Tangent Hyperplane Kernel Principal Component Analysis for Denoising

Joon-Ku Im, Daniel W. Apley, and George C. Runger

***Abstract*— Kernel principal component analysis (KPCA) is a method widely used for denoising multivariate data. Using geometric arguments, we investigate why a projection operation inherent to all existing KPCA denoising algorithms can sometimes cause very poor denoising. Based on this, we propose a modification to the projection operation that remedies this problem and can be incorporated into any of the existing KPCA algorithms. Using toy examples and real datasets, we show that the proposed algorithm can substantially improve denoising performance and is more robust to misspecification of an important tuning parameter.**

***Index Terms*— Denoising, kernel, kernel principal component analysis (KPCA), preimage problem.**

I. Introduction

KERNEL principal component analysis (KPCA) is a nonlinear generalization of linear principal component analysis (PCA) and is widely used for denoising multivariate data to identify underlying patterns and for other purposes [1]. Let $\mathbf{x}$ be a multivariate observation vector having n components, and suppose the data consist of N such observations $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$. In KPCA, one defines a set of M features $\{\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_M(\mathbf{x})\}$, each of which is some appropriately chosen nonlinear function of $\mathbf{x}$, and forms a feature vector $\boldsymbol{\phi}(\mathbf{x}) = [\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_M(\mathbf{x})]'$. One then conducts PCA on the set of feature vectors $\{\boldsymbol{\phi}(\mathbf{x}_1), \boldsymbol{\phi}(\mathbf{x}_2), \ldots, \boldsymbol{\phi}(\mathbf{x}_N)\}$. If we consider $\mathbb{R}^n$ the "input space" in which $\mathbf{x}$ lies and $\mathbb{R}^M$ the "feature space" in which $\boldsymbol{\phi}(\mathbf{x})$ lies, we can view the feature map as $\boldsymbol{\phi}$: $\mathbb{R}^n \rightarrow \mathbb{R}^M$. In practice, the feature map is usually defined implicitly by some kernel function for computational reasons [2], [3].

Several different KPCA denoising methods have been proposed [2]–[9], which we review and contrast more thoroughly in Section II. Their commonality is that they all share a general framework that involves two primary operations, which we refer to as projection and preimage approximation. By projection, we mean that, to denoise an observation $\mathbf{x}$ via KPCA, one projects its feature vector $\boldsymbol{\phi}(\mathbf{x})$ onto the principal subspace in the feature space, as illustrated in Fig. 1. The principal subspace is defined as the span of the dominant eigenvectors of the $M \times M$ covariance matrix of $\{\boldsymbol{\phi}(\mathbf{x}_1), \boldsymbol{\phi}(\mathbf{x}_2), \ldots, \boldsymbol{\phi}(\mathbf{x}_N)\}$.

The projected feature vector, which we denote by $P\boldsymbol{\phi}(\mathbf{x})$, is a point in the feature space. In order to denoise $\mathbf{x}$, we must find a point $\hat{\mathbf{x}}$ in the input space that best corresponds to $P\boldsymbol{\phi}(\mathbf{x})$. The exact preimage of $P\boldsymbol{\phi}(\mathbf{x})$ (under the map $\boldsymbol{\phi}$) generally does not exist. That is, there generally exists no $\hat{\mathbf{x}}$ such that $\boldsymbol{\phi}(\hat{\mathbf{x}}) = P\boldsymbol{\phi}(\mathbf{x})$ exactly. This is the so-called preimage problem in KPCA denoising [3]. By preimage approximation, we mean that one must find a value $\hat{\mathbf{x}}$ that minimizes some appropriately defined measure of dissimilarity between $\hat{\mathbf{x}}$ [or $\boldsymbol{\phi}(\hat{\mathbf{x}})$] and $P\boldsymbol{\phi}(\mathbf{x})$.

All of the aforementioned KPCA denoising methods employ the same orthogonal projection operation. Their distinctions, which we discuss in more detail in Section II, lie in how they perform the preimage approximation. For example, the original method of Schölkopf *et al.* [3] finds the point $\hat{\mathbf{x}}$ that minimizes a dissimilarity measure defined as the Euclidean distance between $\boldsymbol{\phi}(\hat{\mathbf{x}})$ and $P\boldsymbol{\phi}(\mathbf{x})$. That is, they take, $\hat{\mathbf{x}} = \arg\min_{\mathbf{z}} ||\boldsymbol{\phi}(\mathbf{z}) - P\boldsymbol{\phi}(\mathbf{x})||^2$ as the denoised $\mathbf{x}$, where the norm is the standard Euclidean 2-norm. Fig. 2 illustrates the results of this method for a toy example with $N = 100$ observations of $n = 2$ variables, using a polynomial kernel of degree 2. Fig. 2(a) shows the original data, and Fig. 2(b) shows the denoised data using the method of Schölkopf *et al.* [3]. The salient feature of Fig. 2(b) is that the points around the middle of the curve (e.g., $\mathbf{x}_A$) are not denoised nearly as well as those near the ends of the curve (e.g., $\mathbf{x}_B$).

In this paper, we investigate this phenomenon and show that it is a consequence of the projection operation, as opposed to the preimage approximation. Consequently, this undesirable characteristic (much poorer denoising in some regions of the input space than in other regions) is inherent to all the aforementioned KPCA algorithms, and the solution to this problem that we propose can be used to enhance any of the these algorithms.

An intuitive explanation for the poor denoising in some regions is as follows. Consider the image of $\mathbb{R}^n$ under the map $\boldsymbol{\phi}$, which is the subset of the feature space for which each point has an exact preimage. This subset having exact preimages forms a manifold in $\mathbb{R}^M$, which we refer to as the full manifold. As operations in $\mathbb{R}^M$, we can view projection onto the principal subspace as taking the point $\boldsymbol{\phi}(\mathbf{x})$ away from the full manifold to obtain $P\boldsymbol{\phi}(\mathbf{x})$, and preimage approximation as taking the projection $P\boldsymbol{\phi}(\mathbf{x})$ back to some point on the full manifold that is as similar to $P\boldsymbol{\phi}(\mathbf{x})$ as possible, to obtain $\boldsymbol{\phi}(\hat{\mathbf{x}})$, or equivalently $\hat{\mathbf{x}}$. As we will show, how well a point

Manuscript received December 2, 2010; accepted December 28, 2011. Date of publication February 21, 2012; date of current version March 6, 2012. This work was supported in part by the National Science Foundation under Grant CMMI-0826081 and Grant CMMI-0825331.

J.-K. Im and D. W. Apley are with the Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL 60208 USA (e-mail: ijk@u.northwestern.edu; apley@northwestern.edu).

G. C. Runger is with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: george.runger@asu.edu).

Digital Object Identifier 10.1109/TNNLS.2012.2185950

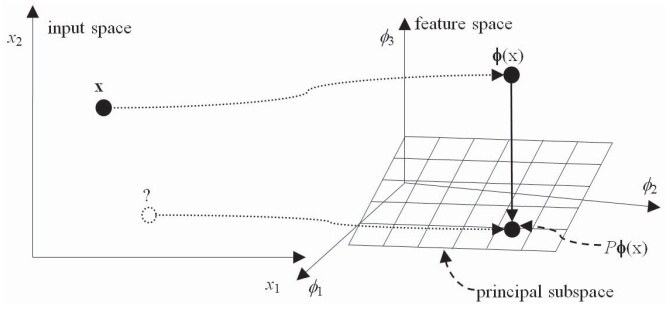


Fig. 1. Denoising an observation $\mathbf{x}$ involves two operations: projection of its feature vector $\boldsymbol{\phi}(\mathbf{x})$ onto the principal subspace and preimage approximation to find an $\hat{\mathbf{x}}$ that minimizes some measure of dissimilarity between $\hat{\mathbf{x}}$(or $\boldsymbol{\phi}(\hat{\mathbf{x}})$) and $P\boldsymbol{\phi}$ ($\mathbf{x}$).

$\mathbf{x}$ is denoised is strongly influenced by how far from the full manifold its projection $P\boldsymbol{\phi}(\mathbf{x})$ falls. For points such as $\mathbf{x}_A$ in Fig. 2, the projection step takes them quite far away from the full manifold. Hence, $P\boldsymbol{\phi}(\mathbf{x}_A)$ must be moved a large distance to get back to the full manifold in the preimage approximation step, which results in poor denoising. On the other hand, for points such as $\mathbf{x}_B$, the projection step keeps them close to the full manifold, which results in more effective denoising.

We propose a modification to the projection step which is designed to keep the "projected" feature vector closer to the full manifold, thereby improving denoising performance, especially at points such as $\mathbf{x}_A$ in Fig. 2, which are more difficult to denoise. We refer to the approach as the tangent hyperplane KPCA algorithm for reasons that will become apparent later. The modified projection can be used in conjunction with the preimage approximation algorithms of any of the aforementioned KPCA approaches.

The remainder of this paper is organized as follows. Section II briefly reviews the general KPCA approach and various preimage approximation algorithms. Section III provides a geometric explanation of why certain points such as $\mathbf{x}_A$ in Fig. 2 are denoised poorly using the regular projection step in KPCA. In Section IV, we introduce our tangent hyperplane KPCA approach as a solution to this problem. Section V presents denoising results for the toy example of Fig. 2 and for real image datasets, together with a discussion of the characteristics of the approach. Section VI concludes this paper.

II. Review of Prior KPCA Work and the Preimage Problem

Regular (linear) PCA on the n-dimensional data $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ finds the set of orthonormal basis vectors along which the components of $\mathbf{x}$ have largest variance. These basis vectors are the eigenvectors of the $n \times n$ (sample) covariance matrix of the data [10]–[12]. KPCA is PCA on the M-dimensional feature vectors, $\{\boldsymbol{\phi}(\mathbf{x}_1), \boldsymbol{\phi}(\mathbf{x}_2), \ldots, \boldsymbol{\phi}(\mathbf{x}_N)\}$, where $M >> n$ usually. Define the centered feature vector as $\tilde{\boldsymbol{\phi}}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x}) - \bar{\boldsymbol{\phi}}$ where $\bar{\boldsymbol{\phi}} = N^{-1} \sum_{j=1}^{N} \boldsymbol{\phi}(\mathbf{x}_j)$, the feature matrix as

$$\tilde{\boldsymbol{\Phi}} = [\,\tilde{\boldsymbol{\phi}}(\mathbf{x}_1)\ \tilde{\boldsymbol{\phi}}(\mathbf{x}_2) \cdots \tilde{\boldsymbol{\phi}}(\mathbf{x}_N)]'$$

and the covariance matrix of the feature vectors as

$$\mathbf{S}_{\boldsymbol{\Phi}} = N^{-1} \sum_{j=1}^{N} \tilde{\boldsymbol{\phi}}(\mathbf{x}_j)\tilde{\boldsymbol{\phi}}'(\mathbf{x}_j) = N^{-1}\tilde{\boldsymbol{\Phi}}'\tilde{\boldsymbol{\Phi}}.$$

KPCA therefore involves the eigenvalues and eigenvectors of the $M \times M$ matrix $\mathbf{S}_{\boldsymbol{\Phi}}$.

Given that direct calculation of the eigenvectors of $\mathbf{S}_{\boldsymbol{\Phi}}$ is usually computationally infeasible (M is often large), the novel enabling aspect of KPCA is the kernel trick, by which we can solve this eigenproblem without actually computing the high-dimensional features or their covariance matrix $\mathbf{S}_{\boldsymbol{\Phi}}$. This is possible because in KPCA the feature map $\boldsymbol{\phi}$ is defined implicitly via its inner products $\langle\boldsymbol{\phi}(\mathbf{x}),\boldsymbol{\phi}(\mathbf{y})\rangle = \boldsymbol{\phi}'(\mathbf{x})\boldsymbol{\phi}(\mathbf{y}) = K(\mathbf{x},\mathbf{y})$ for some appropriately chosen positive definite kernel function $K(\bullet,\bullet)$ (see [2], [3]; for comprehensive treatments of kernel functions, see [13]), and the inner products are all that are required in the computations. Common choices for the kernel are the polynomial kernel $K(\mathbf{x},\mathbf{y}) = (1 + \mathbf{x}'\mathbf{y})^d$ of degree d, and the Gaussian radial basis function (RBF) kernel $K(\mathbf{x},\mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/\rho)$ where the parameters d and ρ are chosen by the users. A polynomial kernel of degree 2 was used in the Fig. 2 example.

More specifically, define the centered kernel function

$$\begin{aligned}
\tilde{K}(\mathbf{x},\mathbf{y}) &= \langle\tilde{\boldsymbol{\phi}}(\mathbf{x}), \tilde{\boldsymbol{\phi}}(\mathbf{y})\rangle = \langle\boldsymbol{\phi}(\mathbf{x}) - \bar{\boldsymbol{\phi}}, \boldsymbol{\phi}(\mathbf{y}) - \bar{\boldsymbol{\phi}}\rangle \\
&= \langle\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{y})\rangle - \langle\boldsymbol{\phi}(\mathbf{x}), \bar{\boldsymbol{\phi}}\rangle - \langle\boldsymbol{\phi}(\mathbf{y}), \bar{\boldsymbol{\phi}}\rangle + \langle\bar{\boldsymbol{\phi}}, \bar{\boldsymbol{\phi}}\rangle \\
&= K(\mathbf{x},\mathbf{y}) - N^{-1}\sum_{i=1}^{N} K(\mathbf{x},\mathbf{x}_i) - N^{-1}\sum_{i=1}^{N} K(\mathbf{y},\mathbf{x}_i) \\
&\quad + N^{-2}\sum_{i=1}^{N}\sum_{j=1}^{N} K(\mathbf{x}_i,\mathbf{x}_j) \\
&= K(\mathbf{x},\mathbf{y}) - N^{-1}\,\mathbf{1}'\,\mathbf{K}_{\mathbf{x}} - N^{-1}\,\mathbf{1}'\,\mathbf{K}_{\mathbf{y}} + N^{-2}\,\mathbf{1}'\mathbf{K}\mathbf{1}
\end{aligned}$$

where $\mathbf{1}$ denotes a column vector of ones, and $\mathbf{K}_{\mathbf{x}} = [K(\mathbf{x},\mathbf{x}_1),\ K(\mathbf{x},\mathbf{x}_2), \ldots, K(\mathbf{x},\mathbf{x}_N)]'$. Also define the centered kernel matrix $\tilde{\mathbf{K}} = \tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{\Phi}}'$, whose $(i,\ j)$th element is $\tilde{K}_{ij} = \tilde{K}(\mathbf{x}_i,\mathbf{x}_j)$. Note that $\tilde{\mathbf{K}}$ is an $N \times N$ matrix that can be calculated via evaluations of the kernel function. This is computationally significant, because the eigenvector/eigenvalue pairs $\{(\mathbf{v}_k,\ \lambda_k)\colon\ k = 1,\ 2,\ \ldots,\ L_0\}$ of $\mathbf{S}_{\boldsymbol{\Phi}}$ with nonzero eigenvalue (we refer to such an eigenvector as a nonzero eigenvector, and assume there exist L_0 of them) and the nonzero eigenvector/eigenvalue pairs $\{(\boldsymbol{\alpha}_k = [\alpha_{k,1},\ \alpha_{k,2},\ \ldots,\ \alpha_{k,N}]',\ \eta_k)\colon\ k = 1, 2, \ldots, L_0\}$ of $\tilde{\mathbf{K}}$ are closely related (see [1] for details): $\mathbf{v}_k = \tilde{\boldsymbol{\Phi}}'\boldsymbol{\alpha}_k = \sum_{i=1}^{N} \tilde{\boldsymbol{\Phi}}(\mathbf{x}_i)\,\alpha_{k,i}$, and $\eta_k = N\lambda_k$, where the convention for scaling the eigenvectors is to take $||\mathbf{v}_k|| = 1$, which implies $||\boldsymbol{\alpha}_k|| = (N\lambda_k)^{-1/2} = \eta_k^{-1/2}$. Using this, one can solve the eigenproblem for the $M \times M$ matrix $\mathbf{S}_{\boldsymbol{\Phi}}$ by solving the much less computationally expensive eigenproblem for the $N \times N$ matrix $\tilde{\mathbf{K}}$ ($N << M$ typically).

As discussed in the introduction, the existing KPCA denoising methods share a common projection operation. That is, given an observation $\mathbf{x}$, they find (implicitly) the orthogonal projection

$$P\tilde{\boldsymbol{\Phi}}(\mathbf{x}) = \sum_{k=1}^{L} \langle\tilde{\boldsymbol{\phi}}(\mathbf{x}), \mathbf{v}_k\rangle\mathbf{v}_k$$

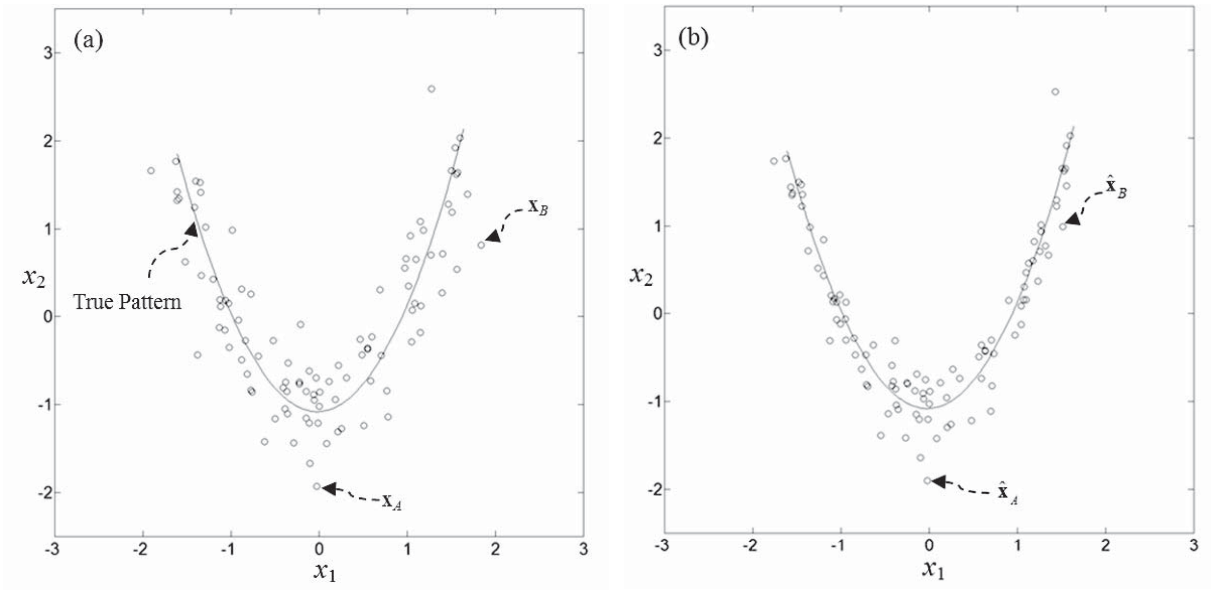


Fig. 2. (a) Original data (white markers) distributed approximately along a quadratic curve (solid curve). (b) Denoised data (white markers) obtained using the method of [3]. Points such as $\mathbf{x}_A$ near the middle of the curve are denoised less effectively than points such as $\mathbf{x}_B$ near the ends of the curve.

of $\tilde{\boldsymbol{\phi}}(\mathbf{x})$ onto the principal subspace spanned by the L principal eigenvectors $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_L\}$ of $\mathbf{S}_{\boldsymbol{\Phi}}$. This is followed by the preimage approximation for finding the $\hat{\mathbf{x}}$ that is most similar to $P\tilde{\boldsymbol{\phi}}(\mathbf{x})$, which is where the KPCA denoising methods mentioned in the introduction differ. Specifically, they differ in how they define the dissimilarity measure and/or how they implement the optimization algorithm to minimize it, including the variables over which they optimize.

In [2] and [3], the authors suggested minimization of the Euclidean distance

$$||\tilde{\boldsymbol{\phi}}(\mathbf{z}) - P\tilde{\boldsymbol{\phi}}(\mathbf{x})||^2 \tag{1}$$

over $\mathbf{z}$. In other words, the dissimilarity measure between $\hat{\mathbf{x}}$ (or $\tilde{\boldsymbol{\phi}}(\hat{\mathbf{x}})$) and $P\tilde{\boldsymbol{\phi}}(\mathbf{x})$ is the Euclidean distance between $\tilde{\boldsymbol{\phi}}(\hat{\mathbf{x}})$ and $P\tilde{\boldsymbol{\phi}}(\mathbf{x})$ in the feature space. The computational kernel trick that is central to all KPCA methods is based on the fact that for any $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{z} \in \mathbb{R}^n$, and nonzero eigenvector $\mathbf{v}_k$, we can evaluate the following inner products via the kernel:

$$\begin{aligned}\langle \tilde{\boldsymbol{\phi}}(\mathbf{x}), \mathbf{v}_k \rangle &= \left\langle \tilde{\boldsymbol{\phi}}(\mathbf{x}), \sum_{i=1}^{N} \tilde{\boldsymbol{\phi}}(\mathbf{x}_i)\, \alpha_{k,i} \right\rangle \\ &= \sum_{i=1}^{N} \langle \tilde{\boldsymbol{\phi}}(\mathbf{x}), \tilde{\boldsymbol{\phi}}(\mathbf{x}_i) \rangle\, \alpha_{k,i} = \tilde{\mathbf{K}}_{\mathbf{x}}' \boldsymbol{\alpha}_k\end{aligned}$$

and

$$\begin{aligned}\langle \tilde{\boldsymbol{\phi}}(\mathbf{z}), P\tilde{\boldsymbol{\phi}}(\mathbf{x}) \rangle &= \left\langle \tilde{\boldsymbol{\phi}}(\mathbf{z}), \sum_{k=1}^{L} \langle \tilde{\boldsymbol{\phi}}(\mathbf{x}), \mathbf{v}_k \rangle \mathbf{v}_k \right\rangle \\ &= \sum_{k=1}^{L} \langle \tilde{\boldsymbol{\phi}}(\mathbf{z}), \mathbf{v}_k \rangle \langle \tilde{\boldsymbol{\phi}}(\mathbf{x}), \mathbf{v}_k \rangle \\ &= \sum_{k=1}^{L} (\tilde{\mathbf{K}}_{\mathbf{z}}' \boldsymbol{\alpha}_k)(\tilde{\mathbf{K}}_{\mathbf{x}}' \boldsymbol{\alpha}_k) \\ &= \tilde{\mathbf{K}}_{\mathbf{z}}' [\, \boldsymbol{\alpha}_1 \cdots \boldsymbol{\alpha}_L \,][\, \boldsymbol{\alpha}_1 \cdots \boldsymbol{\alpha}_L \,]' \tilde{\mathbf{K}}_{\mathbf{x}}\end{aligned}$$

where $\tilde{\mathbf{K}}_{\mathbf{x}} = [\tilde{K}(\mathbf{x}, \mathbf{x}_1),\ \tilde{K}(\mathbf{x}, \mathbf{x}_2),\ \cdots,\ \tilde{K}(\mathbf{x}, \mathbf{x}_N)]'$. Thus, the quantity to be minimized (with respect to $\mathbf{z}$, for a given $\mathbf{x}$ to be denoised) in the preimage approximation of [3] can be written as

$$\begin{aligned}&||\tilde{\boldsymbol{\phi}}(\mathbf{z}) - P\tilde{\boldsymbol{\phi}}(\mathbf{x})||^2 \\ &\quad = \langle \tilde{\boldsymbol{\phi}}(\mathbf{z}), \tilde{\boldsymbol{\phi}}(\mathbf{z}) \rangle - 2\langle \tilde{\boldsymbol{\phi}}(\mathbf{z}), P\tilde{\boldsymbol{\phi}}(\mathbf{x}) \rangle + \langle P\tilde{\boldsymbol{\phi}}(\mathbf{x}), P\tilde{\boldsymbol{\phi}}(\mathbf{x}) \rangle \\ &\quad = \tilde{K}(\mathbf{z}, \mathbf{z}) - 2\boldsymbol{\beta}_{\mathbf{z}}' \boldsymbol{\beta}_{\mathbf{x}} + \boldsymbol{\beta}_{\mathbf{x}}' \boldsymbol{\beta}_{\mathbf{x}}\end{aligned}$$

where

$$\boldsymbol{\beta}_{\mathbf{x}} = \left[\, \mathbf{v}_1\ \mathbf{v}_2 \cdots \mathbf{v}_L \,\right]' \tilde{\boldsymbol{\phi}}(\mathbf{x}) = \left[\, \boldsymbol{\alpha}_1\ \boldsymbol{\alpha}_2 \cdots \boldsymbol{\alpha}_L \,\right]' \tilde{\mathbf{K}}_{\mathbf{x}}$$

is the vector of principal components scores of $\tilde{\boldsymbol{\phi}}(\mathbf{x})$.

In [2] and [3], the authors suggested using a fixed-point iterative algorithm, related to a gradient descent algorithm, to minimize (1) with respect to $\mathbf{z}$ over the entire input space $\mathbb{R}^n$. Takahashi and Kurita [4] modified the fixed-point iterative algorithm by updating certain weighting coefficients at each iteration, based on updating the feature vector to be projected onto the principal subspace. A fixed point of their algorithm is more consistent with minimizing $||\tilde{\boldsymbol{\phi}}(\mathbf{z}) - P\tilde{\boldsymbol{\phi}}(\mathbf{z})||^2$. Teixeira *et al.* [5] proposed an alternative initial guess for the fixed-point iterative algorithm.

While the preceding methods share a common dissimilarity measure (1), others use different dissimilarity measures. Kwok and Tsang [6] proposed a method akin to multidimensional scaling. To denoise $\mathbf{x}$, they found $\hat{\mathbf{x}}$ so that the distances $||\mathbf{x}_i - \hat{\mathbf{x}}||$ $(i = 1, 2, \ldots, n)$ are as close as possible to the distances $||\tilde{\boldsymbol{\phi}}(\mathbf{x}_i) - P\tilde{\boldsymbol{\phi}}(\mathbf{x})||^2$ $(i = 1, 2, \ldots, n)$. Their dissimilarity measure is the distance between the vector of $||\mathbf{x}_i - \hat{\mathbf{x}}||^2$'s and the vector of $||\tilde{\boldsymbol{\phi}}(\mathbf{x}_i) - P\tilde{\boldsymbol{\phi}}(\mathbf{x})||^2$'s for values of i such that $\tilde{\boldsymbol{\phi}}(\mathbf{x}_i)$ is a neighbor of $P\tilde{\boldsymbol{\phi}}(\mathbf{x})$, and to reduce computational expense they restricted $\hat{\mathbf{x}}$ to being a linear combination of the nearest neighbors. Bakır *et al.* [7] suggested a kernel regression method to estimate the map $\Gamma(P\tilde{\boldsymbol{\phi}}(\mathbf{x})) = [\Gamma_1(P\tilde{\boldsymbol{\phi}}(\mathbf{x})), \Gamma_2(P\tilde{\boldsymbol{\phi}}(\mathbf{x})), \ldots, \Gamma_n(P\tilde{\boldsymbol{\phi}}(\mathbf{x}))]'$,

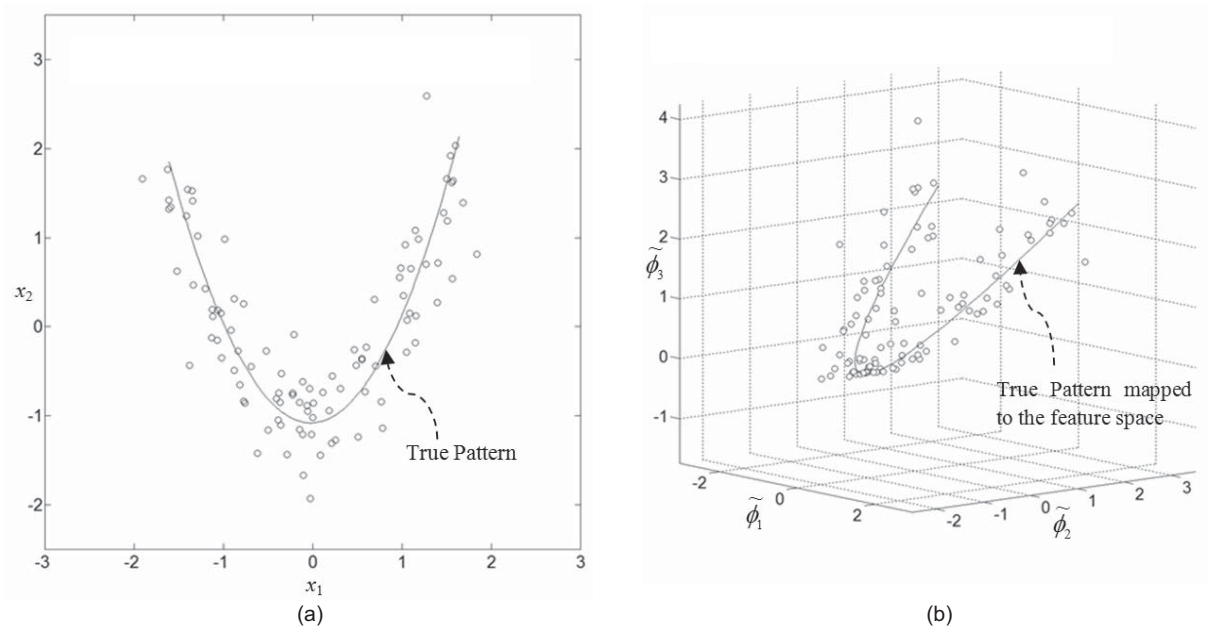


Fig. 3. (a) Data with noise in the input space. The solid curve is the true pattern that the data would follow with no noise. (b) Centered feature vectors and true pattern mapped to the feature space.

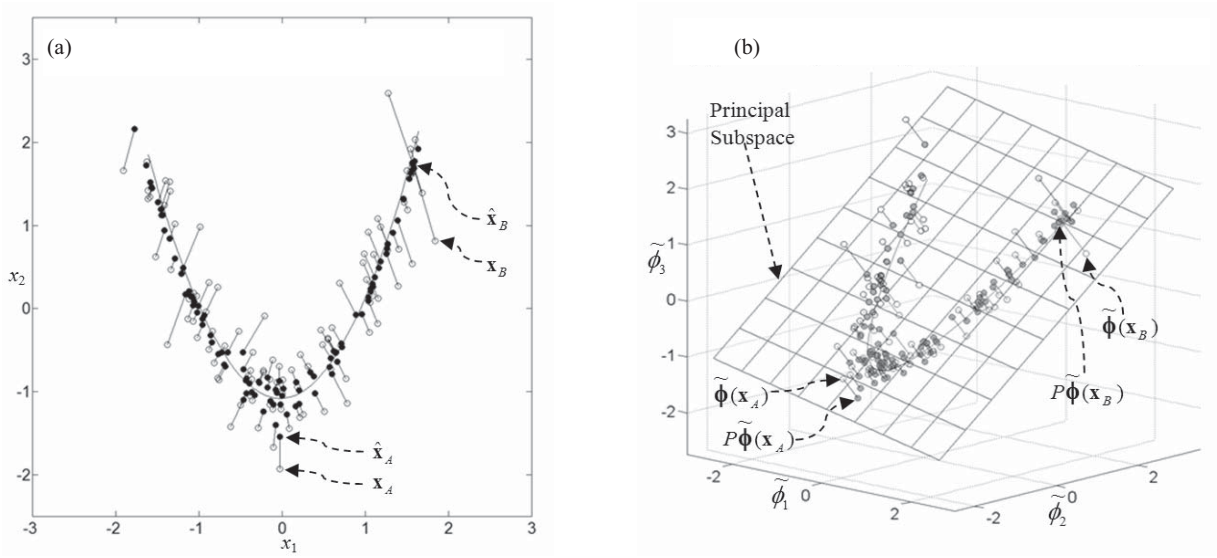


Fig. 4. (a) Result of [3] in the input space. Approximate preimages (black markers). (b) Result of [3] in the feature space. Projections (gray marker) of the features of the noisy data (white marker) using the simple 3-D feature and the method of [3]. The approximate preimages in (a) are supposed to best correspond to the projections in (b). We can observe that $\mathbf{x}_B$ is denoised much more than $\mathbf{x}_A$.

where one views the components of $P\tilde{\boldsymbol{\phi}}(\mathbf{x})$ as the predictor variables and the components of $\hat{\mathbf{x}}$ as the response variables. They assume the kernel regression model $\Gamma_j(P\tilde{\boldsymbol{\phi}}(\mathbf{x})) = \sum_{i=1}^{N} \beta_i^j \kappa(P\tilde{\boldsymbol{\phi}}(\mathbf{x}), P\tilde{\boldsymbol{\phi}}(\mathbf{x}_i))$, where κ is a different kernel operator, and minimize the sum of squared regression residuals $\sum_{i=1}^{N} ||\mathbf{x}_i - \Gamma(P\tilde{\boldsymbol{\phi}}(\mathbf{x}_i))||^2$ with respect to the β_i^j's. Their approach applies mainly to the situation in which noiseless observations are available for training. Zheng *et al.* [8] noted that a denoised observation of $\mathbf{x}$ obtained by most of the preceding methods is a weighted sum of $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$. That is, $\hat{\mathbf{x}} = \sum_{i=1}^{N} w_i \mathbf{x}_i$ for some weights $w_1, \ldots, w_N$. Hence, they used a dissimilarity measure $||\tilde{\boldsymbol{\phi}}(\sum_{i=1}^{N} w_i \mathbf{x}_i) - P\tilde{\boldsymbol{\phi}}(\mathbf{x})||^2$, and chose the weights to minimize this under certain constraints.

Nguyen and De la Torre [9] approached the problem in a different way. They proposed to solve $\hat{\mathbf{x}} = \arg\min_{\mathbf{z}} ||\tilde{\boldsymbol{\phi}}(\mathbf{z}) - P\tilde{\boldsymbol{\phi}}(\mathbf{z})||^2 + C \cdot d(\mathbf{z}, \mathbf{x})$ where $d(\mathbf{z},\mathbf{x})$ is any distance measure between $\mathbf{z}$ and $\mathbf{x}$, and C is a weight that is chosen to balance between the two terms. Although this method slightly deviates from the general framework we have portrayed, it is similar in the sense that it still involves an orthogonal projection onto the principal subspace.

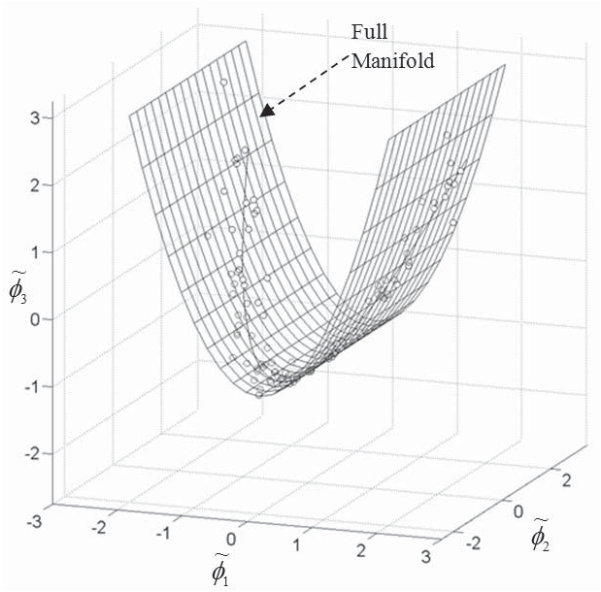


Fig. 5. Feature vectors for the data (open dots), true pattern (solid curve), and full manifold (surface) from the toy example, in the feature space. All feature vectors must reside exactly in the full manifold.

III. Geometric Interpretation of KPCA Denoising Performance

In this section, we provide a geometric interpretation of the inconsistent denoising performance seen in Fig. 2 (i.e., points such as $\mathbf{x}_A$ are denoised less effectively than points such as $\mathbf{x}_B$) and relate it to the orthogonal projection procedure common to all of the KPCA methods. We argue that the orthogonal projection has undesirable properties and that modifying this procedure (as described in the next section) can substantially improve denoising performance.

To illustrate, consider again the toy example introduced in Section I, for which the data are distributed along a quadratic curve in $\mathbb{R}^2$ with Gaussian noise added. In order to better visualize the feature space (to facilitate the illustration), instead of using a feature map induced by a polynomial kernel, we use the simple feature map $\boldsymbol{\phi}(\mathbf{x}) = (x_1, x_2, x_1^2)'$ which is sufficiently rich to capture the quadratic pattern. Fig. 3(a) shows the noisy data and the true quadratic pattern in the input space, and Fig. 3(b) shows the same data and true pattern mapped to the 3-D (centered) feature space. The data points are the feature vectors, and the solid curve is the image of the true pattern under the feature map. Ideally, we would like the denoised observation to lie close to the solid curve.

Consider the orthogonal projection $P\tilde{\boldsymbol{\phi}}(\mathbf{x})$ of a feature vector onto the principal subspace. For the toy example with $L = 2$, the projections of the feature vectors are plotted in Fig. 4(b) and their approximate preimages using the method of [3] are plotted in Fig. 4(a). As with linear PCA, we would hope that the projections in the feature space are close to the true pattern. However, from Fig. 4(b), the projections of points near the middle of the curve are much farther from the true pattern than are the projections of points near both ends of the curve. This is consistent with the variable denoising performance that we see in Fig. 4(a) and discussed in the context of Fig. 2.

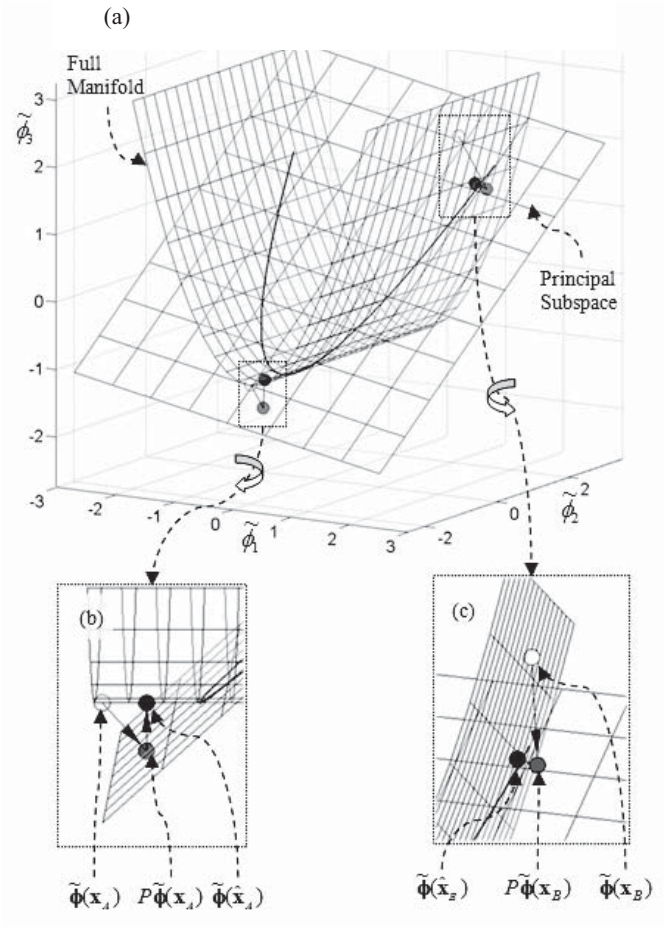


Fig. 6. (a) Method of [3] in the feature space. Depiction in the feature space of the principal projection $P\tilde{\boldsymbol{\phi}}(\mathbf{x})$ and preimage projection $\tilde{\boldsymbol{\phi}}(\hat{\mathbf{x}})$ for points $\mathbf{x}_A$ and $\mathbf{x}_B$, together with the full manifold and principal subspace. (b) Expanded callout for $\mathbf{x}_A$. (c) Expanded callout for $\mathbf{x}_B$.

The reason behind the poor denoising of $\mathbf{x}_A$ and the good denoising of $\mathbf{x}_B$ becomes more apparent when we visualize the full manifold and the projection and preimage approximation steps in the feature space. Recall that the full manifold $\tilde{\boldsymbol{\phi}}(\mathbb{R}^n)$ is the image of $\mathbb{R}^n$ under the feature map, which is an n-dimensional manifold in M-dimensional space. Fig. 5 plots the full manifold for the toy example. Notice that the feature vector for any data point must lie in this full manifold, regardless of the level of noise. More generally, the feature vector for any point in $\mathbb{R}^n$ must lie on the full manifold, and the full manifold is the set of all points in the feature space that have exact preimages. Consequently, after finding the projection $P\tilde{\boldsymbol{\phi}}(\mathbf{x})$ onto the principal subspace, we can view $\tilde{\boldsymbol{\phi}}(\hat{\mathbf{x}})$ from the preimage approximation $\hat{\mathbf{x}} = \arg\min_{\mathbf{z}} ||\tilde{\boldsymbol{\phi}}(\mathbf{z}) - P\tilde{\boldsymbol{\phi}}(\mathbf{x})||^2$ of [3] as an orthogonal projection of $P\tilde{\boldsymbol{\phi}}(\mathbf{x})$ onto the full manifold in the feature space. To distinguish between these two feature space projections, we refer to $P\tilde{\boldsymbol{\phi}}(\mathbf{x})$ as the principal projection and $\tilde{\boldsymbol{\phi}}(\hat{\mathbf{x}})$ as the preimage projection.

Fig. 6 shows the principal projections and preimage projections for points $\mathbf{x}_A$ and $\mathbf{x}_B$, together with the full manifold and the principal subspace. The reason why $\mathbf{x}_A$ is denoised less effectively than $\mathbf{x}_B$ has much more to do with the principal projection than the preimage projection. Because of the local

geometry around $\tilde{\boldsymbol{\phi}}(\mathbf{x}_A)$ of the full manifold and the principal subspace [see Fig. 6(b)], the principal projection $P\tilde{\boldsymbol{\phi}}(\mathbf{x}_A)$ for the point $\mathbf{x}_A$ falls quite far away from the full manifold. That is to say, it is quite far from having a preimage. Hence, in order to get back to the full manifold when finding the preimage projection, we must choose a point $\tilde{\boldsymbol{\phi}}(\hat{\mathbf{x}}_A)$ that is quite far from the principal subspace. In turn, this means that $\tilde{\boldsymbol{\phi}}(\hat{\mathbf{x}}_A)$ is quite far from the image of the true pattern, which means that the denoised point $\hat{\mathbf{x}}_A$ is quite far from the true pattern (i.e., poor denoising).

The situation for the point $\mathbf{x}_B$ is very different. Because of the local geometry around $\tilde{\boldsymbol{\phi}}(\mathbf{x}_B)$ [see Fig. 6(c)], the principal projection $P\tilde{\boldsymbol{\phi}}(\mathbf{x}_B)$ remains relatively close to the full manifold, so that the preimage projection $\tilde{\boldsymbol{\phi}}(\hat{\mathbf{x}}_B)$ remains close to the principal subspace. The result is that $\hat{\mathbf{x}}_B$ is close to the true pattern (i.e., good denoising).

IV. Tangent Hyperplane Method for KPCA Denoising

The arguments in the previous section indicate that the denoising performance is strongly influenced by the projection step. The geometric interpretation in Fig. 6 motivates our proposed modification to the projection step, which can be incorporated into the preimage approximation criteria for any of the KPCA denoising algorithms discussed previously. For each point $\mathbf{x}$ to be denoised, instead of projecting orthogonally onto the principal subspace, we incorporate information on the local geometry of the full manifold in the neighborhood of $\tilde{\boldsymbol{\phi}}(\mathbf{x})$, so that the principal projection remains closer to the full manifold. This is accomplished as follows.

First note that any point in the principal subspace can be represented as $P\tilde{\boldsymbol{\phi}}(\mathbf{x}) + \sum_{k=1}^{L} b_k \mathbf{v}_k$ for some $\mathbf{b} = [b_1, b_2, \ldots, b_L]' \in \mathbb{R}^L$. For any $\mathbf{x}$, we define the tangent hyperplane for $\tilde{\boldsymbol{\phi}}(\mathbf{x})$ as the n-dimensional hyperplane (in the M-dimensional feature space) that is tangent to the full manifold at $\tilde{\boldsymbol{\phi}}(\mathbf{x})$. More precisely, the tangent hyperplane for $\tilde{\boldsymbol{\phi}}(\mathbf{x})$ is the set of all points $\tilde{\boldsymbol{\phi}}(\mathbf{x}) + \sum_{j=1}^{n} a_j \mathbf{u}_j$ for $\mathbf{a} = [a_1, a_2, \ldots, a_n]' \in \mathbb{R}^n$, where

$$\mathbf{u}_j = \frac{\partial \tilde{\boldsymbol{\phi}}(\mathbf{x})}{\partial x_j} = \left[\frac{\partial \tilde{\boldsymbol{\phi}}_1(\mathbf{x})}{\partial x_j} \;\; \frac{\partial \tilde{\boldsymbol{\phi}}_2(\mathbf{x})}{\partial x_j} \;\; \cdots \;\; \frac{\partial \tilde{\phi}_M(\mathbf{x})}{\partial x_j} \right]'.$$

In the following, we refer to the quantity that will replace $P\tilde{\boldsymbol{\phi}}(\mathbf{x})$ in the projection step as the principal projection, although technically it is not a projection. Rather, the principal projection in our algorithm is defined as the point on the principal subspace that is closest to the tangent hyperplane for $\tilde{\boldsymbol{\phi}}(\mathbf{x})$. This is equivalent to finding the minimum distance between the two subspaces (the tangent hyperplane and the principal subspace). As in [14], which computes the minimum distance between two hyperplanes in a different context, the solution is obtained by finding the coefficient vectors $\mathbf{a}$ and $\mathbf{b}$ to minimize the distance between

$$P\tilde{\boldsymbol{\phi}}(\mathbf{x}) + \sum_{k=1}^{L} b_k \mathbf{v}_k \text{ and } \tilde{\boldsymbol{\phi}}(\mathbf{x}) + \sum_{j=1}^{n} a_j \mathbf{u}_j.$$

Then we take the principal projection to be

$$P_T \tilde{\boldsymbol{\phi}}(\mathbf{x}) = P\tilde{\boldsymbol{\phi}}(\mathbf{x}) + \sum_{k=1}^{L} b_k^* \mathbf{v}_k \tag{2}$$

where $\mathbf{a}^* = [a_1^*, a_2^*, \ldots, a_n^*]'$ and $\mathbf{b}^* = [b_1^*, b_2^*, \ldots, b_L^*]'$ denote the coefficient vectors that minimize the distance. The subscript T on the projection operator indicates it is for the tangent hyperplane method.

The solution is obtained by defining $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n]$ and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_L]$, the sets of basis vectors for the tangent hyperplane and for the principal subspace, respectively. The optimal coefficient vector minimizes

$$\begin{aligned} &\left\| \left(\tilde{\boldsymbol{\phi}}(\mathbf{x}) + \mathbf{U}\mathbf{a} \right) - \left(P\tilde{\boldsymbol{\phi}}(\mathbf{x}) + \mathbf{V}\mathbf{b} \right) \right\|^2 \\ &= \left\| \left(\tilde{\boldsymbol{\phi}}(\mathbf{x}) - P\tilde{\boldsymbol{\phi}}(\mathbf{x}) \right) + \left[\mathbf{U}, \, -\mathbf{V} \right] \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} \right\|^2 \\ &= \left\| \left(\tilde{\boldsymbol{\phi}}(\mathbf{x}) - P\tilde{\boldsymbol{\phi}}(\mathbf{x}) \right) + \mathbf{A}\mathbf{c} \right\|^2 \end{aligned}$$

with respect to $\mathbf{c} = [\mathbf{a}', \mathbf{b}']'$, and $\mathbf{A} = [\mathbf{U}, -\mathbf{V}]$, the solution to which is the standard linear least squares solution.

Because the solution $\mathbf{c}^*$ may not be unique (if the columns of $\mathbf{A}$ are linearly dependent) or may be poorly conditioned (if the columns of $\mathbf{A}$ are nearly linearly dependent), we add a regularization penalty and choose $\mathbf{c}^*$ so that

$$\mathbf{c}^* = \arg\min_{\mathbf{c}} ||(\tilde{\boldsymbol{\phi}}(\mathbf{x}) - P\tilde{\boldsymbol{\phi}}(\mathbf{x})) + \mathbf{A}\mathbf{c}||^2 + \delta||\mathbf{c}||^2 \tag{3}$$

for some small user-selected δ as in [14]. In Section V, we compare the results with different levels of δ and discuss its appropriate choice. The regularization penalty has other intuitive appeal. It penalizes solutions for which the tangent hyperplane point $\tilde{\boldsymbol{\phi}}(\mathbf{x}) + \sum_{j=1}^{n} a_j \mathbf{u}_j$ is further away from $\tilde{\boldsymbol{\phi}}(\mathbf{x})$ and/or the principal subspace point $P\tilde{\boldsymbol{\phi}}(\mathbf{x}) + \sum_{k=1}^{L} b_k \mathbf{v}_k$ is further away from $P\tilde{\boldsymbol{\phi}}(\mathbf{x})$. Hence, in the extreme case of $\delta \to \infty$, our principal projection coincides with the standard principal projection $P\tilde{\boldsymbol{\phi}}(\mathbf{x})$. Moreover, the regularization penalty keeps the solution well defined in the case where $\mathbf{A}$ is singular. For example, in the special case of linear PCA [which corresponds to the feature map $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}$], the tangent hyperplane is all of $\mathbb{R}^n$, so that the distance is zero between any point on the principal subspace and the tangent hyperplane. Including the regularization penalty ensures that the solution is unique and coincides with the standard linear PCA solution in this case.

It is straightforward to show that the solution to (3) is

$$\mathbf{c}^* = (\mathbf{A}'\mathbf{A} + \delta\mathbf{I})^{-1}\mathbf{A}'(P\tilde{\boldsymbol{\phi}}(\mathbf{x}) - \tilde{\boldsymbol{\phi}}(\mathbf{x})) \tag{4}$$

from which we obtain our principal projection as defined in (2). One can then use the preimage approximation criterion from any of the existing KPCA denoising methods to find the approximate preimage of $P_T\tilde{\boldsymbol{\phi}}(\mathbf{x})$, which serves as the denoised point $\hat{\mathbf{x}}$.

It is important to note that the computational kernel trick can be used in all steps of the algorithm. Hence, no calculations need to be performed in the M-dimensional feature space. The penalized least squares solution for the principal projection $P_T\tilde{\boldsymbol{\phi}}(\mathbf{x})$ requires only inner products in the feature space,

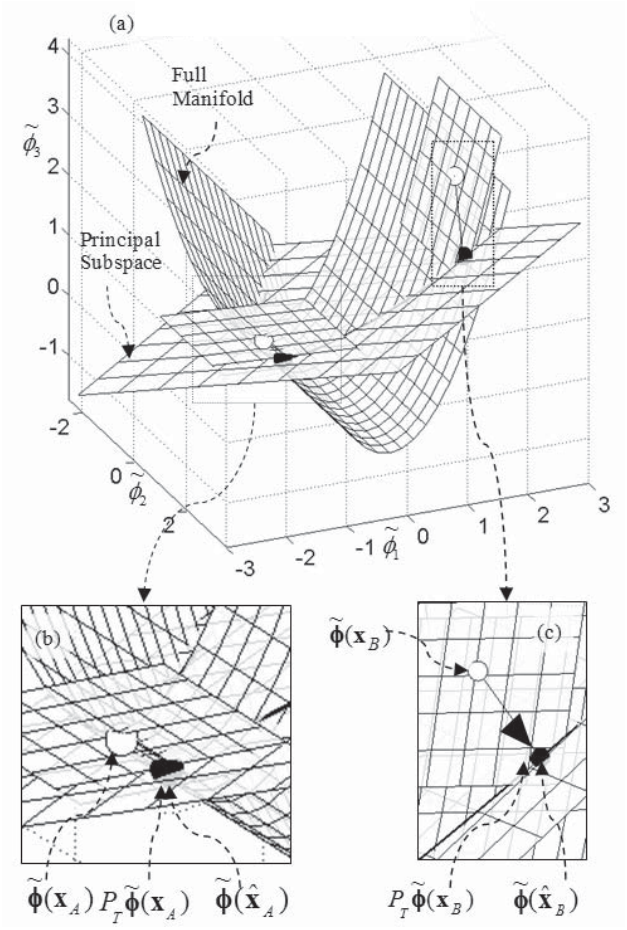


Fig. 7. Proposed method in the feature space. Geometric depiction of the tangent hyperplane method for denoising points $\mathbf{x}_A$ and $\mathbf{x}_B$. Forcing the principal projection $P_T\,\tilde{\boldsymbol{\phi}}(\mathbf{x})$ to be close to the tangent hyperplane also keeps it close to the full manifold and close to having an exact preimage. Notice that because the preimage projections $\tilde{\boldsymbol{\phi}}(\hat{\mathbf{x}}_A)$ and $\tilde{\boldsymbol{\phi}}(\hat{\mathbf{x}}_B)$ are so close to their respective principal projections $P_T\,\tilde{\boldsymbol{\phi}}(\hat{\mathbf{x}}_A)$ and $P_T\,\tilde{\boldsymbol{\phi}}(\hat{\mathbf{x}}_B)$, it is difficult to visually distinguish them in the figure. (b), (c) Expanded callouts of the two regions in (a).

and the Appendix shows how all these can be calculated via the kernel trick. Because the principal projection is a linear combination of the eigenvectors, the kernel trick can be applied as usual in any of the preimage approximation approaches.

Regarding the computational complexity, because the tangent hyperplane method simply modifies the projection step of existing KPCA methods, its computational expense is the same as that of the existing methods but with the additional expense of the modified projection step. The most computationally expensive part of the modified projection step is the inversion of $(n+L)\times(n+L)$ matrix $\mathbf{A}'\mathbf{A}+\delta\mathbf{I}$. This is generally much less expensive than the remainder of the KPCA algorithm (especially if one uses a matrix inversion algorithm that takes advantage of the orthogonality of the columns of $\mathbf{A}$ that are associated with $\mathbf{V}$). Hence, the modified projection step of the tangent hyperplane method does not substantially increase the computational expense of the entire KPCA algorithm.

In Fig. 7, we illustrate the tangent hyperplane method for the same toy example with 3-D feature space. The preimage approximation is still that of [3]. Relative to the denoising results using the conventional orthogonal projection (Figs. 4 and 6), $\mathbf{x}_A$ is denoised substantially better, and $\mathbf{x}_B$ is denoised slightly better. In fact, $\mathbf{x}_A$ is now denoised as well as $\mathbf{x}_B$. The reason for the dramatic improvement for the denoising of points such as $\mathbf{x}_A$ is illustrated geometrically in Fig. 7(a), which also shows the tangent hyperplanes for the two points $\tilde{\boldsymbol{\phi}}(\mathbf{x}_A)$ and $\tilde{\boldsymbol{\phi}}(\mathbf{x}_B)$. The approach forces the principal projection to be close to the tangent hyperplane [compare $P_T\tilde{\boldsymbol{\phi}}(\mathbf{x}_A)$ in Fig. 7(b) with $P\tilde{\boldsymbol{\phi}}(\mathbf{x}_A)$ in Fig. 6(b)], which also keeps it close to the full manifold. If the principal projection is close to the full manifold, then it is close to having an exact preimage, so that its preimage projection $\tilde{\boldsymbol{\phi}}(\hat{\mathbf{x}})$ is more likely to remain close to the principal subspace. The result is that the denoised point is more likely to be close to the true pattern.

V. Results and Discussion

In this section, we apply the proposed tangent hyperplane method (using the preimage approximation criterion of [3]) to three different datasets: the toy data that we have used to illustrate the approach and two different image datasets in which each multivariate observation represents a different image. For the toy dataset, we use a feature map consisting of all polynomial terms of degree ≤ 2, which corresponds to the kernel $K(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}'\mathbf{y})^2$. For the image datasets, we used the Gaussian RBF kernel $K(\mathbf{x},\mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/\rho)$ with $\rho = nc$, where c equals 0.5 times the average of component variances, which we chose because it is more appropriate than the polynomial kernels on both methods. We compare the tangent hyperplane approach with that of [3]. As we have discussed, the tangent hyperplane approach is a modification of the principal projection step that can be used with any of the preimage approximation criteria. For the preimage approximation, we used the quasi-Newton BFGS algorithm to minimize $||\tilde{\boldsymbol{\phi}}(\mathbf{z}) - P\tilde{\boldsymbol{\phi}}(\mathbf{x})||^2$ or $||\tilde{\boldsymbol{\phi}}(\mathbf{z}) - P_T\tilde{\boldsymbol{\phi}}(\mathbf{x})||^2$.

A. Results for the Toy Example

The data for the toy example were generated by standardizing a simulated training set of size $N = 100$ bivariate observations of the random vector $\mathbf{x} = [x_1, x_2]' = [t, t^2]' + [w_1, w_2]'$, where t follows a uniform distribution over the interval $[-1, 1]$, and w_1 and w_2 are independent Gaussian random variables with zero mean and standard deviation 0.1. We can view the quadratic curve $\{[t, t^2]' : t \in [-1, 1]\}$ as the true pattern, and each observation $\mathbf{x}$ as a random point along the true pattern plus Gaussian noise. Fig. 8(a) shows the denoising result using the method of [3], which differs slightly from Fig. 4(a) because of slightly different choice of feature map: $\boldsymbol{\phi}(\mathbf{x}) = (x_1, x_2, x_1^2)'$ in Fig. 4(a) versus the full quadratic feature map $\boldsymbol{\phi}(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)'$ in Fig. 8(a). Fig. 8(b) shows the denoising result using the proposed tangent hyperplane method. For both methods, $L = 4$ principal eigenvectors were retained. $L = 4$ provided the best denoising for both methods (results for other L are omitted for brevity).

Comparing Fig. 8(a) and (b), it is clear that for this example the modified principal projection of the tangent hyperplane method provides much better denoising than the conventional orthogonal principal projection. We also conducted simulations for a modified toy example with higher dimensional data.

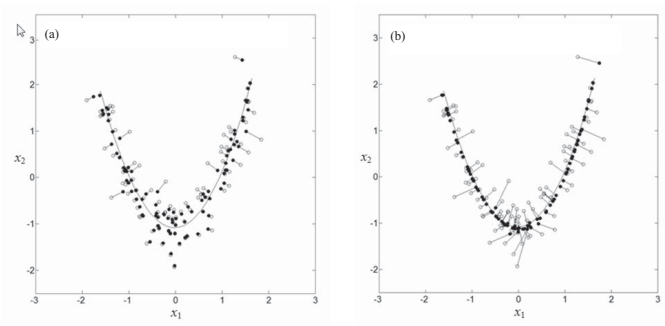


Fig. 8. For the toy example, a comparison of the denoising results using (a) the method of [3] and (b) the tangent hyperplane method, which has substantially better performance.

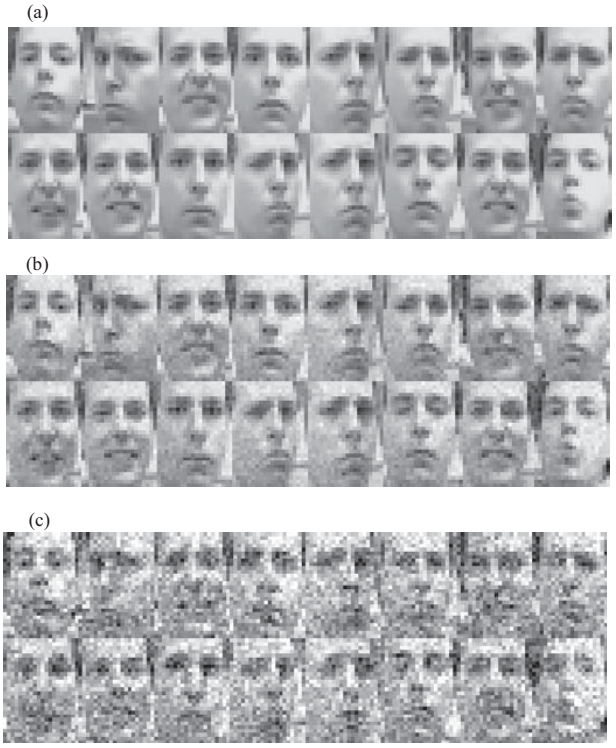


Fig. 9. Examples of (a) clean images, (b) images with low noise, and (c) images with high noise. The noisy ones are the clean ones with Gaussian noise added with different standard deviations (b) 0.04 and (c) 0.12.

The true pattern was a 1-D quadratic curve in a 2-D subspace of $\mathbb{R}^{20}$, rotated so that it had components along all each natural basis vector. The denoising results for the higher dimensional example were quite similar to those shown in Fig. 8, especially the relative performance of the two methods. For brevity, we omit the results for this example and instead focus on the higher dimensional example in the following section.

B. Results for the Image Examples

One of the two image datasets was obtained from the website http://cs.nyu.edu/~roweis/data.html. Each image is of the same person's face but with different facial expressions and angles. The elements of each original multivariate observation $\mathbf{x}$ are the grayscale levels (over the range [0, 1]) for $n = 28 \times 20 = 560$ pixels. The data to be denoised were contaminated versions of the images for which we added independent

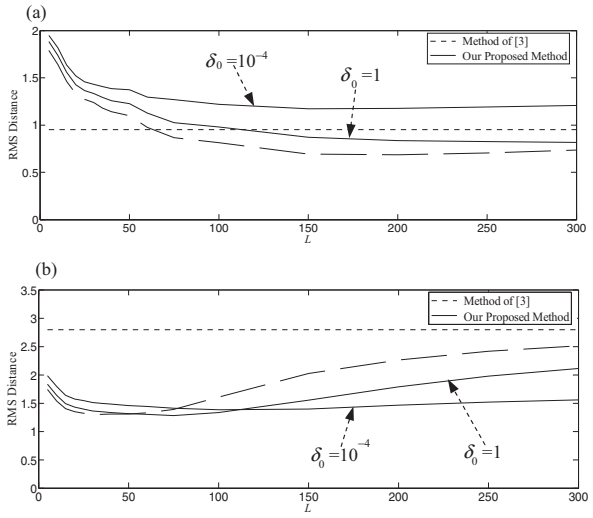


Fig. 10. RMS distance for the method of [3] (dashed) and the tangent hyperplane method (solid), as a function of L for (a) the images with low noise and (b) the images with high noise levels. For the low noise, the method of [3] shows much better performance than the tangent hyperplane method. For high noise, the tangent hyperplane method with $\delta_0 = 1$ has the best optimal performance for this example, and the tangent hyperplane method with both settings is much more robust to misspecification of L.

Gaussian noise at two different noise levels. We standardized the data before applying the denoising methods. The training data consisted of $N = 500$ noisy images. Examples of the original clean images and the noisy images for the two different noise levels are shown in Fig. 9.

As the denoising performance depends on choice of L (the number of retained principal components) and δ [the regularization parameter in (3)], we investigated the effects of these parameters. Fig. 10 plots, as a function of L, the root mean square (RMS) distance

$$\left(N^{-1} \sum_{i=1}^{N} ||\mathbf{x}_{i,\text{true}} - \hat{\mathbf{x}}_i||^2 \right)^{\frac{1}{2}}$$

between the denoised images ($\hat{\mathbf{x}}_i$) and the original clean images ($\mathbf{x}_{i,\text{true}}$) over the set of N training images. After generating the noisy images, the original clean images were used only for evaluation purposes but not within the algorithms in any way.

Fig. 10(a) shows the denoising results for the images with low noise. The dotted horizontal line is the RMS distance between the noisy images and the original clean images, representing the performance with no denoising. The dashed curve is the RMS distance for the method of [3], and the two solid curves are the RMS distance for the proposed tangent hyperplane method with two different values for δ, which we take to be δ_0 times the maximum eigenvalue of $\mathbf{A}'\mathbf{A}$ for $\delta_0 = 1$ and 10^{-4}. Fig. 10(b) is analogous to Fig. 10(a) but for the images with high noise. Notice that the method of [3] corresponds to the tangent hyperplane method with $\delta_0 = \infty$.

A number of notable characteristics can be observed from Fig. 10. First, when the noise level is low, the method of [3] performs better, in terms of RMS distance, than the tangent

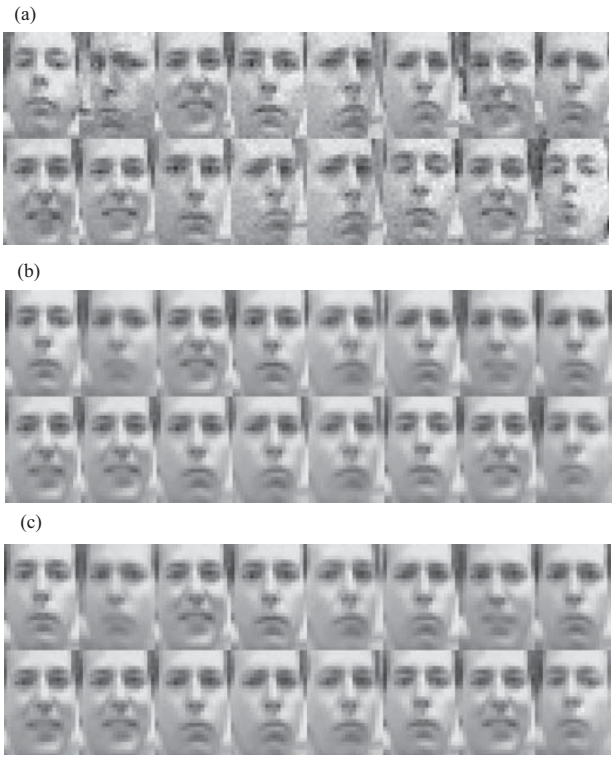


Fig. 11. Sample of the denoised image result for the low noise case using (a) method of [3] with $L = 200$, and (b) and (c) our method with $L = 300$, $\delta_0 = 1$ and $L = 150$, $\delta_0 = 10^{-4}$, respectively. Although RMS distance corresponding to (c) is high, the denoised images are quite clean and retain key characteristics/patterns in the original data.

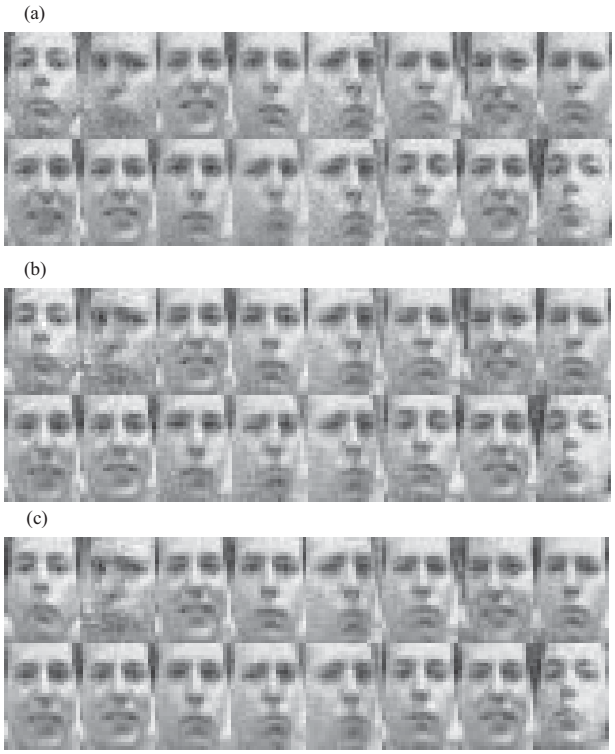


Fig. 12. Sample of the denoised image result for the high noise case using (a) method of [3] with $L = 30$, and (b) and (c) our method with $L = 75$, $\delta_0 = 1$ and $L = 100$, $\delta_0 = 10^{-4}$, respectively. Our method under both settings shows better denoising performance overall, which is consistent with what we observe from Fig. 10(b).

hyperplane method with finite δ_0. However, the RMS distance measure does not necessarily capture all relevant aspects of performance. For example, consider Fig. 11, which visually compares the denoised images for the two approaches for a sample of 16 images. For each method, L was chosen to be optimal, i.e., $L = 200$ was used for the method of [3], $L = 300$ for the tangent hyperplane method with $\delta_0 = 1$, and $L = 150$ for the tangent hyperplane method with $\delta_0 = 10^{-4}$. Although the tangent hyperplane method results in worse RMS distance, it clearly provides more denoising and produces images that appear cleaner than the images produced by [3]. The upside of this is that the facial expressions are generally cleaner in Fig. 11(b) and (c), than in Fig. 11(a). But the downside is that, in addition to removing more of the noise, the tangent hyperplane method has removed some of the real patterns in the images. For example, although some of the facial expression patterns are retained in Fig. 11(b) and (c), others are not retained, and the pattern resulting from the subject looking to his right has been "denoised" out of the data [as apparent from the second image in the top rows of Fig. 11(a)–(c)]. This is perhaps because the subject's rotation to the far right is something of an outlier expression.

The results are quite different when the noise level is higher. Fig. 10(b) shows that, in this case, the tangent hyperplane method with $\delta_0 = 1$ has the best optimal performance according the RMS distance measure, although the optimal performances of the other two methods are not far behind. This can also be seen in Fig. 12, which again compares results for optimal choices of L per each method for the same 16 samples shown in Fig. 11. However, quite significantly, our method is much more robust to the choice of L. For any L between roughly 50 and 200 for $\delta_0 = 10^{-4}$, our method performs nearly as well as for its optimal L. In contrast, the "sweet spot" for the method of [3] is much smaller, with performance rapidly degrading for L outside the range 15–75, roughly. This robustness to choice of L is important, because in practice it is difficult to determine the optimal L. When using the polynomial kernel, the same trend was observed but the overall performances of all methods were not as good. Notice that plots such as Fig. 10 cannot be used in practice, because the plotted RMS distances were calculated using the true noiseless images, which are not available in practice.

The second image dataset, which are also of faces but of a different nature than in the previous example, was obtained from the website http://isomap.stanford.edu. As described in [15], the dataset consists of a sequence of 4096-dimensional vectors, representing the grayscale levels (over the range [0, 1]) for $n = 64 \times 64 = 4096$ pixels. Each image is an animated face with different poses and lighting directions. The data to be denoised were again contaminated versions of the images for which we added independent Gaussian noise at two different noise levels. As the data dimension (4096) is quite high, for dimensionality reduction purpose we preprocessed the data using PCA before applying the denoising methods. The training data consisted of $N = 698$ noisy images. Examples of the original clean images and the noisy ones are shown in Fig. 13.

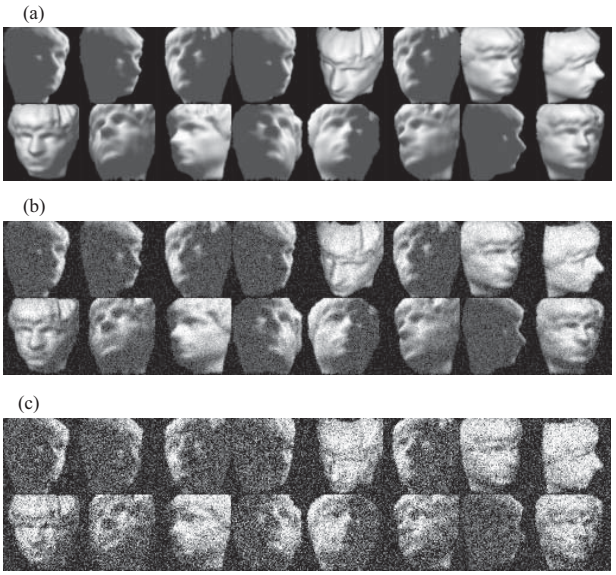


Fig. 13. Examples of (a) clean, (b) and (c) noisy image data. The noisy ones are the clean ones with Gaussian noise added with standard deviations (b) 0.04 and (c) 0.12.

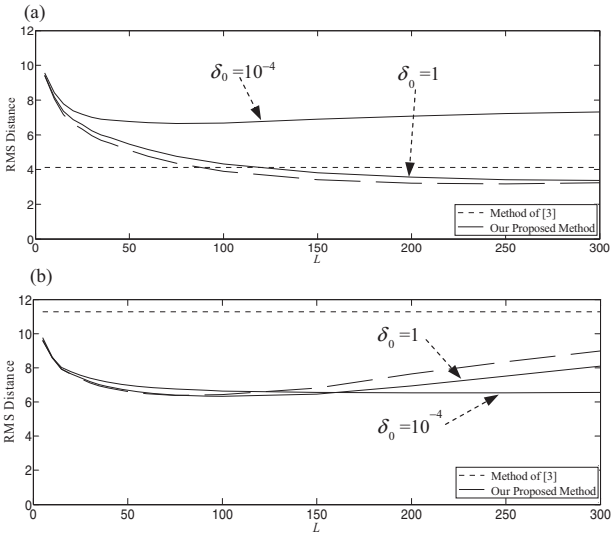


Fig. 14. RMS distance for the method of [3] (dashed) and the tangent hyperplane method (solid), as a function of L for both (a) low noise and (b) high noise cases. As in the previous example, the method of [3] is better in the low noise case while our method better in high noise case and robust to the choice of L.

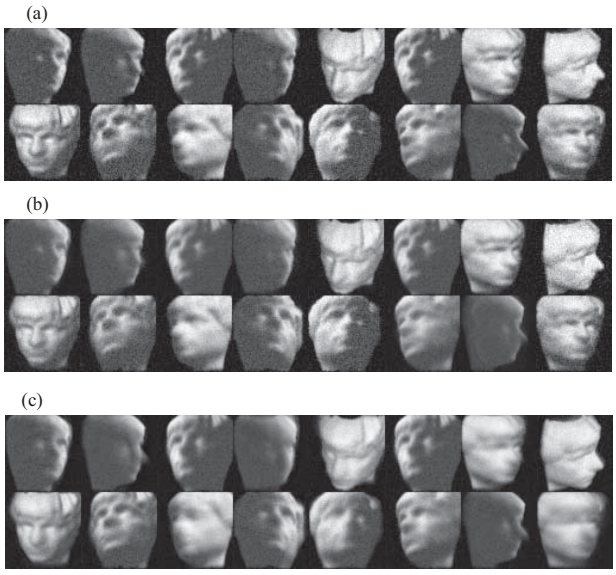


Fig. 15. Sample of the denoised image result for the low noise case using (a) method of [3] with $L = 250$, and (b) and (c) our method with $L = 300$, $\delta_0 = 1$ and $L = 75$, $\delta_0 = 10^{-4}$, respectively.

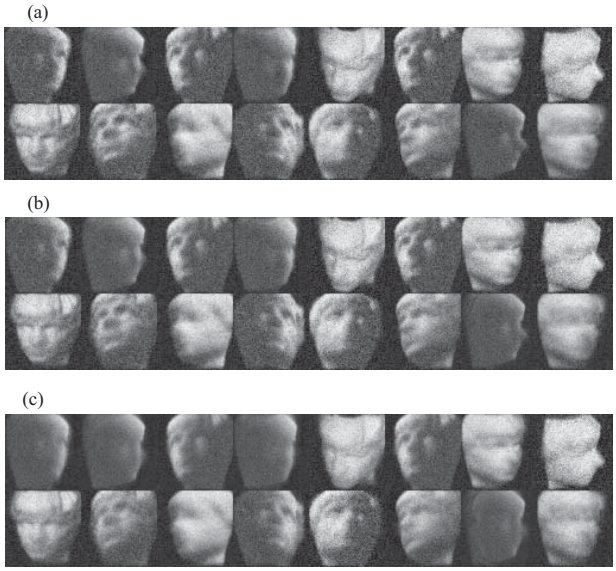


Fig. 16. Sample of the denoised image result for the high noise case using (a) method of [3] with $L = 75$, and (b) and (c) our method with $L = 100$, $\delta_0 = 1$ and $L = 200$, $\delta_0 = 10^{-4}$, respectively.

As in the previous example, we applied both the method of [3] and the tangent hyperplane method with $\delta_0 = 1$ and $\delta_0 = 10^{-4}$. Figs. 14–16, which are analogous to Figs. 10–12, show the results. The key observations are quite consistent with the previous example. In terms of RMS distance, our method is outperformed by the method of [3] when the noise level is low. The tangent hyperplane method appears to result in greater "denoising," but of legitimate patterns as well as noise. At high noise level, on the other hand, the tangent hyperplane method outperforms the method of [3] in terms of RMS distances as well as robustness to L.

We also performed simulations for medium noise cases and for $\delta_0 = 10^{-2}$. Because the results were predictable, falling somewhere between the low and high noise cases and the $\delta_0 = 1$ and $\delta_0 = 10^{-4}$ cases shown previously, we omit them for brevity.

C. Choice of the Parameter δ and Kernels

From the previous experimental results, it can be observed that δ substantially influences the denoising results. Although we envision no clear strategy for optimally choosing δ, we do recommend that it be chosen to at least accomplish its intended purpose: namely, to prevent erratic results caused by poor conditioning of $\mathbf{A}$ when its columns are (nearly) linearly dependent. As a rule of thumb, we suggest choosing $\delta = 10^{-4}$ times the maximum eigenvalue of $\mathbf{A}'\mathbf{A}$. One could also

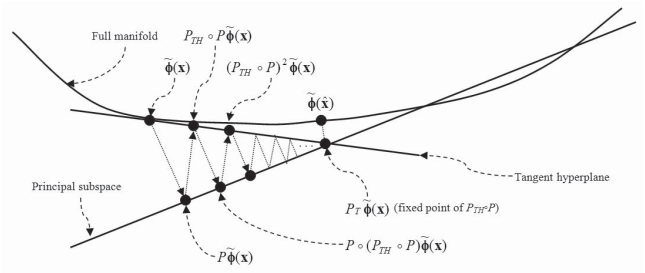


Fig. 17. Interpretation of the tangent hyperplane method as a series of paired projections.

conceivably use plots of some characteristic of the denoised images (e.g., $\sum_{i=1}^{N} ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2$) versus δ and look for the elbow, as is commonly done in ridge regression.

Our modification of the principal projection in the tangent hyperplane method was motivated by the geometry of the full manifold and the principal subspace in the feature space. Although we illustrated this motivation with a polynomial kernel, based on the facial images examples and other simulation results that are not shown, we have observed that the relative improvement in denoising is as significant when the feature map is that for a Gaussian RBF kernel.

D. Interpretation as a Series of Projections

In this section, we provide an alternative interpretation of the tangent hyperplane method. For the ridge penalty term $\delta = 0$, taking $P_T\tilde{\boldsymbol{\phi}}(\mathbf{x})$ as the principal projection in the tangent hyperplane method can be interpreted as a series of paired projections (the first onto the principal subspace and the second onto the tangent hyperplane) until convergence, followed by an additional projection onto the principal subspace. We illustrate this in Fig. 17 and provide an informal proof in the following paragraph. In Fig. 17, P_{TH} and P denote the projection operators onto the tangent hyperplane and the principal subspace, respectively. Hence, the tangent hyperplane approach is like applying the method of Schölkopf *et al.* [3] iteratively with the preimage projection onto the full manifold replaced by the projection onto the tangent hyperplane. The final step of projecting back onto the full manifold is the same for both methods. We discuss this iterative scheme only for interpretation purposes. It should not be used to implement the approach, because we have a closed-form expression for the solution $P_T\tilde{\boldsymbol{\phi}}(\mathbf{x})$.

To justify this interpretation, it is easy to show that, with no ridge penalty, any solution $P_T\tilde{\boldsymbol{\phi}}(\mathbf{x})$ to the least squares problem (3) (regardless of whether it is unique) is a fixed point of the map $P_{TH}{\circ}P$ (by the orthogonality characterization of minimum norm problems). Likewise, it is easy to show that any fixed point of the map $P_{TH}{\circ}P$ is a solution to the least squares problem (also by the orthogonality characterization of minimum norm problems). Furthermore, if the solution to the least squares problem is unique, then repeated application of $P_{TH}{\circ}P$ converges to a unique fixed point (which is the same as the unique least squares solution). This last statement follows because if the least squares solution is unique, then the vectors $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$ and $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_L\}$ are linearly independent. Hence, no point in $\tilde{\boldsymbol{\phi}}(\mathbf{x}) + \text{span}\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$ lies exactly in $\text{span}\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_L\}$ and vice versa (unless they intersect at a unique point). It therefore follows that for any two points $\mathbf{u}$ and $\tilde{\mathbf{u}}$ in $\tilde{\boldsymbol{\phi}}(\mathbf{x}) + \text{span}\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$, we have $||P_{TH}{\circ}P{\circ}(\mathbf{u}-\tilde{\mathbf{u}})|| < ||\mathbf{u}-\tilde{\mathbf{u}}||$. Hence, $P_{TH}{\circ}P$ is a contraction map in this subspace, and the statement follows because contraction maps have unique fixed points.

VI. Conclusion

We have argued that all existing KPCA denoising methods share a general framework that involves two operations—an orthogonal projection (onto the principal subspace in the feature space) and a preimage approximation (moving the projection $P\tilde{\boldsymbol{\phi}}(\mathbf{x})$ to the full manifold in the feature space). Moreover, they all share a common orthogonal projection procedure, and their distinctions lie entirely in the preimage approximation procedure.

Using geometrical arguments, we have shown that the orthogonal projection step can result in variable denoising performance, by which points in some regions are denoised effectively, and in other regions poorly. We proposed, as a remedy for this shortcoming, a modification to the orthogonal projection step that we termed the tangent hyperplane method. Essentially, the method attempts to project onto the principal subspace in a manner that keeps the principal projection as close to the full manifold as possible. In other words, it helps to ensure that the principal projection is close to having a preimage, which helps to ensure that the approximate preimage is close to the underlying pattern. Because the tangent hyperplane method is a modification to only the projection step, it can be used in conjunction with the preimage approximation step of any of the existing KPCA denoising algorithms.

We compared the performance of the tangent hyperplane method and the method of [3] for three datasets: a toy example in which the data represent a quadratic pattern buried in noise; and two different image data sets under different noise levels and ridge parameter choices. In the toy example, the tangent hyperplane method resulted in remarkably better denoising (see Fig. 8). For the image data examples with high noise levels, the tangent hyperplane method performed better than the method of [3] using the optimal L for each method, and it was also substantially more robust to misspecification of L. Specifically, the performance of the tangent hyperplane method was nearly optimal for a wide range of L. For the image data examples with low noise levels, the method of [3] performed better than the tangent hyperplane method with finite δ (they are the same for infinite δ) in terms of RMS distance. However, the tangent hyperplane method appears to result in more extreme denoising (of noise as well as of legitimate patterns that one wishes to retain).

Appendix

Using the Kernel Trick to Calculate the Principal Projection $P_T\tilde{\boldsymbol{\phi}}(\mathbf{x})$

Here we show how the kernel trick is used to calculate (4) and implicitly calculate the principal projection (2) for the tangent hyperplane method described in Section IV.

First note that we can express the matrix $\mathbf{A}$ via the feature space inner products

$$\mathbf{A}'\mathbf{A} = \begin{bmatrix} \mathbf{u}_1' \\ \vdots \\ \mathbf{u}_n' \\ -\mathbf{v}_1' \\ \vdots \\ -\mathbf{v}_L' \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \cdots \mathbf{u}_n & -\mathbf{v}_1 \cdots -\mathbf{v}_L \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{u}_1'\mathbf{u}_1 & \cdots & \mathbf{u}_1'\mathbf{u}_n & -\mathbf{u}_1'\mathbf{v}_1 & \cdots & -\mathbf{u}_1'\mathbf{v}_L \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{u}_n'\mathbf{u}_1 & \cdots & \mathbf{u}_n'\mathbf{u}_n & -\mathbf{u}_n'\mathbf{v}_1 & \cdots & -\mathbf{u}_n'\mathbf{v}_L \\ -\mathbf{v}_1'\mathbf{u}_1 & \cdots & -\mathbf{v}_1'\mathbf{u}_n & \mathbf{v}_1'\mathbf{v}_1 & \cdots & \mathbf{v}_1'\mathbf{v}_L \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -\mathbf{v}_L'\mathbf{u}_1 & \cdots & -\mathbf{v}_L'\mathbf{u}_n & \mathbf{v}_L'\mathbf{v}_1 & \cdots & \mathbf{v}_L'\mathbf{v}_L \end{bmatrix}.$$

Notice that the $\mathbf{v}_i$'s are standardized by definition. For numerical stability of the least squares solution, we will later standardize the $\mathbf{u}_i$'s by dividing ith column and row of $\mathbf{A}'\mathbf{A}$ by $\sqrt{\mathbf{u}_i'\mathbf{u}_i}$ for $i = 1, 2, \ldots, n$ (and rescale the elements of $\mathbf{A}$ accordingly). However, for ease of presentation, here we assume they are not yet standardized.

We consider separately each of the four submatrices of $\mathbf{A}'\mathbf{A}$ indicated above. The lower-right submatrix (of size $L \times L$) is the identity matrix, because the $\mathbf{v}_i$'s are defined as an orthonormal set of eigenvectors of $\mathbf{S}_\Phi$. The upper-right submatrix (of size $n \times L$) requires that we evaluate terms of the form $\mathbf{u}_i{}'\mathbf{v}_j$. Recall that for a specific $\mathbf{x}$ to be denoised, $\mathbf{u}_i$ is defined as

$$\mathbf{u}_i = \left.\frac{\partial \tilde{\boldsymbol{\phi}}(\mathbf{y})}{\partial y_i}\right|_{\mathbf{y}=\mathbf{x}} = \left.\left[\begin{array}{cccc} \frac{\partial \tilde{\phi}_1(\mathbf{y})}{\partial y_i} & \frac{\partial \tilde{\phi}_2(\mathbf{y})}{\partial y_i} & \cdots & \frac{\partial \tilde{\phi}_M(\mathbf{y})}{\partial y_i} \end{array}\right]'\right|_{\mathbf{y}=\mathbf{x}}.$$

Combining this with $\mathbf{v}_j = \tilde{\boldsymbol{\Phi}}'\boldsymbol{\alpha}_j$, we have

$$\begin{aligned} \mathbf{u}_i'\mathbf{v}_j &= \left.\frac{\partial \tilde{\boldsymbol{\phi}}(\mathbf{y})'}{\partial y_i}\right|_{\mathbf{y}=\mathbf{x}} [\,\tilde{\boldsymbol{\phi}}(\mathbf{x}_1)\ \tilde{\boldsymbol{\phi}}(\mathbf{x}_2) \cdots \tilde{\boldsymbol{\phi}}(\mathbf{x}_N)\,]\boldsymbol{\alpha}_j \\ &= \left.\left[\begin{array}{cccc} \frac{\partial \tilde{K}(\mathbf{y},\mathbf{x}_1)}{\partial y_i} & \frac{\partial \tilde{K}(\mathbf{y},\mathbf{x}_2)}{\partial y_i} & \cdots & \frac{\partial \tilde{K}(\mathbf{y},\mathbf{x}_N)}{\partial y_i} \end{array}\right]\boldsymbol{\alpha}_j\right|_{\mathbf{y}=\mathbf{x}} \end{aligned}$$

which can be evaluated by differentiating the kernel function. The lower-left submatrix is the transpose of the upper-right submatrix, by the symmetry of $\mathbf{A}'\mathbf{A}$.

For the upper-left matrix (of size $n \times n$), we have

$$\begin{aligned} \mathbf{u}_i'\mathbf{u}_j &= \left.\frac{\partial \tilde{\boldsymbol{\phi}}(\mathbf{y})'}{\partial y_i}\frac{\partial \tilde{\boldsymbol{\phi}}(\mathbf{z})}{\partial z_j}\right|_{\mathbf{y}=\mathbf{z}=\mathbf{x}} \\ &= \left.\frac{\partial^2 \tilde{\boldsymbol{\phi}}(\mathbf{y})'\tilde{\boldsymbol{\phi}}(\mathbf{z})}{\partial y_i \partial z_j}\right|_{\mathbf{y}=\mathbf{z}=\mathbf{x}} = \left.\frac{\partial^2 K(\mathbf{y},\mathbf{z})}{\partial y_i \partial z_j}\right|_{\mathbf{y}=\mathbf{z}=\mathbf{x}} \end{aligned}$$

which can also be evaluated by differentiating the kernel function.

After inverting $\mathbf{A}'\mathbf{A} + \delta\mathbf{I}$, it remains to calculate the $(n+L)$-length vector $\mathbf{A}'(P\tilde{\boldsymbol{\phi}}(\mathbf{x}) - \tilde{\boldsymbol{\phi}}(\mathbf{x}))$ via the kernel. We note that its first n components are $\mathbf{u}_i'(P\tilde{\boldsymbol{\phi}}(\mathbf{x}) - \tilde{\boldsymbol{\phi}}(\mathbf{x}))$ for $i = 1, 2, \ldots, n$, and its last L components are $-\mathbf{v}_j'(P\tilde{\boldsymbol{\phi}}(\mathbf{x}) - \tilde{\boldsymbol{\phi}}(\mathbf{x}))$ for $j = 1, 2, \ldots, L$. The latter are zero, because $P\tilde{\boldsymbol{\phi}}(\mathbf{x}) - \tilde{\boldsymbol{\phi}}(\mathbf{x})$ is orthogonal to each eigenvector $\mathbf{v}_j$ for $j = 1, 2, \ldots, L$. The former are

$$\begin{aligned} \mathbf{u}_i'\left(P\tilde{\boldsymbol{\phi}}(\mathbf{x}) - \tilde{\boldsymbol{\phi}}(\mathbf{x})\right) &= \mathbf{u}_i'\left(\sum_{k=1}^{L}\left(\tilde{\mathbf{K}}_{\mathbf{x}}'\boldsymbol{\alpha}_k\right)\mathbf{v}_k - \tilde{\boldsymbol{\phi}}(\mathbf{x})\right) \\ &= \sum_{k=1}^{L}\left(\tilde{\mathbf{K}}_{\mathbf{x}}'\boldsymbol{\alpha}_k\right)\mathbf{u}_i'\mathbf{v}_k - \left.\frac{\partial \tilde{K}(\mathbf{y},\mathbf{x})}{\partial y_i}\right|_{\mathbf{y}=\mathbf{x}} \end{aligned}$$

in which the terms $\mathbf{u}_i{}'\mathbf{v}_k$ are evaluated as above, and the far-right term is evaluated by differentiating the kernel. Therefore, $\mathbf{c}^*$, and hence the principal projection $P_T\tilde{\boldsymbol{\phi}}(\mathbf{x})$, can be evaluated using the kernel trick.

ACKNOWLEDGMENT

The authors would to thank the three anonymous referees and the editor for their numerous helpful comments.

REFERENCES

[1] 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.

[2] S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rätsch, "Kernel PCA and de-noising in feature spaces," in *Advances in Neural Information Processing Systems 11*. Cambridge, MA: MIT Press, 1999, pp. 536–542.

[3] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. J. Smola, "Input space versus feature space in kernel-based methods," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1000–1017, Sep. 1999.

[4] T. Takahashi and T. Kurita, "Robust de-noising by kernel PCA," in *Proc. Int. Conf. Artif. Neural Netw.*, vol. 2415. 2002, pp. 739–744.

[5] A. R. Teixeira, A. M. Tomé, K. Stadlthanner, and E. W. Lang, "KPCA denoising and the pre-image problem revisited," *Dig. Signal Process.*, vol. 18, no. 4, pp. 568–580, Jul. 2008.

[6] J. T.-Y. Kwok and I. W.-H. Tsang, "The pre-image problem in kernel methods," *IEEE Trans. Neural Netw.*, vol. 15, no. 6, pp. 1517–1525, Nov. 2004.

[7] G. H. Bakir, J. Weston, and B. Schölkopf, "Learning to find pre-images," in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. K. Saul, and B. Schölkopf, Eds. Cambridge, MA: MIT Press, 2003, pp. 449–456.

[8] W.-S. Zheng, J. Lai, and P. C. Yuen, "Penalized preimage learning in kernel principal component analysis," *IEEE Trans. Neural Netw.*, vol. 21, no. 4, pp. 551–570, Apr. 2010.

[9] M. H. Nguyen and F. De la Torre, "Robust kernel principal component analysis," in *Proc. 22nd Annu. Conf. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2009, pp. 1185–1192.

[10] I. T. Jolliffe, *Principal Component Analysis*. New York: Springer-Verlag, 2002.

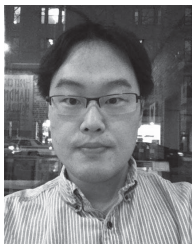
[11] R. A. Johnson and D. W. Wichern, *Applied Multivariate Statistical Analysis*. Englewood Cliffs, NJ: Prentice Hall, 2007.

[12] A. J. Izenman, *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*. New York: Springer-Verlag, 2008.

[13] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press, 2001.

[14] P. Y. Simard, Y. A. LeCun, J. S. Denker, and B. Victorri, "Transformation invariance in pattern recognition tangent distance and tangent propagation," in *Neural Networks: Tricks of the Trade*, G. B. Orr and K.-R. Muller, Eds. New York: Springer-Verlag, 1998.

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



Joon-Ku Im received the B.S. degree in mathematics from Seoul National University, Seoul, South Korea, the M.S. degree in mathematics and the Master of Statistics degree from Ohio State University, Columbus. He is currently pursuing the Ph.D. degree in industrial engineering and management sciences with Northwestern University, Evanston, IL.

His current research interests include statistical modeling and data mining for discovering knowledge in large datasets with emphasis on manufacturing variation reduction for processes with advanced measurement technologies, healthcare applications, and financial risk modeling.

Daniel W. Apley received the B.S., M.S., and Ph.D. degrees in mechanical engineering and the M.S. degree in electrical engineering from the University of Michigan, Ann Arbor.

He is an Associate Professor of industrial engineering and management sciences with Northwestern University, Evanston, IL. His research has been supported by numerous industries and government agencies. His current research interests include the interface of engineering modeling, statistical analysis, and data mining, with particular emphasis on manufacturing variation reduction applications.

Dr. Apley received the National Science Foundation CAREER Award in 2001, the IIE Transactions Best Paper Award in 2003, and the Wilcoxon Prize for Best Practical Application Paper appearing in Technometrics in 2008. He currently serves as an Editor-in-Chief for the *Journal of Quality Technology* and has served as the Chair of the Quality, Statistics, and Reliability Section of INFORMS, the Director of the Manufacturing and Design Engineering Program at Northwestern, and an Associate Editor for *Technometrics*.

George C. Runger received the Degree in industrial engineering and statistics and the Ph.D. degree.

He was a Senior Engineer at IBM, Murray Hill, NJ. He is a Professor with the School of Computing, Informatics and Decision Systems Engineering, Arizona State University, Tempe. His current research interests include real-time monitoring and other data-analysis methods with a focus on large, complex data sets with applications in areas such as manufacturing and health care.