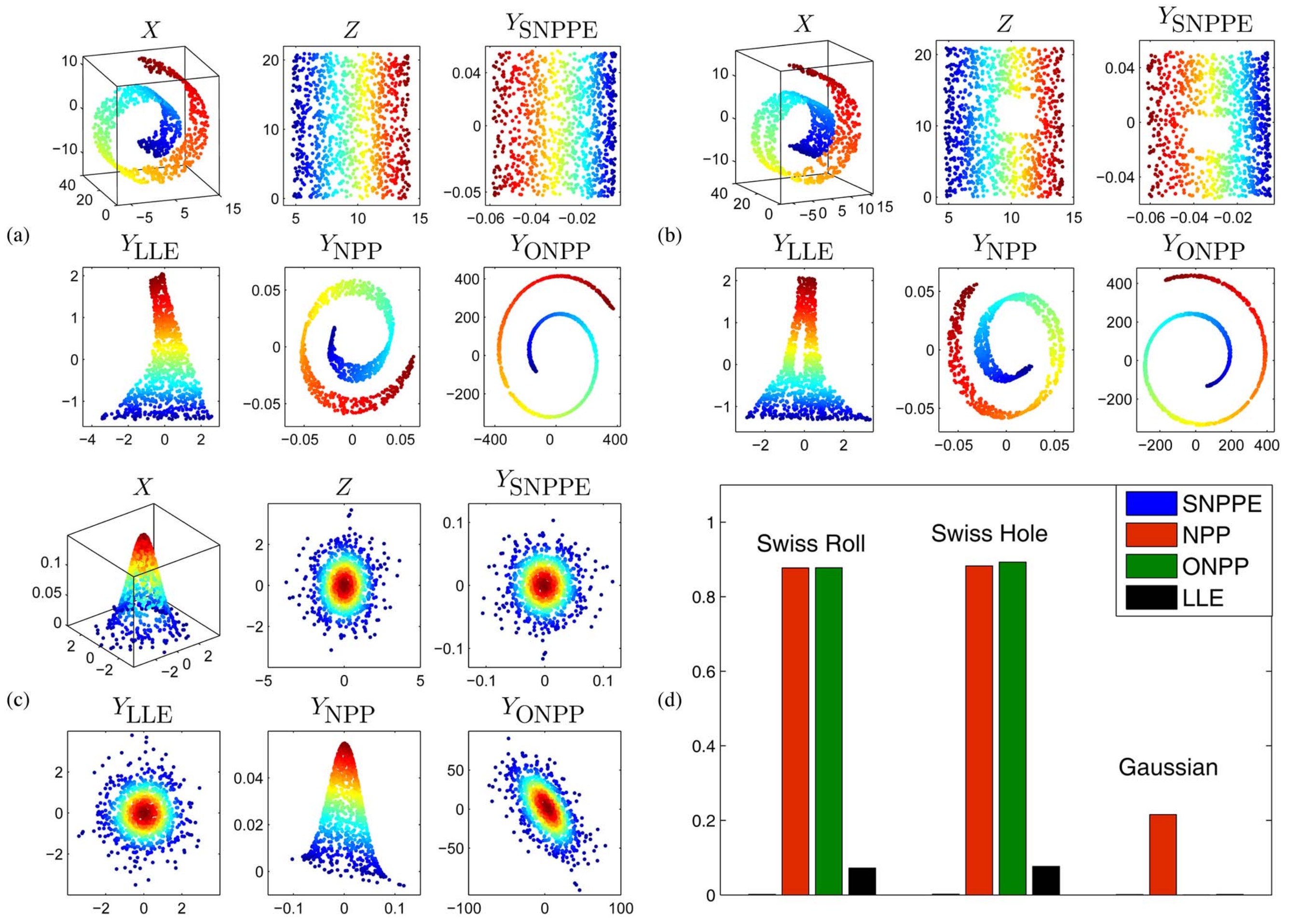


Fig. 1. Experiments on unfolding surfaces embedded in $\mathbb{R}^3$. In each subfigure, X stands for the training data, and Z stands for the generating data. Y_{SNPPE}, Y_{LLE}, Y_{NPP}, and Y_{ONPP} stand for the embedding given by SNPPE, LLE, NPP, and ONPP, respectively. (a) Learning results on `SwissRoll`. (b) Learning results on `SwissHole`. (c) Learning results on `Gaussian`. (d) Bar plot of PM values between Y and Z. (Note that the smaller the PM value is, the better the embedding result is; PM values for results obtained by SNPPE are too small to be clearly seen and are given in Table IV.)

- The SNPPE algorithm provides a closed-form explicit mapping from $\mathcal{X}$ to $\mathcal{Y}$, which does not depend on the training data and is very fast to embed new coming data. To the contrary, the mapping obtained by the kernel embedding methods is expressed in terms of the kernel and the training data; hence, it may be inefficient in dealing with very large data sets.

VI. Experimental Tests

Here, experiments on both synthetic and real-world data sets are conducted to illustrate the validity and effectiveness of the proposed NPPE algorithm. In the experiments, the simplified version of NPPE with a quadratic mapping, that is, the SNPPE algorithm with polynomial degree $p = 2$, is implemented. The SNPPE method is also compared with the linear methods, including NPP [25] and ONPP [27] (which apply the linear mapping and orthogonal linear projection mapping to the training procedure for LLE, respectively), as well as nonlinear methods, including the KE method [30], the KONPP method [27], and the LLEoos [3]. Some abbreviations for data sets and methods used in this section are summarized in Table III.

There are two parameters in the SNPPE algorithm, namely, the number k of nearest neighbors and the polynomial degree p.

TABLE IV
PM Values of Data Sets in Fig. 1

Dataset	SNPPE	NPP	ONPP	LLE
`SwissRoll`	0.0044	0.9937	0.9883	0.2329
`SwissHole`	0.0074	0.9904	0.9910	0.1570
`Gaussian`	0.0012	0.7336	0.0017	0.0013

k is usually set to be 1% of the number of training samples, and the experimental tests show that SNPPE is stable around this number. The choice of p depends on the dimension m. When m is small, p can be large to make SNPPE more accurate. When m is large, p should be small to make SNPPE computationally efficient. Experiments show that SNPPE with $p = 2$ is already accurate enough.

The experiments are conducted on a desktop PC with Intel(R) Pentium 3.6-GHz CPU and 2-G RAM. All algorithms are implemented in MATLAB.

A. *Learning Surfaces in $\mathbb{R}^3$ With SNPPE*

In the first experiment, SNPPE, NPP, ONPP, and LLE are applied to the task of unfolding surfaces embedded in $\mathbb{R}^3$. The surfaces are the `SwissRoll`, `SwissHole`, and `Gaussian`, all of which are generated by the Matlab Demo available at http://www.math.umn.edu/~wittman/mani/. On each manifold,

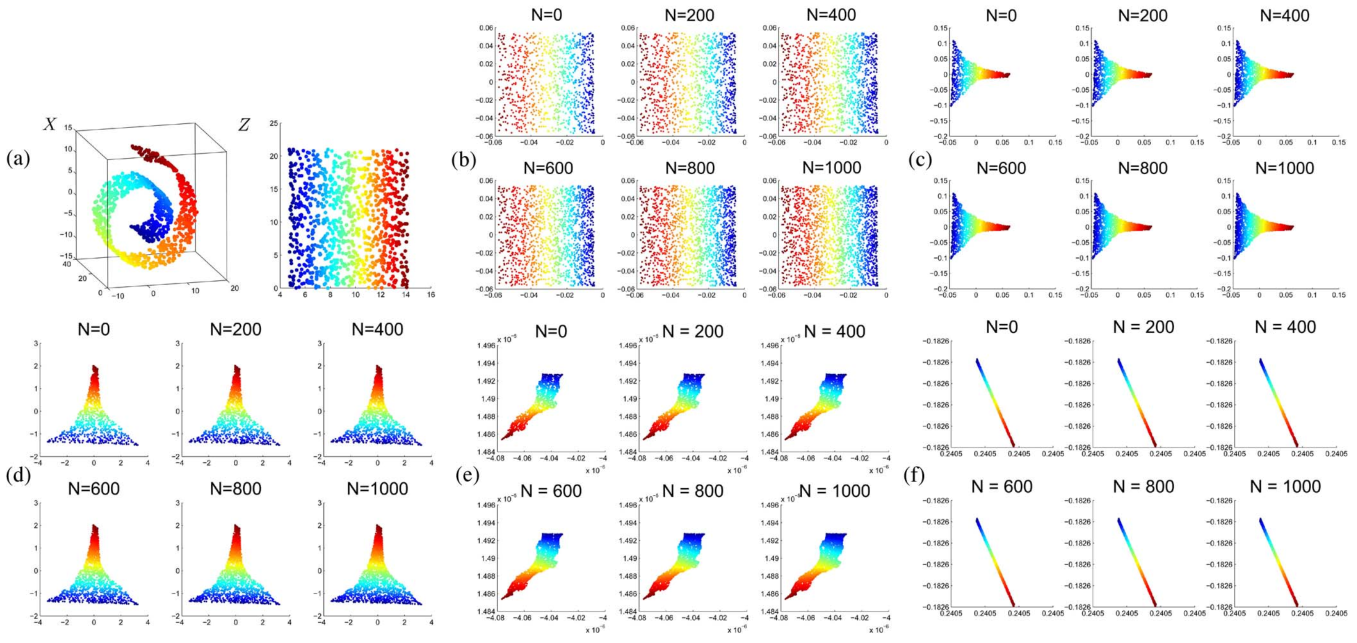


Fig. 2. Experiment on locating new samples for randomly distributed `SwissRoll` data set `rsw1k`. (a) Training and generating data. (b)–(f) Locating results by SNPPE, KE, LLEoos, g-KONPP, and p-KONPP, respectively. In (b)–(f), $N = 0$ stands for the training result.

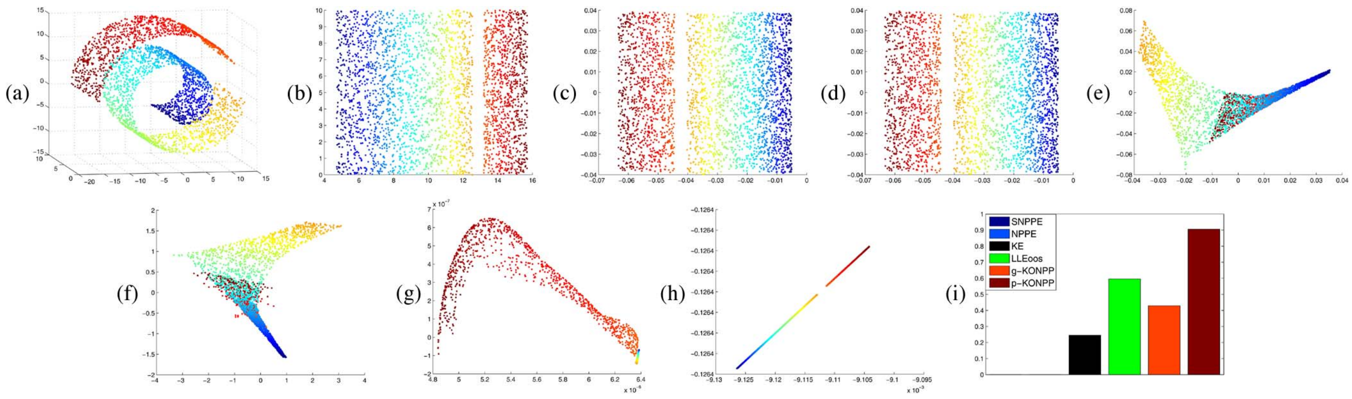


Fig. 3. Experiment on locating new samples that are out of the range of training data. (a) Training (lower part) and testing data (upper and red part). (b) Ground truth of training and testing data (red part). (c)–(h) Locating results (red part) by SNPPE, NPPE, KE, LLEoos, g-KONPP, and p-KONPP, respectively. (i) Bar plot of PM values. (Note that the smaller the PM value is, the better the embedding result is; PM values for results obtained by SNPPE are too small to be clearly seen and are given in Table V.)

1000 data samples are randomly generated for training. The number of nearest neighbors is $k = 10$ and the polynomial degree $p = 2$. The experimental results are shown in Fig. 1. In each subfigure, $Z = [z_1\, z_2\, \cdots\, z_N]$ stands for the generating data such that $x_i = f(z_i)$, where f is the nonlinear mapping that embeds Z in $\mathbb{R}^3$. It is shown in Fig. 1 that SNPPE outperforms all the other three methods, even the LLE method itself. NPP and ONPP fail to unfold these nonlinear manifolds (except for ONPP on `Gaussian`).

Furthermore, in order to quantitatively evaluate the similarity between the learned low-dimensional representations and the generating data, the Procrustes measure (PM) [42], which is a nonlinear measurement of goodness and currently used in embedding quality assessment [40], [41], is used to access the performance of SNPPE and its counterparts. PM evaluates the similarity between $\mathcal{Y}$ and $\mathcal{Z}$ under rigid motion and scaling. More accurate embedding corresponds to a smaller PM value. The evaluation results are shown in Table IV. It can be seen that the embedding given by SNPPE is the most accurate one.

TABLE V
PM VALUES OF DATA SETS IN FIG. 3

	SNPPE	NPPE	KE	LLEoos	g-KONPP	p-KONPP
PM	0.00013	0.00009	0.25	0.60	0.58	0.91

B. Locating New Data Samples With SNPPE

In the second experiment, we apply SNPPE, KE, LLEoos, g-KONPP, and p-KONPP to locating new coming samples in the learned low-dimensional space. First, 2000 data samples are randomly generated from the `SwissRoll` manifold, as shown in Fig. 2(a). Then, 1000 samples are randomly selected as the training data to learn the mapping relationship from $\mathbb{R}^3$ to $\mathbb{R}^2$ by these nonlinear mapping methods. For the Gaussian and polynomial kernels, the kernel width and polynomial degree are set to be 10 and 2, respectively. The learned mappings are used to provide the low-dimensional representations for the rest 1000 samples.

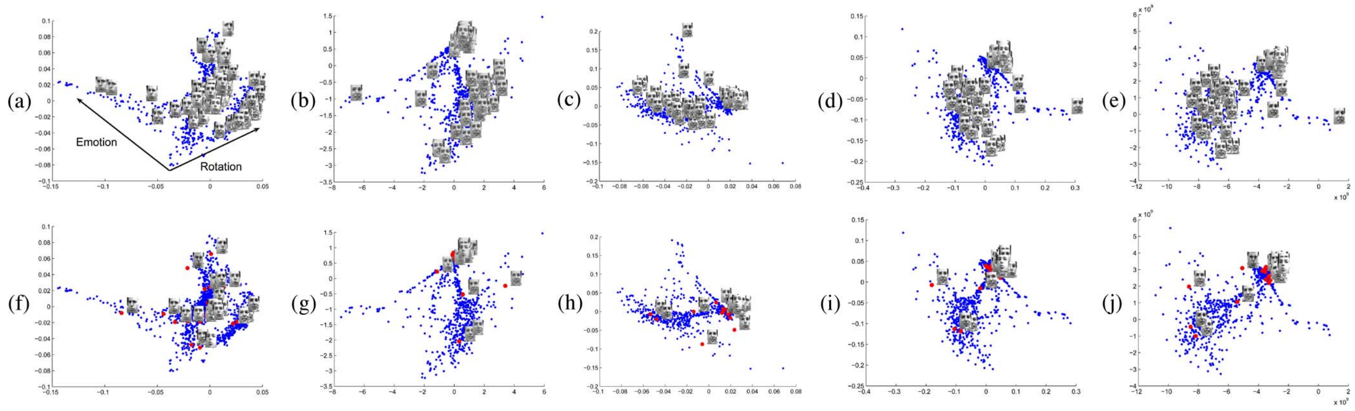


Fig. 4. Experiment on `lleface` data. Training results are plotted by blue dots, whereas testing results are marked with filled red circles. (a) and (f) Learning and testing results by SNPPE. (b) and (g) Learning and testing results by LLEoos. (c) and (h) Learning and testing results by KE. (d) and (i) Learning and testing results by g-KONPP. (e) and (j) Learning and testing results by p-KONPP.

Experimental results are shown in Fig. 2. It can be seen that SNPPE outperforms all the other methods and gives an accurate locating result.

Besides, we conducted an experimental test to validate the generalization ability of SNPPE to new samples that are out of the range of training data. 2000 points on the `SwissRoll` data set are randomly generated for training [lower part in Fig. 3(a)]. 1000 different points, which are out of the range of training data [upper part in Fig. 3(a)], are used for testing. The learned embedding given by SNPPE and NPPE is shown in Fig. 3(c) and (d), and the PM values are illustrated in Table V. It can be seen that SNPPE and NPPE both generalize well to the out-of-range data with a similar performance. Moreover, SNPPE and NPPE outperform all the other algorithms.

To further validate the performance of SNPPE on large data sets, we randomly generate 11 000 samples on the `SwissRoll` manifold, i.e., 1000 for training and 10 000 for testing. The experimental procedure is just the same as the preceding one. The PM values between the generating data of the testing samples and their low-dimensional representations given by the five methods are illustrated in Fig. 7(a). Time cost versus number of testing samples is shown in Fig. 7(b). The experimental results show that SNPPE is more accurate than all the other four methods. Note that, in all the above experiments, the time cost of SNPPE is comparable to that of the linear methods and much lower than the other nonlinear methods.

C. Learning Image Manifolds With SNPPE

In the last experiment, SNPPE is applied to extract intrinsic degrees of freedom underlying two image manifolds, namely, the `lleface` [2] and `usps` $-$ `0`.

The `lleface` consists of 1965 face images of the same person at resolution 28×20, and the two intrinsic degrees of freedom underlying the face images are rotation of the head and facial emotion. We randomly select 1500 samples as the training data and 400 samples as the testing data. The number of nearest neighbors is 15, the kernel width is 256, and the degree of the polynomial kernel is 2. The experimental results are shown in Fig. 4. 100 training samples and 40 testing samples are randomly selected and attached to the learned embedding. It can be seen that SNPPE has successfully recovered the underlying structure of `lleface`, whereas the results given by g-KONPP and p-KONPP are not satisfactory. The rotation degree is not extracted by the learned embedding with the two methods. KE and LLEoos fail to provide meaningful embedding results for this data set. Time cost on locating new data samples by these three methods is shown in Fig. 7(c). The time cost of SNPPE is lower than that of all the other nonlinear methods, which supports the analysis of computational complexity in Section IV-C.

The `usps` $-$ `0` data set consists of 765 images of handwritten digit "0" at resolution 16×16, and the two underlying intrinsic degrees of freedom are the line width and the shape of "0." 600 samples are randomly selected as training data, and 150 samples are chosen to be testing data. The number of nearest neighbors is set to be 5, the kernel width is 2, and the degree of the polynomial kernel is 2. Fig. 5 illustrates the experimental results. 100 training samples and 20 testing samples are randomly selected and shown in the learned embedding. It can be seen that SNPPE and p-KONPP have successfully recovered the underlying structure, whereas it is hard to see the change of line width in the embeddings given by KE, g-KONPP, and LLEoos. The time cost on locating new data samples by these methods is shown in Fig. 7(d). The time cost of SNPPE is much lower than all the other nonlinear methods.

Finally, we present experimental results on a large real data set, the `PIE` face data set. It consists of 11 554 images of different human faces at resolution 32×32, and the two underlying intrinsic degrees of freedom are the orientation of faces and lighting directions. 10 000 samples are randomly selected as training data, and 1554 samples are chosen to be testing data. The number of nearest neighbors is set to be 5, the kernel width is 2326.2, and the polynomial degree is 2. Fig. 6 illustrates the experimental results. 100 training samples and 50 testing samples are randomly selected and shown in the learned embedding. It can be seen that SNPPE have successfully recovered the underlying structure, whereas it is hard to see the change of the face orientation or lighting direction in the embedding given by KE, p-KONPP, g-KONPP, and LLEoos. The time cost of these methods is shown in Fig. 7(e), which shows that the time cost of SNPPE is much lower compared with the other algorithms.

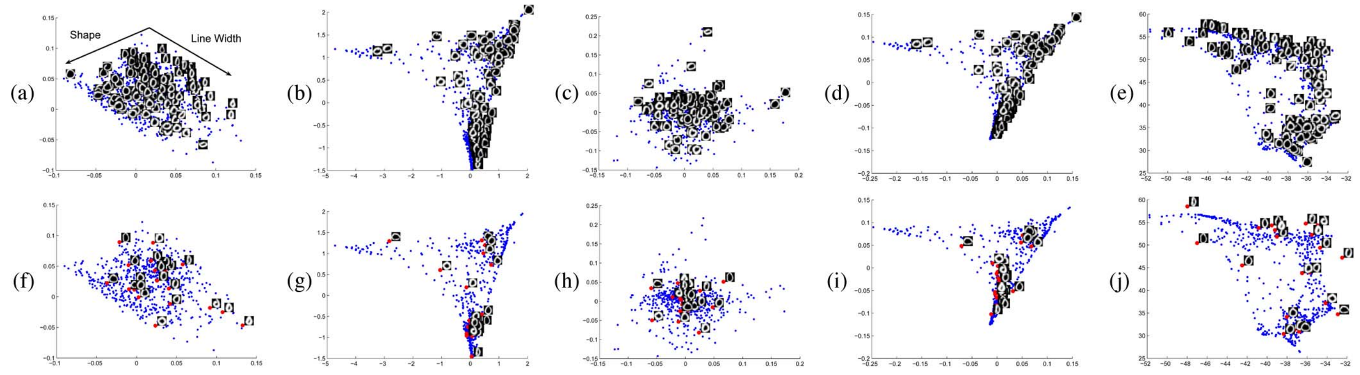


Fig. 5. Experiment on `usps` data. Training results are plotted by blue dots, whereas testing results are marked with filled red circles. (a) and (f) Learning and testing results by SNPPE. (b) and (g) Learning and testing results by LLEoos. (c) and (h) Learning and testing results by KE. (d) and (i) Learning and testing results by g-KONPP. (e) and (j) Learning and testing results by p-KONPP.

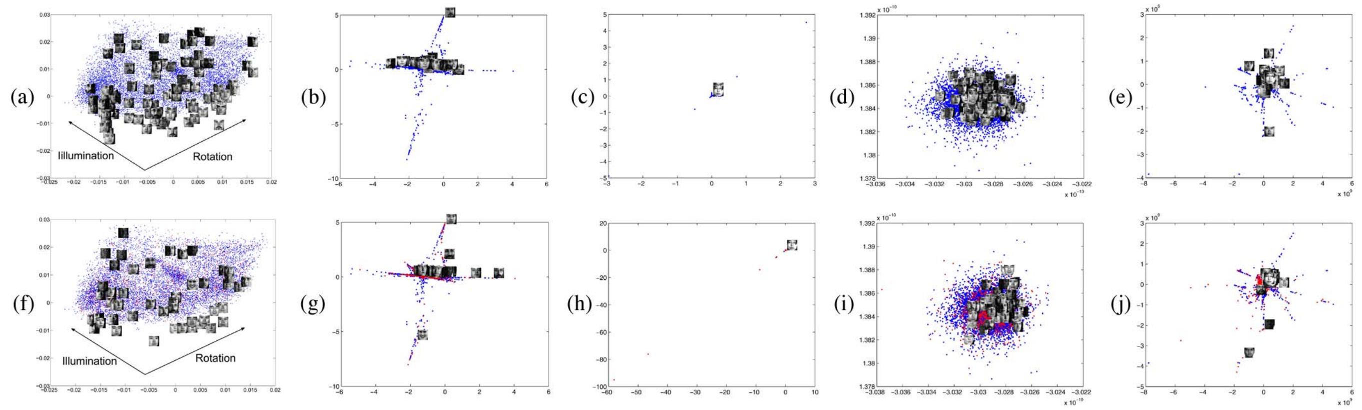


Fig. 6. Experiment on `PIE` data. Training results are plotted by blue dots, whereas testing results are marked with filled red circles. (a) and (f) Learning and testing results by SNPPE. (b) and (g) Learning and testing results by LLEoos. (c) and (h) Learning and testing results by KE. (d) and (i) Learning and testing results by g-KONPP. (e) and (j) Learning and testing results by p-KONPP.

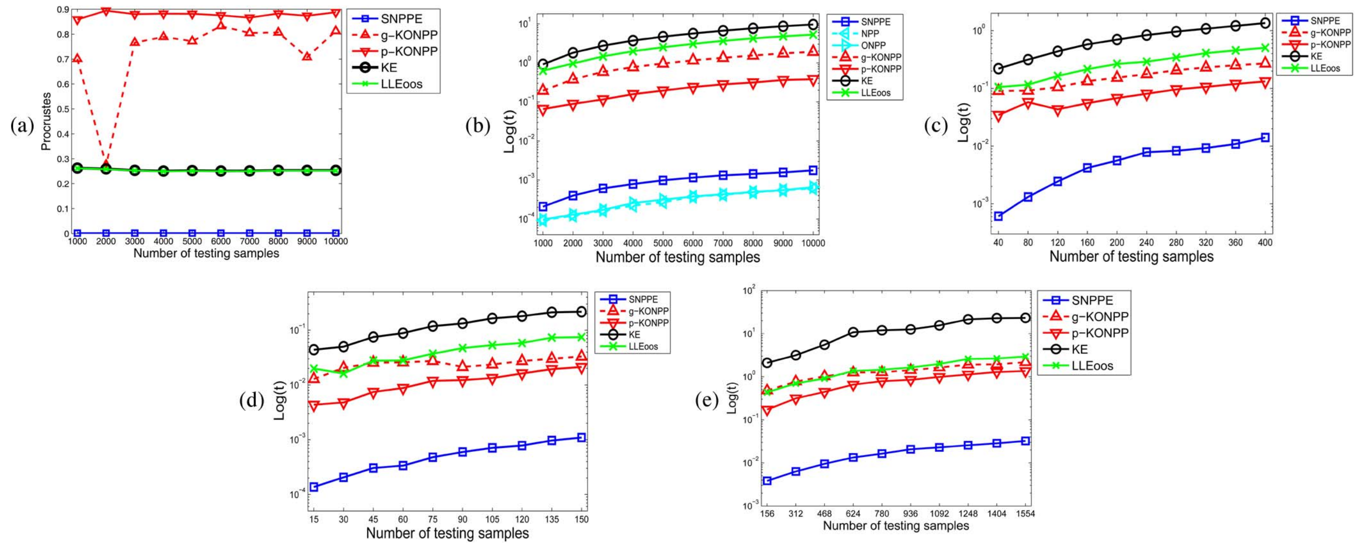


Fig. 7. Time cost and PM values on experiments conducted in Section VI. (a) PM value versus number of testing samples on `rsw10k`. (b) Log plot of time cost versus number of testing samples on `rsw10k`. (c) Log plot of time cost versus number of testing samples on `lleface`. (d) Log plot of time cost versus number of testing samples on `usps`. (e) Log plot of time cost versus number of testing samples on `pie`.

VII. Conclusion and Future Works

A. Conclusion

In this paper, an explicit nonlinear mapping for manifold learning is proposed for the first time. Based on the assumption that there exists a polynomial mapping between high- and low-dimensional representations, an explicit polynomial mapping is obtained by applying this assumption to a generic model of manifold learning. Furthermore, the SNPPE algorithm is an NDR technique with an explicit nonlinear mapping, which tends to preserve not only the locality but also the nonlinear geometry of the high-dimensional data samples. SNPPE can provide convincing embedding results and locate new coming data samples in the reduced low-dimensional space simply and quickly at the same time. Experimental tests on both synthetic and real-world data have validated the effectiveness of the proposed SNPPE algorithm.

Compared with linear projection methods (such as LPP and ONPP), kernel-based nonlinear mapping methods (such as KLPP and KONPP) and incremental methods (such as RML and IAML), SNPPE gives a much better performance with linear running time complexity in locating new coming samples. However, kernel-based and incremental methods are time consuming for very large data sets. Since SNPPE is very fast and more accurate in embedding new data samples on the learned low-dimensional manifold, it may be expected to be used for practical online learning problems such as online classification, detection, and visual tracking.

B. Future Works

A challenging and important issue for manifold learning is how to obtain an explicit and nonlinear bidirectional mapping that can compress high-dimensional data and also reconstruct them from their low-dimensional representations directly. There are already some work trying to address this issue (see, for example, the neural network approach [38] and the regression approach [39]). The mapping in [38] is constructed as an artificial neural network, which has no closed-form expression. The mapping obtained in [39] is given in terms of a kernel function and depends on the training samples; hence, it will be time consuming in dealing with large data sets.

In future work, we hope to address this issue as well and to extend SNPPE to a bidirectional mapping directly.

Acknowledgment

The authors would like to thank the referees for their valuable comments and suggestions, which helped improve the presentation of this paper.

References

[1] H. S. Seung and D. D. Lee, "The manifold ways of perception," *Science*, vol. 290, no. 5500, pp. 2268–2269, Dec. 2000.

[2] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, Dec. 2000.

[3] L. K. Saul and S. T. Roweis, "Think globally, fit locally: Unsupervised learning of low dimensional manifold," *J. Mach. Learn. Res.*, vol. 4, pp. 119–155, Dec. 2003.

[4] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 2000.

[5] V. de Silva and J. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2003, vol. 15, pp. 705–712.

[6] L. Yang, "Alignment of overlapping locally scaled patches for multidimensional scaling and dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 3, pp. 438–490, Mar. 2008.

[7] K. Q. Weinberger and L. K. Saul, "Unsupervised learning of image manifolds by semidefinite programming," *Int. J. Comput. Vis.*, vol. 70, no. 1, pp. 77–90, Oct. 2006.

[8] Z. Zhang and H. Zha, "Principal manifolds and nonlinear dimension reduction via local tangent space alignment," *SIAM J. Sci. Comput.*, vol. 26, no. 1, pp. 313–338, 2004.

[9] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, Jun. 2003.

[10] T. Lin and H. Zha, "Riemannian manifold learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 5, pp. 796–809, May 2008.

[11] Z. Han, D. Meng, Z. Xu, and N. Gu, "Incremental alignment manifold learning," *J. Comput. Sci. Technol.*, vol. 26, no. 1, pp. 153–165, Jan. 2011.

[12] R. R. Coifman and S. Lafon, "Diffusion maps," *Appl. Comput. Harmon. Anal.*, vol. 21, pp. 5–30, 2006.

[13] D. Donoho and C. Grimes, "Hessian eigenmaps: New locally linear embedding techniques for high-dimensional data," *Proc. Nat. Acad. Sci.*, vol. 100, no. 10, pp. 5591–5596, May 2003.

[14] J. Lee and M. Verleysen, *Nolinear Dimensionality Reduction*. Berlin, Germany: Springer-Verlag, 2007.

[15] L. Wang and D. Suter, "Learning and matching of dynamics shape manifolds for human action recognition," *IEEE Trans. Image Process.*, vol. 16, no. 6, pp. 1646–1661, Jun. 2007.

[16] J. Chen, R. Wang, S. Yan, S. Shan, X. Chen, and W. Gao, "Enhancing human face detection by resampling examples through manifolds," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 6, pp. 1017–1028, Nov. 2007.

[17] C. M. Bachmann, T. L. Ainsworth, and R. A. Fusina, "Exploiting manifold geometry in hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 3, pp. 441–454, Mar. 2005.

[18] A. Elgammal and C. S. Lee, "Nonlinear manifold learning for dynamic shape and dynamic appearance," *Comput. Vis. Image Understand.*, vol. 106, no. 1, pp. 31–46, Apr. 2007.

[19] Q. Wang, G. Xu, and H. Ai, "Learning object intrinsic structure for robust visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recog.*, 2003, vol. 2, pp. 227–233.

[20] H. Qiao, P. Zhang, B. Zhang, and S. Zheng, "Learning an intrinsic variable preserving manifold for dynamic visual tracking," *IEEE Trans. Syst., Man., Cybern. B, Cybern.*, vol. 40, no. 3, pp. 868–880, Jun. 2010.

[21] H. Qiao, P. Zhang, B. Zhang, and S. Zheng, "Tracking feature extraction based on manifold learning framework," *J. Exper. Theor. Artif. Intell.*, vol. 23, no. 1, pp. 23–38, Mar. 2011.

[22] X. He and P. Niyogi, "Locality preserving projections," *Proc. Adv. Neural Inf. Process. Syst.*, pp. 1–8, 2003.

[23] X. He, S. Yan, Y. Hu, P. Niyogi, and H. Zhang, "Face recognition using Laplacianfaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 3, pp. 328–340, Mar. 2005.

[24] X. He, D. Cai, S. Yan, and H. Zhang, "Neighborhood preserving embedding," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2005, vol. 2, pp. 1208–1213.

[25] Y. Pang, L. Zhang, Z. Liu, N. Yu, and H. Li, "Neighborhood preserving projections (NPP): A novel linear dimension reduction method," in *Proc. ICIC (1)*, 2005, pp. 117–125.

[26] D. Cai, X. He, J. Han, and H. Zhang, "Orthogonal Laplacianfaces for face recognition," *IEEE Trans. Image Prcess.*, vol. 15, no. 11, pp. 3608–3614, Nov. 2006.

[27] E. Kokiopoulou and Y. Saad, "Orthogonal neighborhood preserving projections: A projection-based dimensionality reduction technique," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 12, pp. 2143–2156, Dec. 2007.

[28] S. Yan, D. Xu, B. Zhang, H. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 40–51, Jan. 2007.

[29] 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 *Proc. Advances Neural Inf. Process. Syst.*, 2003, vol. 16, pp. 177–184.

[30] S. V. N. Vishwanathana, K. M. Borgwardt, O. Guttmana, and A. Smola, "Kernel extrapolation," *Neurocomputing*, vol. 69, no. 7–9, pp. 721–729, Mar. 2006.

[31] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labelled and unlabelled examples," *J. Mach. Learn. Res.*, vol. 7, pp. 2399–2434, Dec. 2006.

[32] T. Chin and D. Suter, "Out-of-sample extrapolation of learned manifolds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 9, pp. 1547–1556, Sep. 2008.

[33] Y. Bengio, O. Delalleau, N. Le Roux, J.-F. Paiement, P. Vincent, and M. Ouimet, "Learning eigenfunctions links spectral embedding and kernel PCA," *Neural Comput.*, vol. 16, no. 10, pp. 2197–2219, Oct. 2004.

[34] M. Law and A. Jain, "Incremental nonlinear dimensionality reduction by manifold learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 3, pp. 377–391, Mar. 2006.

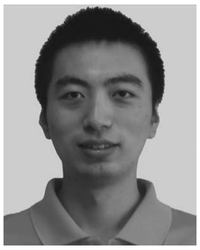
[35] C. Baker, *The Numerical Treatment of Integral Equations*. Oxford, U.K.: Clarendon, 1977.

[36] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998.
[37] J. R. Magnus and H. Neudecker, *Matrix Differential Calculus With Applications in Statistics and Econometrices*. New York: Wiley, 1999, Revised Ed.
[38] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.
[39] M. A. Carreira-Perpinan and Z. Lu, "Dimensionality reduction by unsupervised regression," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recog.*, 2008, vol. 1–12, pp. 2523–2530.
[40] S. Xiang, F. Nie, C. Zhang, and C. Zhang, "Nonlinear dimensionality reduction with local spline embedding," *IEEE. Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1285–1298, Sep. 2009.
[41] Y. Goldberg and Y. Ritov, "Local procrustes for manifold embedding: a measure of embedding quality and embedding algorithms," *Mach. Learn.*, vol. 77, no. 1, pp. 1–25, Oct. 2009.
[42] G. A. F. Seber, *Multivariate Observations*. Hoboken, NJ: Wiley, 2004.

Hong Qiao (SM'06) received the B.Eng. degree in hydraulics and control and the M.Eng. degree in robotics from Xian Jiaotong University, Xian, China; the M.Phil. degree in robotics control from the University of Strathclyde, Strathclyde, U.K.; and the Ph.D. degree in robotics and artificial intelligence from De Montfort University, Leicester, U.K., in 1995.

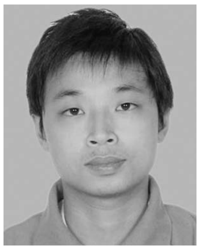
From 1995 to 1997, she was a University Research Fellow with De Montfort University. She was a Research Assistant Professor from 1997 to 2000 and an Assistant Professor from 2000 to 2002 with the Department of Manufacturing Engineering and Engineering Management, City University of Hong Kong, Kowloon, Hong Kong. Since January 2002, she has been a Lecturer with the School of Informatics, University of Manchester, Manchester, U.K. Currently, she is also a Professor with the Laboratory of Complex Systems and Intelligent Science, Institute of Automation, Chinese Academy of Sciences, Beijing, China. She first proposed the concept of "the attractive region in strategy investigation," which has been successfully applied by herself in robot assembly, robot grasping, and part recognition. The work has been reported in Advanced Manufacturing Alert (Wiley, 1999). Her current research interests include information-based strategy investigation, robotics and intelligent agents, animation, machine learning (neural networks and support vector machines), and pattern recognition.

Dr. Qiao was a member of the Program Committee of the IEEE International Conference on Robotics and Automation from 2001 to 2004. She is currently an Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS PART B and of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING.



Peng Zhang (M'10) received the B.Sc. degree in pure mathematics from Shandong University, Jinan, China, in 2006 and the Ph.D. degree in applied mathematics from the Academy of Mathematics and System Science, Chinese Academy of Sciences, Beijing, China, in 2012.

He is currently an Assistant Research Fellow with the Data Center, National Disaster Reduction Center of China, Beijing. His current research interests include machine learning, pattern recognition, data mining, and their applications to analyzing and visualizing natural disaster data.



Di Wang received the B.Sc. degree from Shandong University, Jinan, China, in 2007. He is currently working toward the Ph.D. degree in applied mathematics in the Institute of Applied Mathematics, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China.

His current research interests are the theory and application of support vector machines.

Bo Zhang (M'10) received the B.Sc. degree in mathematics from Shandong University, Jinan, China, in 1983; the M.Sc. degree in mathematics from Xi'an Jiaotong University, Xi'an, China, in 1985; and the Ph.D. degree in applied mathematics from the University of Strathclyde, Strathclyde, U.K., in 1992.

From 1985 to 1988, he was a Lecturer with the Department of Mathematics, Xi'an Jiaotong University. He was a Postdoctoral Research Fellow with the Department of Mathematics, Keele University, Keele, U.K., from January 1992 to December 1994, and with the Department of Mathematical Sciences, Brunel University, Uxbridge, U.K., from January 1995 to October 1997. In November 1997, he joined the School of Mathematical and Informational Sciences, Coventry University, Coventry, U.K., as a Senior Lecturer, where he was promoted to Reader in Applied Mathematics in September 2000 and to Professor of Applied Mathematics in September 2003. Currently, he is a Professor with the Institute of Applied Mathematics, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China. His current research interests include direct and inverse scattering problems, radar and medical imaging, machine learning, and pattern recognition.

Dr. Zhang is currently an Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS PART B.