Penalized Preimage Learning in Kernel Principal Component Analysis

Wei-Shi Zheng, *Member, IEEE*, JianHuang Lai, and Pong C. Yuen, *Member, IEEE*

***Abstract*—Finding the preimage of a feature vector in kernel principal component analysis (KPCA) is of crucial importance when KPCA is applied in some applications such as image preprocessing. Since the exact preimage of a feature vector in the kernel feature space, normally, does not exist in the input data space, an approximate preimage is learned and encouraging results have been reported in the last few years. However, it is still difficult to find a "good" estimation of preimage. As estimation of preimage in kernel methods is ill-posed, how to guide the preimage learning for a better estimation is important and still an open problem. To address this problem, a penalized strategy is developed in this paper, where some penalization terms are used to guide the preimage learning process. To develop an efficient penalized technique, we first propose a two-step general framework, in which a preimage is directly modeled by weighted combination of the observed samples and the weights are learned by some optimization function subject to certain constraints. Compared to existing techniques, this would also give advantages in directly turning preimage learning into the optimization of the combination weights. Under this framework, a penalized methodology is developed by integrating two types of penalizations. First, to ensure learning a well-defined preimage, of which each entry is not out of data range, convexity constraint is imposed for learning the combination weights. More insight effects of the convexity constraint are also explored. Second, a penalized function is integrated as part of the optimization function to guide the preimage learning process. Particularly, the weakly supervised penalty is proposed, discussed, and extensively evaluated along with Laplacian penalty and ridge penalty. It could be further interpreted that the learned preimage can preserve some kind of pointwise conditional mutual information. Finally, KPCA with preimage learning is applied on face image data sets in the aspects of facial expression normalization, face image denoising, recovery of missing parts from occlusion, and illumination normalization. Experimental results show that the proposed preimage learning algorithm obtains lower mean square error (MSE) and better visual quality of reconstructed images.**

***Index Terms*—Kernel, kernel principal component analysis (KPCA), locality preservation, penalty function, preimage problem.**

Manuscript received December 29, 2007; revised June 27, 2009 and October 21, 2009; accepted December 09, 2009. Date of publication February 08, 2010; date of current version April 02, 2010. This work was supported by the National Science Foundation of China (NSFC) under Grants 60675016 and 60633030, the 973 Program 2006CB303104, the National Science Foundation (NSF) Guangdong under Grant U0835005, and the Hong Kong Earmarked Grant HKBU2113/06E.

W.-S. Zheng was with the School of Mathematics and Computational Science, Sun Yat-sen University, Guangzhou 510275, China. He is now with the Department of Computer Science, Queen Mary University of London, London E1 4NS, U.K. (e-mail: wszheng@ieee.org).

J. Lai is with the School of Information Science and Technology, Sun Yat-sen University, Guangzhou 510275, China (e-mail: stsljh@mail.sysu.edu.cn).

P. C. Yuen is with the Department of Computer Science, Hong Kong Baptist University, Hong Kong (e-mail: pcyuen@comp.hkbu.edu.hk).

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/TNN.2009.2039647

I. Introduction

PRINCIPAL COMPONENT ANALYSIS (PCA) [1] is a well-known technique and has been widely applied in unsupervised learning and dimension reduction. However, real-world data are always generated by a nonlinear system, and this nonlinearity hidden in data is difficult to be explored by PCA, which is designed for data of linear source. Kernel principal component analysis (KPCA) [2] is therefore developed as a tractable technique for such nonlinear modeling.

KPCA is developed based on the theory of reproducing kernel Hilbert space (RKHS) [3], [4]. Suppose $\mathbf{X}(\subset \Re^n)$ is the input data space and $\mathbf{H}_\kappa$ is an RKHS associated with a kernel function $\kappa(\mathbf{x},\mathbf{y}) =< \varphi(\mathbf{x}), \varphi(\mathbf{y}) >$, where $\mathbf{x},\mathbf{y} \in \mathbf{X}$ and $\varphi(\cdot)$ is an implicit mapping induced by κ such that $\varphi(\mathbf{x}) : \mathbf{X} \rightarrow \mathbf{H}_\kappa$. With the kernel trick [3], [4], only the kernel function κ rather than $\varphi(\cdot)$ needs to be explicitly defined.

KPCA has recently been used for nonlinear data preprocessing. Encouraging results have been seen in image denoising [5]–[10], image compression [7], image super-resolution [11], etc. PCA approximates data in a principal component subspace that preserves the greatest variations of data, so that noises or other nonmain variations of data are expected to be removed in that subspace. KPCA essentially has almost the same philosophy but differs in that it performs PCA in the kernel feature space $\mathbf{H}_\kappa$, so nonlinear processing can be performed on input data.

In order to realize the nonlinear preprocessing by KPCA, the preimage learning in KPCA is an important step. In this paper, we call this whole process as "KPCA + preimage learning." To specify the preimage problem, we demonstrate "KPCA + preimage learning" for data preprocessing in Fig. 1. Any input pattern $\mathbf{x}(\in \mathbf{X})$ is first mapped to $\varphi(\mathbf{x})$ in the feature space $\mathbf{H}_\kappa$. Then, $\varphi(\mathbf{x})$ is projected onto the kernel principal component subspace in $\mathbf{H}_\kappa$ and the projection is denoted by $P_\kappa\varphi(\mathbf{x})$. Finally, a preimage $\hat{\mathbf{x}}$ is found in the input data space $\mathbf{X}$ such that

$$\varphi(\hat{\mathbf{x}}) = P_\kappa\varphi(\mathbf{x}). \tag{1}$$

Then, $\hat{\mathbf{x}}$ is the output of this nonlinear process.

In this process, the central problem is the preimage learning at the final step. It is difficult (always impossible) to find an exact preimage $\hat{\mathbf{x}}$ entirely satisfying $\varphi(\hat{\mathbf{x}}) = P_\kappa\varphi(\mathbf{x})$. Note that when using popular kernels such as radial basis function (RBF) and polynomial kernels, the feature space $\mathbf{H}_\kappa$ may always hold much higher or infinite dimensionality while the input data space $\mathbf{X}$ holds finite dimensionality [12]. In this sense, all points in the input data space $\mathbf{X}$ may be mapped onto the manifold $\varphi(\mathbf{X})$, which may be a nonlinear hyperplane in $\mathbf{H}_\kappa$ as shown in Fig. 1. Therefore, if $P_\kappa\varphi(\mathbf{x}) \notin \varphi(\mathbf{X})$, then one

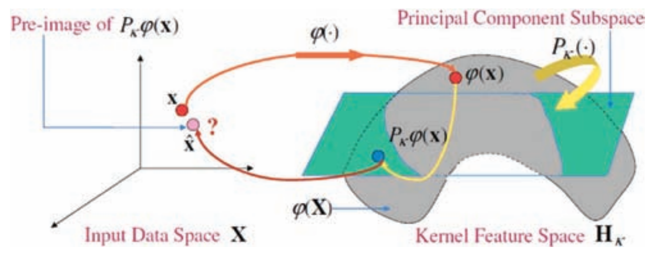

Pre-image of $P_\kappa\varphi(\mathbf{x})$
$\varphi(\cdot)$
Principal Component Subspace
$P_\kappa(\cdot)$
$\varphi(\mathbf{x})$
$\mathbf{x}$
?
$\hat{\mathbf{x}}$
$P_\kappa\varphi(\mathbf{x})$
$\varphi(\mathbf{X})$
Input Data Space $\mathbf{X}$
Kernel Feature Space $\mathbf{H}_\kappa$

Fig. 1. Illustration of the process "KPCA+ preimage learning." Step 1: input sample $\mathbf{x}$ is mapped into a kernel feature space $\mathbf{H}_\kappa$ by mapping φ and $\varphi(\mathbf{x})$ is the corresponding feature point. Step 2: $\varphi(\mathbf{x})$ is projected onto the kernel principal component subspace by transform P_κ for some preprocessing and the projection is $P_\kappa\varphi(\mathbf{x})$. Step 3: find the preimage $\hat{\mathbf{x}}$ of the feature vector $P_\kappa\varphi(\mathbf{x})$.

cannot find $\hat{\mathbf{x}}$ satisfying (1), and therefore, an exact preimage $\hat{\mathbf{x}}$ does not exist.

To address the preimage learning problem, some algorithms have been developed. Mika *et al.* [5] first reported the concept of preimage and proposed an iterative method to determine the preimage by minimizing least square distance error. This work gave a foundation for preimage learning. Later, Kwok and Tsang proposed to find the preimage under distance constraints [6] using a similar technique used in multidimensional scaling (MDS) [13]. It is good that some knowledge in manifold learning [14]–[16] is used for preimage learning [6]. Bakır *et al.* [7] proposed to learn preimage by learning a preimage mapping using the idea of kernel regression. Some other recent works have been reported by Dambreville *et al.* [17] and Arias *et al.* [10]. In [17], Mika's algorithm was modified and an approximate solution was given in a noniterative manner; in [10], Nystrom extension was introduced, but methods developed in [5] and [6] were still required to find the preimage finally.

Existing methods always perform based on some assumptions or models. However, it is still difficult to find a "good" preimage, as the estimation is ill-posed. The distance constraint based method proposed by Kwok and Tsang is based on the assumption that the exact preimage exists, and an approximate solution could be found under distance constraints [6]. In preimage map, some intuitive label information [7] is explicitly used in a kernel regression manner, but as analyzed later, negative information is not used. Also, how to choose the proper kernel and parameters for kernel regression is still a problem. Therefore, guiding the preimage learning process for a better estimation is still an open problem.

To address this ill-posed estimation problem, we propose a penalization model called *penalized preimage learning* ($\mathrm{P^2L}$). To develop an efficient penalized methodology, we first formulate a two-step general framework, in which the preimage is directly learned by weighted combination of the observed samples subject to some constraints. Compared to existing methods, it is advantageous to directly turn preimage learning into the optimization of the combination weights. Under this framework, a penalized strategy is developed to alleviate the ill-posed estimation problem. Two types of penalizations are considered and proposed. First, the combination weights are learned subject to convexity constraint. This ensures learning a well-defined preimage whose entries are not out of range. More insight effects of the convexity constraint will be explored. Second, a penalty function is integrated to guide the preimage learning process for a better estimation. Particularly, we propose the weakly supervised penalty and extensively evaluate it along with Laplacian penalty and ridge penalty. We further interpret that some kind of pointwise conditional mutual information is preserved by the learned preimage. In experiments, we apply the proposed preimage learning to human face applications. Some ideas of this work were reported in two preliminary works [8], [9], but the presented methodology in this paper is significantly different.

The rest of this paper is organized as follows. Section II will give review and analysis of existing methods. A general framework of learning preimage in KPCA is formulated in Section III and the penalized preimage learning model is proposed in Section IV. The experimental results are reported in Section V. Finally, conclusions and discussions are given in Section VI.

II. Review of Existing Methods

Suppose $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbf{X}(\subset \Re^n)$ are N training samples. Let $\mathbf{\Psi} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$, $\mathbf{\Psi}_\varphi = [\varphi(\mathbf{x}_1), \ldots, \varphi(\mathbf{x}_N)]$, and $\boldsymbol{\mu}^\varphi = (1/N)\sum_{i=1}^N \varphi(\mathbf{x}_i) = (1/N)\mathbf{\Psi}_\varphi \mathbf{e}_N$, where $\mathbf{e}_N = (1, \ldots, 1)^T \in \Re^N$. For KPCA, its eigenvectors can be learned by computing the eigenvectors of $N^{-1}\mathbf{\Psi}_\varphi(\mathbf{I} - N^{-1}\mathbf{e}_N\mathbf{e}_N^T)\mathbf{\Psi}_\varphi^T$. The projection of $\varphi(\mathbf{x})$ onto the kernel principal subspace $P_\kappa\varphi(\mathbf{x})$ can be written as $P_\kappa\varphi(\mathbf{x}) = \mathbf{\Psi}_\varphi\boldsymbol{\gamma}^{\mathbf{x}}$ for some $\boldsymbol{\gamma}^{\mathbf{x}} = (\gamma_1^{\mathbf{x}}, \ldots, \gamma_N^{\mathbf{x}})^T \in \Re^N$. The details are given in Appendix I. For derivation of a general framework, we review and analyze three major approaches in preimage learning.

A. Least Square Distance Minimization

Mika *et al.* proposed to find the preimage by minimizing the following least square distance [5]:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}'} G(\mathbf{x}'), \qquad G(\mathbf{x}') = \|\varphi(\mathbf{x}') - P_\kappa\varphi(\mathbf{x})\|^2. \quad (2)$$

Particularly, for RBF kernel, $\kappa(\mathbf{x}, \mathbf{y}) = \exp -\|\mathbf{x} - \mathbf{y}\|^2/c), \mathbf{x}, \mathbf{y} \in \Re^n$ and $c > 0, G(\mathbf{x}')$ becomes

$$G(\mathbf{x}') = 1 - 2\sum_{i=1}^N \gamma_i^{\mathbf{x}} \exp(-c^{-1}\|\mathbf{x}' - \mathbf{x}_i\|^2) + < P_\kappa\varphi(\mathbf{x}), P_\kappa\varphi(\mathbf{x}) > . \quad (3)$$

Taking its derivative with respect to $\mathbf{x}'$ and setting it to zero yield a (local) minimum as follows:

$$\hat{\mathbf{x}} = \frac{\sum_{i=1}^N \gamma_i^{\mathbf{x}} \exp(-c^{-1}\|\hat{\mathbf{x}} - \mathbf{x}_i\|^2)\mathbf{x}_i}{\sum_{j=1}^N \gamma_j^{\mathbf{x}} \exp(-c^{-1}\|\hat{\mathbf{x}} - \mathbf{x}_j\|^2)}. \quad (4)$$

For implementation, iterative method was used to find a (local) minimum as follows:

$$\hat{\mathbf{x}}_{t+1} = \frac{\sum_{i=1}^N \gamma_i^{\mathbf{x}} \exp(-c^{-1}\|\hat{\mathbf{x}}_t - \mathbf{x}_i\|^2)\mathbf{x}_i}{\sum_{j=1}^N \gamma_j^{\mathbf{x}} \exp(-c^{-1}\|\hat{\mathbf{x}}_t - \mathbf{x}_j\|^2)}. \quad (5)$$

Such kind of update is a particular form of gradient descent [9]

$$\hat{\mathbf{x}}_{t+1} = \hat{\mathbf{x}}_t - \rho \cdot \left.\frac{\partial G}{\partial \mathbf{x}'}\right|_{\mathbf{x}'=\hat{\mathbf{x}}_t}, \qquad \rho \geq 0. \quad (6)$$

For (5), $\rho = \rho_t = 0.25 \cdot c \cdot (\sum_{i=1}^N \gamma_i^{\mathbf{x}}\kappa(\mathbf{x}_i, \hat{\mathbf{x}}_t))^{-1}$. When the fixed-point update scheme in (5) may not be effective for other kernel functions (e.g., polynomial kernel), the gradient descent update can be used. Note that (5) is iterative, so the initial position needs to be determined in advance. To avoid iterative computation, Dambreville *et al.* did some modification on the right-hand side of (4) based on some assumption [17].

B. Distance Constraint

Assume the exact preimage $\hat{\mathbf{x}}$ of $P_\kappa\varphi(\mathbf{x})$ exists and suppose the distances between $P_\kappa\varphi(\mathbf{x})$ and its s neighbors $\varphi(\tilde{\mathbf{x}}_1),\ldots,\varphi(\tilde{\mathbf{x}}_s)$ are $d_1^\varphi,\ldots,d_s^\varphi$ in the feature space, respectively, where $\tilde{\mathbf{x}}_1,\ldots,\tilde{\mathbf{x}}_s \in \{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$. By inferring the corresponding distance d_i between $\hat{\mathbf{x}}$ and $\tilde{\mathbf{x}}_i$ [6], Kwok and Tsang suggested an MDS-based technique [13] to learn an approximate preimage $\hat{\mathbf{x}}$ subject to these distance constraints [6]. Let $\mathbf{\Psi}_{s,\mathrm{x}} = [\tilde{\mathbf{x}}_1,\ldots,\tilde{\mathbf{x}}_s]$ and $\mathbf{H}_s = \mathbf{I}_s - s^{-1}\mathbf{e}_s\mathbf{e}_s^T$, where $\mathbf{I}_s$ is an $s\times s$ identity matrix and $\mathbf{e}_s = (1,\ldots,1)^T \in \Re^s$. SVD decomposition yields $\mathbf{\Psi}_{s,\mathrm{x}}\mathbf{H}_s = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$. Define $\mathbf{Z}_s = [\mathbf{z}_1,\ldots,\mathbf{z}_s] = \mathbf{\Lambda}\mathbf{V}^T$. Then, the approximated preimage $\hat{\mathbf{x}}$ is learned [6] by

$$\hat{\mathbf{x}} = \mathbf{U}\hat{\mathbf{z}} + \bar{\mathbf{x}} \tag{7}$$

where $\hat{\mathbf{z}} = -0.5\cdot(\mathbf{Z}_s\mathbf{Z}_s^T)^{-1}\mathbf{Z}_s(\mathbf{d}_s^2 - \mathbf{d}_{s,0}^2)$, $\mathbf{d}_s^2 = (d_1^2,\ \ldots,d_s^2)^T, \mathbf{d}_{s,0}^2 = (\|\mathbf{z}_1\|^2,\ldots,\|\mathbf{z}_s\|^2)^T$, and $\bar{\mathbf{x}} = s^{-1}\sum_{j=1}^s \tilde{\mathbf{x}}_j$.

This method avoids iterative computation but the optimal solution may not be unique as analyzed in [6] and [8], because the number of variables (the data dimensionality) always seriously exceeds the number of constraints. Thus, only approximate solution can be currently obtained in [6]. This partially explains why the algorithm may sometimes be unsatisfactory in dealing with high-dimensional data [8].

C. Preimage Map

Bakır *et al.* [7] proposed to learn a preimage map

$$\mathbf{\Gamma}(P_\kappa\varphi(\mathbf{x})) = [\mathbf{\Gamma}_1(P_\kappa\varphi(\mathbf{x})),\ldots,\mathbf{\Gamma}_n(P_\kappa\varphi(\mathbf{x}))]^T$$

where $\mathbf{\Gamma}_j(P_\kappa\varphi(\mathbf{x})) = \sum_{i'=1}^N \beta_{i'}^j\kappa'(P_\kappa\varphi(\mathbf{x}), P_\kappa\varphi(\mathbf{x}_{i'}))$ and κ' is a new kernel function different from κ. The following kernel-regression-based criterion [7] is then suggested:

$$\mathbf{\Gamma} = \arg\min_{\mathbf{\Gamma}} \sum_{i=1}^N l(\mathbf{x}_i - \mathbf{\Gamma}(P_\kappa\varphi(\mathbf{x}_i))) + \upsilon\cdot\Phi(\mathbf{\Gamma}), \qquad \upsilon \ge 0 \tag{8}$$

where $\Phi(\mathbf{\Gamma})$ is a regularization term and l is a loss function.

Define $\mathbf{B} = [\boldsymbol{\beta}^1,\ldots,\boldsymbol{\beta}^n]$ where $\boldsymbol{\beta}^j = (\beta_1^j,\ldots,\beta_N^j)^T$. When $l(\mathbf{x}_i - \mathbf{\Gamma}(P_\kappa\varphi(\mathbf{x}_i))) = \|\mathbf{x}_i - \mathbf{\Gamma}(P_\kappa\varphi(\mathbf{x}_i))\|^2$ and $\Phi(\mathbf{\Gamma}) = \Phi(\{\boldsymbol{\beta}^j\}) = \sum_{j=1}^n \|\boldsymbol{\beta}^j\|^2$, the criterion can be reformulated and the solution is as follows [7]:

$$\begin{aligned}\mathbf{B} &= \arg\min_{\mathbf{B}} \mathrm{tr}([\mathbf{\Psi}^T - \mathbf{K}'\mathbf{B}][\mathbf{\Psi}^T - \mathbf{K}'\mathbf{B}]^T) + \upsilon\cdot\mathrm{tr}(\mathbf{B}\mathbf{B}^T)\\ &= (\mathbf{K}'^T\mathbf{K}' + \upsilon\cdot\mathbf{I}_N)^{-1}\mathbf{K}'^T\mathbf{\Psi}^T\end{aligned} \tag{9}$$

where $\mathbf{K}' = (\kappa'(P_\kappa\varphi(\mathbf{x}_i), P_\kappa\varphi(\mathbf{x}_j)))$ is a new kernel matrix. Define $\boldsymbol{\kappa}'_{\mathbf{x}} = (\kappa'(P_\kappa\varphi(\mathbf{x}), P_\kappa\varphi(\mathbf{x}_1)),\ldots,\kappa'(P_\kappa\varphi(\mathbf{x}), P_\kappa\varphi(\mathbf{x}_N)))$ and preimage map learns the preimage $\hat{\mathbf{x}}$ of $P_\kappa\varphi(\mathbf{x})$ by

$$\hat{\mathbf{x}} = \mathbf{\Gamma}(P_\kappa\varphi(\mathbf{x})) = (\boldsymbol{\kappa}'_{\mathbf{x}}\mathbf{B})^T. \tag{10}$$

Preimage map exploits some intuitive label information of data for kernel dependency estimation-based regression [7]. However, currently only good samples could be used for training. For example, if $\mathbf{x}_i$ is a noisy image, it is not suitable to anticipate that the preimage $\mathbf{\Gamma}(P_\kappa\varphi(\mathbf{x}_i))$ would be close to its original noisy image $\mathbf{x}_i$. In addition, how to select a proper kernel function κ' for kernel dependency estimation is still a problem.

III. A Two-Step General Framework

In this section, we formulate a two-step general framework for preimage learning. It would facilitate the development of penalization methodology presented later. By studying existing preimage learning algorithms, there is a common point on their final solutions. Though all these algorithms are formulated differently, the estimated preimages are the linear combination of training samples with different weights.[1] That is, given a training sample set $\{\mathbf{x}_i, i = 1,\ldots,N\}$, the preimage of a feature vector $P_\kappa\varphi(\mathbf{x})$ is actually determined by $\hat{\mathbf{x}} = \sum_{i=1}^N w_i\mathbf{x}_i$, with coefficients $w_i, i = 1,\ldots,N$. Different methods have different estimates of w_i. The following outlines these scenarios.

In Mika *et al.*'s method [5], when using RBF kernel, the preimage of $P_\kappa\varphi(\mathbf{x})$ is determined by

$$\hat{\mathbf{x}} = \frac{\sum_{i=1}^N \gamma_i^{\mathbf{x}}\kappa(\mathbf{x}_i,\hat{\mathbf{x}})\mathbf{x}_i}{\sum_{j=1}^N \gamma_j^{\mathbf{x}}\kappa(\mathbf{x}_j,\hat{\mathbf{x}})} = \sum_{i=1}^N \frac{\gamma_i^{\mathbf{x}}\kappa(\mathbf{x}_i,\hat{\mathbf{x}})}{\sum_{j=1}^N \gamma_j^{\mathbf{x}}\kappa(\mathbf{x}_j,\hat{\mathbf{x}})}\mathbf{x}_i. \tag{11}$$

So we have $w_i = [\sum_{j=1}^N \gamma_j^{\mathbf{x}}\kappa(\mathbf{x}_j,\hat{\mathbf{x}})]^{-1}\gamma_i^{\mathbf{x}}\kappa(\mathbf{x}_i,\hat{\mathbf{x}})$. For polynomial kernel, if the initialization of $\mathbf{x}'$ in (6) resides in the range space of training samples, it is still true that $\hat{\mathbf{x}} = \sum_{i=1}^N w_i\mathbf{x}_i$ for some w_i.

In Kwok and Tsang's method [6], define $\mathbf{B}_{s,\mathbf{x}} = \mathbf{H}_s\mathbf{V}\mathbf{\Lambda}^{-1}\hat{\mathbf{z}}$, and then (7) can be reexpressed as

$$\hat{\mathbf{x}} = \mathbf{\Psi}_{s,\mathbf{x}}\mathbf{B}_{s,\mathbf{x}} + s^{-1}\mathbf{\Psi}_{s,\mathbf{x}}\mathbf{e}_s = \mathbf{\Psi}_{s,\mathbf{x}}(\mathbf{B}_{s,\mathbf{x}} + s^{-1}\mathbf{e}_s). \tag{12}$$

Note that matrix $\mathbf{\Psi}_{s,\mathbf{x}}$ is part of $\mathbf{\Psi}$. Denote $\mathbf{w} = (w_1,\ldots,w_N)^T$. Let $[\mathbf{x}_{i_1},\ldots,\mathbf{x}_{i_s}] = [\tilde{\mathbf{x}}_1,\ldots,\tilde{\mathbf{x}}_s], (w_{i_1},\ldots,w_{i_s})^T = \mathbf{B}_{s,\mathbf{x}} + s^{-1}\mathbf{e}_s$ and $w_j = 0$ for $j \notin \{i_1,\ldots,i_s\}$. Thereby, we have $\hat{\mathbf{x}} = \sum_{i=1}^N w_i\mathbf{x}_i$.

In Bakır *et al.*'s algorithm [7], $\hat{\mathbf{x}} = \mathbf{\Gamma}\ (P_\kappa\varphi(\mathbf{x})) = (\mathbf{k}'_{\mathbf{x}}\mathbf{B})^T = \mathbf{\Psi}\mathbf{K}'(\mathbf{K}'^T\mathbf{K}' + \upsilon\mathbf{I}_N)^{-1}\boldsymbol{\kappa}_{\mathbf{x}}'^T$. Define $\mathbf{w} = (w_1,\ldots,w_N)^T = \mathbf{K}'(\mathbf{K}'^T\mathbf{K}' + \upsilon\mathbf{I}_N)^{-1}\boldsymbol{\kappa}_{\mathbf{x}}'^T$, and then, we have $\hat{\mathbf{x}} = \sum_{i=1}^N w_i\mathbf{x}_i$.

The above analysis justifies that the preimage values determined by existing algorithms all lie in the space spanned by training samples. When the sample size is large enough, the range space of training samples will cover the input data space so that the preimage can be completely represented by them. When the sample size is not enough, the rationale of this kind of weighted representation is that the preimage is found in

[1]Kwok and Tsang were motivated before to use the (nearest) training samples to estimate the preimage in [6].

the entire principal component space [3] if data are centered.[2] Therefore, by devising the optimization function G to learn the weights $\{w_i^{\mathbf{x}}\}$ for a given training set $\{\mathbf{x}_i\}$ and feature vector $P_\kappa\varphi(\mathbf{x})$, we formulate a general framework for pursuing the preimage $\hat{\mathbf{x}}$ of feature vector $P_\kappa\varphi(\mathbf{x})$ in KPCA by solving the following problem:

$$\hat{\mathbf{x}} = \mathbf{\Psi}\mathbf{w}_{\mathbf{x}} = \sum_{i=1}^{N} w_i^{\mathbf{x}}\mathbf{x}_i$$

where

$$\begin{cases} \mathbf{w}_{\mathbf{x}} = \underset{\mathbf{w}=(w_1,\dots,w_N)^T\in\Re^N}{\arg\min} G(\mathbf{w}|P_\kappa\varphi(\mathbf{x}), \{\mathbf{x}_i\}) \\ \text{s.t. } \mathbf{w}_{\mathbf{x}} \text{ satisfies some constraints.} \end{cases} \tag{13}$$

In this two-step framework, we directly formulate the preimage solution by linear combination of training samples. This would directly turn preimage learning into the optimization of the combination weights. Note that $\varphi(\mathbf{x}_1),\dots,\varphi(\mathbf{x}_N)$ have their exact preimages $\mathbf{x}_1,\dots,\mathbf{x}_N$, respectively, so some local structures associated to the feature vector $P_\kappa\varphi(\mathbf{x})$ explored in the kernel feature space can therefore be preserved in the input data space by the weights.

It is important to point out that this framework is inspired by existing work, but it differs in that the combination weights of training samples are directly modeled and the learning problem can be directly turned into the optimization of the combination weights, while indirect modeling is done by existing methods. Another benefit of this framework is that the function G can be constructed more easily by incorporating prior information and this would facilitate the development of more efficient and effective preimage algorithms. As preimage learning is always ill-posed, prior knowledge can be useful to alleviate the problem.

IV. Penalized Preimage Learning

Based on the introduced framework in Section III, we design and develop a new method called penalized preimage learning (P^{2}L) model for preimage estimation in order to alleviate the ill-posed estimation. The penalization will be twofold. First, convexity constraint is imposed for learning weights $\{w_i\}$ to penalize the range of preimage such that it is well defined and also to help explore some pointwise conditional mutual information as interpreted later; second, a penalized term is further formulated in the optimization function G. So, prior knowledge could be integrated to guide preimage learning to alleviate the ill-posed estimation problem.

Moreover, in this section, to realize the proposed framework, we design the function G by partially absorbing some knowledge in manifold learning. We absorb the idea of locally linear embedding (LLE) [14], which preserves local information through the local least square reconstruction weights. This is motivated by the perspective that the dimension of feature space $\mathbf{H}_\kappa$ is always much higher than that of input data space $\mathbf{X}$ when using popular kernels such as RBF [12]. Thus, finding the preimage $\hat{\mathbf{x}}$ of $P_\kappa\varphi(\mathbf{x})$ could be viewed as finding an embedding point for $P_\kappa\varphi(\mathbf{x})$ in $\mathbf{X}$. However, preimage learning could be different from manifold learning. We aim at finding the well-defined real-world data of a feature vector value rather than estimating its intrinsic impact parameters. With penalization on the combination weights in the proposed model, some penalized local nonnegative reconstruction information is explored and the insight of this kind of preservation will be given in our study. While Kwok and Tsang suggested using some local distance information [6] for preimage learning, the proposed penalized framework explores different local information which will be discussed in latter sections.

A. Proposed Model

Given the projection of the feature vector $\varphi(\mathbf{x})$ onto the kernel principal component subspace, $P_\kappa\varphi(\mathbf{x})$, we define

$$U(\mathbf{x},\delta) = \{\mathbf{x}_i | \|\varphi(\mathbf{x}_i) - P_\kappa\varphi(\mathbf{x})\|^2 \le \delta, i = 1,\dots,N\} \tag{14}$$

where $\delta > 0$. Let $s = |U(\mathbf{x},\delta)|$ be the cardinality of $U(\mathbf{x},\delta)$ where $s \le N$, and let $\mathbf{\Psi}_{s,\varphi(\mathbf{x})} = [\varphi(\tilde{\mathbf{x}}_1),\dots,\varphi(\tilde{\mathbf{x}}_s)]$, where $\tilde{\mathbf{x}}_i \in U(\mathbf{x},\delta), i = 1,\dots,s$, and $\varphi(\tilde{\mathbf{x}}_j) \neq \varphi(\tilde{\mathbf{x}}_{j'})$ if $j \neq j'$. In practice, it may be difficult to determine the value of δ and an alternative way is to determine the parameter s, the threshold number of neighbors of a feature vector.

Redefine $\mathbf{\Psi}_{s,\mathbf{x}} = [\tilde{\mathbf{x}}_1,\dots,\tilde{\mathbf{x}}_s]$. Under the general framework, we design the new criterion (15), shown at the bottom of the page, for preimage learning with convexity constraint on weights $w_i^{\mathbf{x}}$.

Regardless of the convexity constraint "$\sum_{i=1}^s w_i^{\mathbf{x}} = 1$ and $w_i^{\mathbf{x}} \ge 0$," the optimization function $\|\mathbf{\Psi}_{s,\varphi(\mathbf{x})}\mathbf{w} - P_\kappa\varphi(\mathbf{x})\|^2$ looks similar to (2) proposed by Mika *et al.* However, they are notably different. First, criterion (15) directly models the preimage as the combination of training samples, while it is not modeled directly[3] in Mika's method (2); the proposed modeling gives the advantage in avoiding iterative manner for optimization and also brings efficiency to incorporation of prior knowledge. Second, the proposed algorithm is localized; that is the preimage is determined by the assumed nearby neighbors $\tilde{\mathbf{x}}_1,\dots,\tilde{\mathbf{x}}_s$ rather than all training samples, but this cannot be realized in (2). Moreover, with the penalty on the range of the weights, we can further immediately get the following proposition.

Proposition: The preimage learned by criterion (15) is well defined.

This is because convexity constraint would ensure the learned preimage $\hat{\mathbf{x}}$ to be well defined, i.e., the entries of $\hat{\mathbf{x}}$ are not out of range. For example, the range of entry is always set to be 0–1 or

[2]If data are not centered, the estimated preimage is the combination of data mean and principal components of training data.

[3]Though the solution of Mika's algorithm can be represented by weighted combination of training samples, exactly speaking, this conclusion can only be obtained from its iterative form in (4) rather than by a direct modeling.

$$\hat{\mathbf{x}} = \mathbf{\Psi}_{s,\mathbf{x}}\mathbf{w}_{\mathbf{x}} = \sum_{i=1}^{s} w_i^{\mathbf{x}}\tilde{\mathbf{x}}_i, \qquad \text{where} \quad \begin{cases} \mathbf{w}_{\mathbf{x}} = (w_1^{\mathbf{x}},\dots,w_s^{\mathbf{x}})^T = \underset{\mathbf{w}=(w_1,\dots,w_s)^T\in\Re^s}{\arg\min} \left\|\mathbf{\Psi}_{s,\varphi(\mathbf{x})}\mathbf{w} - P_\kappa\varphi(\mathbf{x})\right\|^2 \\ \text{s.t. } \sum_{i=1}^s w_i^{\mathbf{x}} = 1 \quad \text{and} \quad w_i^{\mathbf{x}} \ge 0, \qquad i = 1,\dots,s. \end{cases} \tag{15}$$

0–255 for 8-bit image. More specifically, if the jth entry $\tilde{\mathbf{x}}_i(j)$ of data vector $\tilde{\mathbf{x}}_i$ is within the domain $[a_j, b_j]$, then $\hat{\mathbf{x}}(j)$ will also be in this domain, because

$$a_j = a_j \sum_{i=1}^{s} w_i^{\mathbf{x}} \leq \sum_{i=1}^{s} w_i^{\mathbf{x}} \tilde{\mathbf{x}}_i(j) \leq b_j \sum_{i=1}^{s} w_i^{\mathbf{x}} = b_j.$$

Besides the well-defined property, as shown in the experiment, the convexity constraint would also regularize the effect of weakly supervised penalty proposed in model (16). Moreover, with the convexity constraint, model (16) can still perform well, even though some popular normalization techniques are not used finally to refine the range of the preimage values. In Appendix II, we show some example of this scenario.

In addition, the convexity constraint also benefits for interpretation of the penalized model, which will be given later. Specially, if $s = N$ in (15), global information would be learned; if $s \ll N$, we in fact learn an appropriate preimage of $P_\kappa \varphi(\mathbf{x})$ by preserving some local nonnegative linear reconstruction information between $P_\kappa \varphi(\mathbf{x})$ and its neighbors. The latter case is preferred in this paper. It will be further interpreted later and partially justified in the experiment.

As the exact preimage does not typically exist, penalization would therefore be useful for guiding the learning process to find a better preimage value, such as incorporating some prior knowledge. For this reason, the proposed criterion of learning $\mathbf{w}_{\mathbf{x}}$ in (15) can be further penalized by integrating a penalty function $F(\mathbf{w})$ into the optimization function. Note that

$$\begin{aligned}\left\|\Psi_{s,\varphi(\mathrm{x})}\mathbf{w} - P_\kappa \varphi(\mathbf{x})\right\|^2 &= \mathbf{w}^T \mathbf{\Psi}_{s,\varphi(\mathbf{x})}^T \mathbf{\Psi}_{s,\varphi(\mathbf{x})} \mathbf{w} \\ &\quad -2(\mathbf{\Psi}_\varphi \boldsymbol{\gamma}^{\mathrm{x}})^T \mathbf{\Psi}_{s,\varphi(\mathrm{x})}\mathbf{w} + (\mathbf{\Psi}_\varphi \boldsymbol{\gamma}^{\mathrm{x}})^T \mathbf{\Psi}_\varphi \boldsymbol{\gamma}^{\mathrm{x}}.\end{aligned}$$

Therefore, the *penalized preimage learning* ($\mathrm{P}^2\mathrm{L}$) model can be devised as shown in (16), at the bottom of the page, by excluding the term $(\mathbf{\Psi}_\varphi \boldsymbol{\gamma}^{\mathbf{x}})^T \mathbf{\Psi}_\varphi \boldsymbol{\gamma}^{\mathbf{x}}$ which is independent of $\mathbf{w}$, where the parameter $\lambda \geq 0$.

In our study, three types of penalty functions, namely, Laplacian penalty, ridge penalty, and particularly, the proposed weakly supervised penalty will be considered, devised, and discussed in Section IV-B. Before going into the details of the penalized functions, more insight of the learned preimage is explored and interpreted below.

Define a function $\Theta_{s,\lambda,F}(P_\kappa \varphi(\mathbf{x})) = \sum_{i=1}^{s} w_i^{\mathbf{x}} \varphi(\tilde{\mathbf{x}}_i)$, where $w_i^{\mathbf{x}}$ are learned by criterion (16). Since $\sum_i w_i^{\mathbf{x}} = 1$ and $w_i^{\mathbf{x}} \geq 0$, we can estimate the following conditional probability[4]

$$P(\varphi(\tilde{\mathbf{x}}_i)|\Theta_{s,\lambda,F}(P_\kappa \varphi(\mathbf{x}))) = w_i^{\mathbf{x}}, \qquad i = 1, \ldots, s.$$

We call this the penalized probability relationship between $P_\kappa \varphi(\mathbf{x})$ and its neighbor $\varphi(\tilde{\mathbf{x}}_i)$, as $\Theta_{s,\lambda,F}(P_\kappa \varphi(\mathbf{x}))$ can be viewed as a penalized approximation value to $P_\kappa \varphi(\mathbf{x})$ given penalty function F and parameters s and λ. Also, as the preimage $\hat{\mathbf{x}}$ is found by additive combination of s neighboring preimages of $\varphi(\tilde{\mathbf{x}}_i)$ in (16), i.e., $\hat{\mathbf{x}} = \sum_{i=1}^{s} w_i^{\mathbf{x}} \tilde{\mathbf{x}}_i$, so we can also estimate the conditional probability $P(\tilde{\mathbf{x}}_i|\hat{\mathbf{x}}, s, \lambda, F) = w_i^{\mathbf{x}}$. Therefore, we have the following:

$$P(\tilde{\mathbf{x}}_i|\hat{\mathbf{x}}, s, \lambda, F) = w_i^{\mathbf{x}} = P(\varphi(\tilde{\mathbf{x}}_i)|\Theta_{s,\lambda,F}(P_\kappa \varphi(\mathbf{x}))). \quad (17)$$

In this sense, the penalized probability relationship between $P_\kappa \varphi(\mathbf{x})$ and its neighbors $\varphi(\tilde{\mathbf{x}}_i)$ in the feature space is preserved by the learned preimage in the input data space. Moreover, if it is assumed that the prior probabilities $P(\varphi(\tilde{\mathbf{x}}_i))$ and $P(\tilde{\mathbf{x}}_i)$ are the same,[5] i.e., $P(\varphi(\tilde{\mathbf{x}}_i)) = P(\tilde{\mathbf{x}}_i) = N^{-1}$, where N is the sample size, we can further have the following relationship:

$$\begin{aligned}&\mathrm{SI}(\varphi(\tilde{\mathbf{x}}_i), \Theta_{s,\lambda,F}(P_\kappa \varphi(\mathbf{x}))|s, \lambda, F) \\ &\quad = \log \frac{P(\varphi(\tilde{\mathbf{x}}_i), \Theta_{s,\lambda,F}(P_\kappa \varphi(\mathbf{x}))|s, \lambda, F)}{P(\varphi(\tilde{\mathbf{x}}_i))P(\Theta_{s,\lambda,F}(P_\kappa \varphi(\mathbf{x}))|s, \lambda, F)} \\ &\quad = \log \frac{P(\varphi(\tilde{\mathbf{x}}_i)|\Theta_{s,\lambda,F}(P_\kappa \varphi(\mathbf{x})))}{P(\varphi(\tilde{\mathbf{x}}_i))} \\ &\quad = \log (N w_i^{\mathbf{x}}) \\ &\quad = \log \frac{P(\tilde{\mathbf{x}}_i|\hat{\mathbf{x}}, s, \lambda, F)P(\hat{\mathbf{x}}|s, \lambda, F)}{P(\tilde{\mathbf{x}}_i)P(\hat{\mathbf{x}}|s, \lambda, F)} \\ &\quad = \mathrm{SI}(\tilde{\mathbf{x}}_i, \hat{\mathbf{x}}|s, \lambda, F). \qquad (18)\end{aligned}$$

Here

$$\begin{aligned}P(\varphi(\tilde{\mathbf{x}}_i)|\Theta_{s,\lambda,F}(P_\kappa \varphi(\mathbf{x})), s, \lambda, F) &= P(\varphi(\tilde{\mathbf{x}}_i)|\Theta_{s,\lambda,F}(P_\kappa \varphi(\mathbf{x}))) \\ P(\varphi(\tilde{\mathbf{x}}_i)|s, \lambda, F) &= P(\varphi(\tilde{\mathbf{x}}_i)) \\ P(\tilde{\mathbf{x}}_i|s, \lambda, F) &= P(\tilde{\mathbf{x}}_i).\end{aligned}$$

We call $\mathrm{SI}(\varphi(\tilde{\mathbf{x}}_i), \Theta_{s,\lambda,F}(P_\kappa \varphi(\mathbf{x}))|s, \lambda, F)$ the penalized pointwise conditional mutual information[6] between $P_\kappa \varphi(\mathbf{x})$ and its neighbor $\varphi(\tilde{\mathbf{x}}_i)$. This indicates that the penalized pointwise mutual information between $P_\kappa \varphi(\mathbf{x})$ and $\varphi(\tilde{\mathbf{x}}_i)$ in the feature space is preserved as $\mathrm{SI}(\tilde{\mathbf{x}}_i, \hat{\mathbf{x}}|s, \lambda, F)$ the pointwise conditional mutual information between the preimages $\hat{\mathbf{x}}$ and $\tilde{\mathbf{x}}_i$ in the input data space.

B. *The Penalty Function*

Since the best preimage is not known, we propose to make use of penalization to guide the learning process so that a better (if not the best) preimage can be obtained. In this paper, for performing penalization on the reconstruction weights, we introduce the use of Laplacian penalty and ridge penalty in preimage learning, and propose the weakly supervised penalty. The Laplacian penalty and ridge penalty are task independent and formulated based on prior common knowledge. The weakly supervised penalty is proposed for finding task-dependent preimage.

[4]For any $\mathbf{x}_j \notin U(\mathbf{x}, \delta)$, we can set $P(\varphi(\mathbf{x}_j)|\Theta_{s,\lambda,F}(P_\kappa \varphi(\mathbf{x}))) = 0$.

[5]This condition is mild, since such assumption is always held in many linear and kernel (nonlinear) systems.

[6]The mutual information between random variables $\mathbf{q}_1$ and $\mathbf{q}_2$ is the expectation of pointwise mutual information between the outcomes of them. So for any specific outcomes $\hat{\mathbf{q}}_1$ and $\hat{\mathbf{q}}_2$ of $\mathbf{q}_1$ and $\mathbf{q}_2$, the pointwise mutual information between them is defined as $\log\{p(\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2) \cdot [p(\hat{\mathbf{q}}_1)p(\hat{\mathbf{q}}_2)]^{-1}\}$.

$$\hat{\mathbf{x}} = \mathbf{\Psi}_{s,\mathbf{x}} \mathbf{w}_{\mathbf{x}} = \sum_{i=1}^{s} w_i^{\mathbf{x}} \tilde{\mathbf{x}}_i, \qquad \text{where} \quad \begin{cases} \mathbf{w}_{\mathbf{x}} = \displaystyle\arg\min_{\mathbf{w}=(w_1,\ldots,w_s)^T \in \Re^s} \left\{ \mathbf{w}^T \mathbf{\Psi}_{s,\varphi(\mathbf{x})}^T \mathbf{\Psi}_{s,\varphi(\mathbf{x})} \mathbf{w} - 2(\mathbf{\Psi}_\varphi \boldsymbol{\gamma}^{\mathbf{x}})^T \mathbf{\Psi}_{s,\varphi(\mathbf{x})} \mathbf{w} + \lambda \cdot F(\mathbf{w}) \right\} \\ \text{s.t.} \sum_{i=1}^{s} w_i^{\mathbf{x}} = 1 \quad \text{and} \quad w_i^{\mathbf{x}} \geq 0, \qquad i = 1, \ldots, s \end{cases} \quad (16)$$

1) Laplacian Penalty: When the preimage value is an image vector, Laplacian penalty can impose appearance smoothness constraint on the image. Smoothness is conceived to be a generic assumption on image appearance, characterizing the coherence and homogeneity [18], [19]. Suppose $\hat{\mathbf{x}}$ is the vector representation of image matrix $\hat{\mathbf{x}}_{2d}$ in row order and let a and b be the horizontal and vertical coordinates in an image. Laplacian penalty is formulated as [19]

$$F(\mathbf{w}) = \int_{a,b} \left[\frac{\partial^2 \hat{\mathbf{x}}_{2d}}{\partial a^2} + \frac{\partial^2 \hat{\mathbf{x}}_{2d}}{\partial b^2} \right]^2 da\, db. \tag{19}$$

For computation, the discrete form of Laplacian penalty is always used for approximation [19] and (19) can be approximated as follows:

$$F(\mathbf{w}) = \hat{\mathbf{x}}^T \boldsymbol{\Omega} \hat{\mathbf{x}} = \mathbf{w}^T \boldsymbol{\Psi}_{s,\mathbf{x}}^T \boldsymbol{\Omega} \boldsymbol{\Psi}_{s,\mathbf{x}} \mathbf{w} \tag{20}$$

where $\boldsymbol{\Omega} = \Delta^T \Delta, \Delta = \mathbf{D}_a \otimes \mathbf{I}_b + \mathbf{I}_a \otimes \mathbf{D}_b$, $\mathbf{D}_a$, and $\mathbf{D}_b$ are matrices that approximate a second derivative by second difference [19]. Then, the objective function becomes (21), shown at the bottom of the page.

2) Ridge Penalty: An extension of the penalty form of Laplacian penalty could be to find a metric matrix $\mathbf{C}$ such that $F(\mathbf{w}) = \mathbf{w}^T \mathbf{C} \mathbf{w}$. In particular, letting $\mathbf{C} = \mathbf{I}$ would yield the well-known ridge penalty, i.e., $F(\mathbf{w}) = \mathbf{w}^T \mathbf{w}$. Then, the objective function for minimization becomes (22), shown at the bottom of the page.

Unlike Laplacian penalty, ridge penalty used in our case addresses the smoothness of geometry structure between $P_\kappa \varphi(\mathbf{x})$ and its neighbors $\varphi(\tilde{\mathbf{x}}_i)$ and also between $\hat{\mathbf{x}}$ and $\tilde{\mathbf{x}}_i$. Ridge penalty would make the weights $w_i^{\mathbf{x}}$ not differ too much, because $\sum_{i=1}^s (w_i^{\mathbf{x}})^2$ is minimized subject to $\sum_{i=1}^s w_i^{\mathbf{x}} = 1$ and $w_i^{\mathbf{x}} \geq 0$ if and only if $w_1^{\mathbf{x}} = \cdots = w_s^{\mathbf{x}}$. This penalty would prevent any $\tilde{\mathbf{x}}_i$ from dominating the value of $\hat{\mathbf{x}}$ and may be helpful to make the learned preimage robust to noise, since more information from other training samples is used. As the exact preimage of a feature vector is difficult to be determined, properly using more information may sometimes be useful for getting a better estimation.

3) Weakly Supervised Penalty: It is found that no supervised information is used in the former two penalty functions. However, KPCA is an unsupervised technique and class labels are not available. For this reason, we propose a way to incorporate some weakly supervised prior knowledge if it is available. The weakly supervised knowledge means only positive class information and negative class information are available and the exact class labels of samples are indeed unknown. In this paper, positive class is defined as the sample set which the preimage is expected to be close to and the negative class is defined as the sample set which the preimage is expected to be far away from.

Denote the positive class by $C^+ = \{\mathbf{z}_1^+, \ldots, \mathbf{z}_{N_+}^+\}$ and the negative class by $C^- = \{\mathbf{z}_1^-, \ldots, \mathbf{z}_{N_-}^-\}$, where $\mathbf{z}_i^+$ and $\mathbf{z}_j^-$ are samples in the positive class and the negative class, respectively. Here $\mathbf{z}_i^+$ and $\mathbf{z}_j^-$ are not restricted to be out of the training samples for learning KPCA. We here assume these prior information is available. As show in the experiment, such weakly supervised knowledge may not be difficult to be obtained in most applications. We also give discussions on this issue at the end of Section VI. For penalization, the learned preimage is expected close to the local positive class information and far away from the local negative class information. Therefore, we design the weakly supervised penalty as follows:

$$\begin{aligned}
F(\mathbf{w}) &= \eta^+ |H_\theta^+(\mathbf{x})|^{-1} \left[\sum_{\mathbf{z}^+ \in H_\theta^+(\mathbf{x})} \|\boldsymbol{\Psi}_{s,\mathbf{x}} \mathbf{w} - \mathbf{z}^+\|^2 \right] \\
&\quad - \eta^- |H_\theta^-(\mathbf{x})|^{-1} \left[\sum_{\mathbf{z}^- \in H_\theta^-(\mathbf{x})} \|\boldsymbol{\Psi}_{s,\mathbf{x}} \mathbf{w} - \mathbf{z}^-\|^2 \right] \\
&= \eta^+ |H_\theta^+(\mathbf{x})|^{-1} \left[\sum_{\mathbf{z}^+ \in H_\theta^+(\mathbf{x})} \mathbf{w}^T \boldsymbol{\Psi}_{s,\mathbf{x}}^T \boldsymbol{\Psi}_{s,\mathbf{x}} \mathbf{w} \right. \\
&\qquad \left. - 2\mathbf{z}^{+T} \boldsymbol{\Psi}_{s,\mathbf{x}} \mathbf{w} + \mathbf{z}^{+T} \mathbf{z}^+ \right] \\
&\quad - \eta^- |H_\theta^-(\mathbf{x})|^{-1} \left[\sum_{\mathbf{z}^- \in H_\theta^-(\mathbf{x})} \mathbf{w}^T \boldsymbol{\Psi}_{s,\mathbf{x}}^T \boldsymbol{\Psi}_{s,\mathbf{x}} \mathbf{w} \right. \\
&\qquad \left. - 2\mathbf{z}^{-T} \boldsymbol{\Psi}_{s,\mathbf{x}} \mathbf{w} + \mathbf{z}^{-T} \mathbf{z}^- \right] \\
&= (\eta^+ - \eta^-) \mathbf{w}^T \boldsymbol{\Psi}_{s,\mathbf{x}}^T \boldsymbol{\Psi}_{s,\mathbf{x}} \mathbf{w} \\
&\quad - \left(2\eta^+ |H_\theta^+(\mathbf{x})|^{-1} \sum_{\mathbf{z}^+ \in H_\theta^+(\mathbf{x})} \mathbf{z}^{+T} \right. \\
&\qquad \left. - 2\eta^- |H_\theta^-(\mathbf{x})|^{-1} \sum_{\mathbf{z}^- \in H_\theta^-(\mathbf{x})} \mathbf{z}^{-T} \right) \boldsymbol{\Psi}_{s,\mathbf{x}} \mathbf{w}
\end{aligned}$$

$$\begin{aligned}
\mathbf{w}_{\mathbf{x}} = &\arg\min_{\mathbf{w}=(w_1,\ldots,w_s)^T} \left\{ \mathbf{w}^T \left(\boldsymbol{\Psi}_{s,\varphi(\mathbf{x})}^T \boldsymbol{\Psi}_{s,\varphi(\mathbf{x})} + \lambda \cdot \boldsymbol{\Psi}_{s,\mathbf{x}}^T \boldsymbol{\Omega} \boldsymbol{\Psi}_{s,\mathbf{x}} \right) \mathbf{w} - 2(\boldsymbol{\Psi}_\varphi \boldsymbol{\gamma}^{\mathbf{x}})^T \boldsymbol{\Psi}_{s,\varphi(\mathbf{x})} \mathbf{w} \right\} \\
&\text{s.t.} \sum_{i=1}^s w_i^{\mathbf{x}} = 1 \quad \text{and} \quad w_i^{\mathbf{x}} \geq 0, \qquad i = 1, \ldots, s.
\end{aligned} \tag{21}$$

$$\begin{aligned}
\mathbf{w}_{\mathbf{x}} = &\arg\min_{\mathbf{w}=(w_1,\ldots,w_s)^T} \left\{ \mathbf{w}^T (\boldsymbol{\Psi}_{s,\varphi(\mathbf{x})}^T \boldsymbol{\Psi}_{s,\varphi(\mathbf{x})} + \lambda \cdot \mathbf{I}) \mathbf{w} - 2(\boldsymbol{\Psi}_\varphi \boldsymbol{\gamma}^{\mathbf{x}})^T \boldsymbol{\Psi}_{s,\varphi(\mathbf{x})} \mathbf{w} \right\} \\
&\text{s.t.} \quad \sum_{i=1}^s w_i^{\mathbf{x}} = 1 \quad \text{and} \quad w_i^{\mathbf{x}} \geq 0, \qquad i = 1, \ldots, s.
\end{aligned} \tag{22}$$

$$+\left(\eta^+|H_\theta^+(\mathbf{x})|^{-1}\sum_{\mathbf{z}^+\in H_\theta^+(\mathbf{x})}\mathbf{z}^{+T}\mathbf{z}^+ -\eta^-|H_\theta^-(\mathbf{x})|^{-1}\sum_{\mathbf{z}^-\in H_\theta^-(\mathbf{x})}\mathbf{z}^{-T}\mathbf{z}^-\right) \quad (23)$$

where $H_\theta^+(\mathbf{x}) = \{\mathbf{z}^+|\mathbf{z}^+ \in C^+, \varphi(\mathbf{z}^+)$ is one of the θ^+ nearest positive samples of $P_\kappa\phi(\mathbf{x})\}$, $H_\theta^-(\mathbf{x}) = \{\mathbf{z}^-|\mathbf{z}^- \in C^-, \varphi(\mathbf{z}^-)$ is one of the θ^- nearest negative samples of $P_\kappa\varphi(\mathbf{x})\}, \eta^+ \geq 0, \eta^- \geq 0$, and $|H_\theta^+(\mathbf{x})|$ and $|H_\theta^-(\mathbf{x})|$ are the cardinalities of $H_\theta^+(\mathbf{x})$ and $H_\theta^-(\mathbf{x})$, respectively. Define

$$\mathbf{z}_\mathbf{x}^{\eta^+,\eta^-,\theta} = 2\eta^+|H_\theta^+(\mathbf{x})|^{-1}\sum_{\mathbf{z}^+\in H_\theta^+(\mathbf{x})}\mathbf{z}^{+T} -2\eta^-|H_\theta^-(\mathbf{x})|^{-1}\sum_{\mathbf{z}^-\in H_\theta^-(\mathbf{x})}\mathbf{z}^{-T}.$$

Then, ignoring the last term independent of $\mathbf{w}$ in (23) would yield criterion (24), shown at the bottom of the page, where we let $\eta^+ \leftarrow \lambda\eta^+$ and $\eta^- \leftarrow \lambda\eta^-$ for simplification in the above criterion.

For utilizing the weakly supervised penalty function, it is required to properly design the positive and negative information depending on the applications.

To solve the criterion (24) as well as (21) and (22), a simple quadratic program could be used to get the optimal solution. Sometimes, when the training set of KPCA is the same as the positive sample set, we would set $\eta^+ = 0$ in (24), as the preimage is already the combination of positive samples by utilizing the criterion (16).

V. Experimental Results

In this section, by utilizing the proposed preimage learning algorithm as well as existing techniques, KPCA was applied to four different image preprocessing applications on face images, namely facial expression normalization, image denoising, occlusion recovery, and illumination normalization. Human face image analysis has been received great attractions in the last few years. To measure the quality of the processed images, this paper adopts the mean square error (MSE) measurement similar as in [5]–[7]. The visual processing results are also illustrated for comparison.

A. Experiment Settings

1) Data Sets: Two data sets, namely Cohn–Kanade facial expression (CKFE) database [20] and YALEB [21] database were used for experiments. CKFE includes image sequences of different persons, and each sequence describes the variations from natural expression to a particular facial expression. The numbers of images in different sequences are different. The available portion supplied by Carnegie Mellon University (CMU) was used in this paper. We cropped all face images from the original images manually. In total, there are 8795 images of different kinds of expressions from 97 persons, including surprise, fear, joy, sadness, disgust, and anger. YALEB consists of ten persons with nine different poses. For each pose, there are 65 face images undergoing various illuminations. Face images of each pose in YALEB are divided into five subsets [21] according to the light-source directions. The light-source directions of images in subset 1 are frontal or nearly frontal. In the experiment, each image was aligned and resized to 60×80.

2) Kernel Function: In the experiments, all images were linearly stretched to full range of pixel values of $[0, 1]$. The performances of preimage learning algorithms are reported using two kernel functions, namely RBF kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-c^{-1}\|\mathbf{x}_i - \mathbf{x}_j\|^2)$ and polynomial kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (<\mathbf{x}_i, \mathbf{x}_j> +\theta)^d$. For RBF kernel, we set $c = 10^4$; for polynomial kernel, we set $\theta = 0.1$ and $d = 0.5$. In all experiments, KPCA was learned on training set (will be specified later) and a KPCA subspace was determined by retaining the largest kernel principal components that were selected to preserve 95% energy.

3) Design of Experiments: For evaluation, we applied KPCA with preimage learning in four applications: facial expression normalization, face image denoising, recovery of occluded face images, and illumination normalization of face images. Experiment settings are as follows.

a) Expression normalization: CKFE database was used. The first three images from each sequence of facial expression were used for training, since they are nearly natural facial expression. The rest of images in the sequence except the last image were used for testing. The last image will be used later. The last two images of a sequence were captured at different time, albeit similar with the exact facial expression. So there were 1461 training images and 6847 testing images. Note that there is no unique ground truth natural facial expression for each person in CKFE, but a set of training images of nearly natural facial expression is available. The MSE value of a normalized facial expression image was then obtained by computing the minimum MSE between it and all training images of the same person.

b) Image denoising: The experiment is conducted on subset 1 of YALEB database, of which images were captured under normal or nearly normal illumination condition. In that subset, for each pose, there are seven images for each person. We established the training set by randomly selecting six images from each pose for each person in the subset 1 to train a KPCA model, and the rest one image of each pose from

$$\mathbf{w}_\mathbf{x} = \underset{\mathbf{w}=(w_1,\ldots,w_s)^T}{\arg\min}\left\{\mathbf{w}^T\left(\mathbf{\Psi}_{s,\varphi(\mathbf{x})}^T\mathbf{\Psi}_{s,\varphi(\mathbf{x})} + (\eta^+ - \eta^-)\mathbf{\Psi}_{s,\mathbf{x}}^T\mathbf{\Psi}_{s,\mathbf{x}}\right)\mathbf{w} - \left[2(\mathbf{\Psi}_\varphi\boldsymbol{\gamma}^\mathbf{x})^T\mathbf{\Psi}_{s,\varphi(\mathbf{x})} + \mathbf{z}_\mathbf{x}^{\eta^+,\eta^-,\theta}\mathbf{\Psi}_{s,\mathbf{x}}\right]\mathbf{w}\right\}$$
$$\text{s.t.} \quad \sum_{i=1}^s w_i^\mathbf{x} = 1 \quad \text{and} \quad w_i^\mathbf{x} \geq 0, \qquad i = 1,\ldots,s, \quad (24)$$

TABLE I
MSE: CONVEXITY CONSTRAINT IN $\mathrm{P}^2\mathrm{L}$, NORMALIZED FACIAL EXPRESSION (CKFE)

Criterion	RBF				Polyn.			
	s=3	s=5	s=10	s=15	s=3	s=5	s=10	s=15
Non-convex Criterion (15)	13.079	14.427	16.74	18.598	13.398	14.754	17.174	19.466
Non-convex Criterion (15), CUT	13.07	14.416	16.719	18.566	13.381	14.733	17.14	19.417
Criterion (15)	**12.42**	**13.206**	**14.22**	**14.917**	**12.696**	**13.177**	**14.062**	**14.927**

each person was treated as the clean testing image and two testing noisy images were generated by adding noise on each clean testing image. This procedure repeated ten times and the average MSE of each preimage learning method was obtained, where totally 1800 testing noisy images were processed over the ten run experiment. Note that only Gaussian noise with zero mean and random variance between 0 and 1 was used.

c) Reconstruction of occluded images: The experiment setting is almost the same as the one for image denoising except the only change that each occluded image was generated by placing a rectangle black patch onto each clean testing image at a random coordinate. The width and height of the patch were random, and they varied from 15 to 40 in pixels.

d) Illumination normalization of face images: Similarly, all images in subset 1 were used for KPCA learning, as they are under or almost under normal illumination condition. All images from nine poses in subset 2 and subset 3, in total 2160 images, were used for testing. For evaluation, each normalized face image was compared with the corresponding face image under normal illumination condition.

B. Evaluation of the Proposed Method

This section evaluates the proposed method. In this paper, two kinds of penalizations are used in $\mathrm{P}^2\mathrm{L}$, i.e., the convexity constraint for learning the combination weights $w_i^{\mathbf{x}}$ and the penalty function $F(\mathbf{w})$. Under the developed two-step framework, these penalizations yield criteria (15) and (16), respectively. In Section V-B1, we first evaluate the usefulness of convexity constraint in criterion (15); in Section V-B2, the effects of different penalty functions are demonstrated.

1) Evaluation of Convexity Constraint in Criterion (15): Besides some statistical interpretation in Section IV, this part experimentally justifies the usefulness of convexity constraint. The comparison between criterion (15) and "nonconvex criterion (15)" is reported in Tables I–IV. The "nonconvex criterion (15)" means the constraint "$\sum_{i=1}^{s} w_i^{\mathbf{x}} = 1$ and $w_i^{\mathbf{x}} \geq 0$" in criterion (15) is removed and is a special case of our previous algorithm [8]. Except the case of denoising on YALEB, the results show that the MSEs are overall lower[7] when the convexity constraint is applied. Although the nonconvex version is a bit better than criterion (15) for image denoising on YALEB, we will particularly show later that criterion (15) plus weakly supervised penalty would always obtain a notable improvement with (much) lower MSE than "nonconvex criterion (15)." In addition, we conducted the "nonconvex criterion (15) CUT," which truncated the results obtained by "nonconvex criterion (15)" in order to make the preimage well-defined directly. As shown, the truncation only gives slight improvements over "nonconvex criterion (15)." This suggests that the well-defined property had better be considered during the optimization process of the proposed preimage learning criteria, since the convexity constraint can also help learn some pointwise conditional mutual information that is difficult to be realized by the truncation process. Next section will further justify its usefulness in another proposed criterion (16) when weakly supervised penalty is used. In a word, the experimental results indicate that the convexity constraint is overall useful in two proposed criteria.

[7]Note that the pixel value of each image is defined in $[0, 1]$. If it is defined in $[0, 255]$, then all MSEs reported have to be multiplied by 255^2.

2) Evaluation of the Proposed Penalty Function: This section evaluates the effects of using penalty functions. The goal is to show that $\mathrm{P}^2\mathrm{L}$ with proper penalty function can enhance the first proposed criterion (15) and therefore get a much better preimage.

a) Parameters/penalty function setting: We report the results of $\mathrm{P}^2\mathrm{L}$ with different values of λ the importance weight and s the number of nearest neighbors, where $\lambda \in \{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ and $s \in \{3, 5, 10, 15\}$. For implementation of weakly supervised penalty function, we set $\theta^+ = \theta^- = s$ in (24). As stated in Section IV-B, we assume the prior knowledge, i.e., positive and negative class samples for learning the weakly supervised penalty are available. The selection of negative class samples in the experiments are described as follows.

1) For facial expression normalization, the last image of each facial expression image sequence was treated as the negative sample. In total, there were 487 negative class samples.
2) For face image denoising, the noisy samples in the negative class for each run were additionally generated on the training set; that is, two noisy images were generated on each image in the training set using Gaussian noise with zero mean and random variance. So there were 1080 negative class samples for each run.
3) For reconstruction of occluded face images, samples of the negative class were also additionally generated on the training data for each run; that is, for each face image in the training set, two occluded face images were generated with rectangle black patch randomly placed in the image. So there were also 1080 negative class samples for training for each run.
4) For illumination normalization, we only simply selected all illuminated images in subset 5 from nine poses of YALEB database as the negative samples. Those images are with extremely serious illumination. So, there were 1800 images treated as the negative class data.

It should be noted that there was no overlap between the negative data set and testing data set.

Besides, since the training samples of KPCA were good in our experiments, it can therefore be viewed that the positive

TABLE II
MSE: CONVEXITY CONSTRAINT IN $\mathrm{P}^2\mathrm{L}$, DENOISING (YALEB)

Criterion	RBF				Polyn.			
	s=3	s=5	s=10	s=15	s=3	s=5	s=10	s=15
Non-convex Criterion (15)	39.448	38.123	38.461	39.199	39.882	39.161	40.478	41.74
Non-convex Criterion (15), CUT	**39.448**	**38.122**	**38.459**	**39.195**	39.882	39.16	40.476	41.737
Criterion (15)	40.3	38.979	39.149	39.479	40.96	39.769	**39.758**	**39.967**

TABLE III
MSE: CONVEXITY CONSTRAINT IN $\mathrm{P}^2\mathrm{L}$, OCCLUSION RECOVERY (YALEB)

Criterion	RBF				Polyn.			
	s=3	s=5	s=10	s=15	s=3	s=5	s=10	s=15
Non-convex Criterion (15)	34.951	34.148	36.286	38.549	30.799	29.337	29.869	31.314
Non-convex Criterion (15), CUT	34.946	34.127	36.22	38.448	30.783	29.305	29.797	31.197
Criterion (15)	**34.63**	**33.258**	**32.727**	**32.851**	**28.138**	**27.062**	**26.983**	**27.245**

TABLE IV
MSE: CONVEXITY CONSTRAINT IN $\mathrm{P}^2\mathrm{L}$, ILLUMINATION NORMALIZATION (YALEB)

Criterion	RBF				Polyn.			
	s=3	s=5	s=10	s=15	s=3	s=5	s=10	s=15
Non-convex Criterion (15)	67.792	71.656	84.983	93.553	69.315	74.944	87.946	96.483
Non-convex Criterion (15), CUT	67.726	71.479	84.345	92.567	69.113	74.489	86.795	94.825
Criterion (15)	**65.283**	**64.088**	**64.358**	**64.769**	**62.151**	**61.356**	**61.469**	**61.872**

information (i.e., training samples) has already been used. So, as stated in (24), we did not employ additional positive data set; that is, we set η^+ to be 0 in (24) and only let the parameter η^- be active in (24). In the experiment, we also let $\eta^- \in \{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. Please note that we only tuned η^- for weakly supervised penalty, as it is written $\eta^- \leftarrow \lambda\eta^-$ in (24). Though λ and η^- were set small, we would show that the proposed method can get superior performance to the popular preimage learning methods in the next section.

b) Experimental analysis: Results of the MSE performances of penalized preimage learning $(\mathrm{P}^2\mathrm{L})$ with different penalty functions are tabulated in Tables V–VIII, where the lowest MSEs with respect to each parameter value of λ or η^- across different configurations of penalty functions are shown in bold. From Tables IX–XII, the comparison results between two proposed criteria (15) and (16) are shown, where only the penalty on data range is used in criterion (15). For fixed parameter s, the MSE results of using penalty functions are the corresponding best ones reported in Tables V–VIII. In Tables IX–XII, for each parameter s, the lowest MSE value is shown in bold.

With the results shown in Tables V–VIII, we find that utilizing weakly supervised penalty could be overall better, because it always gets lower MSE except several cases when λ or η^- equals to 10^{-3} using RBF kernel as shown in Tables VI–VIII. Moreover, results in Tables IX–XII show $\mathrm{P}^2\mathrm{L}$ using weakly supervised penalty can always get the lowest MSE values. Compared to another proposed criterion (15), using weakly supervised penalty function can enhance the performance of $\mathrm{P}^2\mathrm{L}$ in all applications and using Laplacian penalty and ridge penalty can mainly enhance the performances in face illumination normalization. On the one hand, it is true that a further integrated penalty function has the chances to make $\mathrm{P}^2\mathrm{L}$ get a further better preimage; on the other hand, compared with the other two penalty functions, it is suggested that weakly supervised penalty, a task-dependent penalty, would be more useful and effective, though it is still difficult to prove that it would always perform the best. The reason may be because some weakly supervised information that defines the positive and negative information is used, while Laplacian and ridge penalties are task independent. Although weakly supervised penalty depends on the positive and negative information used as prior knowledge, as shown here it is sometimes not difficult to obtain them. More discussions on this issue will be given in Section VI. Additionally, the success of using weakly supervised penalty shows that the negative information from data could be useful, while negative information is not used in preimage map.

Finally, as weakly supervised penalty is preferred in general, we additionally implemented "nonconvex criterion (16)" and "non-convex criterion (16) CUT" with weakly supervised penalty $(\eta^- = 10^{-4})$ where the convexity constraint in criterion (16) is removed and the truncation process is further applied, respectively. By comparing the results of nonconvex criterion (15) from Tables I–IV with the results in Table XIII, we show that under the same parameter setting weakly supervised penalty can make much more improvements when convexity constraint is imposed in criterion (16); otherwise, the improvement may be slight or not happen. Moreover, when convexity constraint is used, the performance of weakly supervised penalty would be more stable, as the MSE changes relatively small across different numbers of neighbors, as compared to the case when convexity constraint is removed. The reason for this could be because weakly supervised penalty would make the preimage be placed away from its nearby negative samples, but it might also simultaneously make the preimage be out of data range as the direction of the movement of preimage value could not be accurately specified. So, the convexity constraint actually restricts the variation range of preimage; that is, some regularization is

TABLE V
MSEs of Normalized Facial Expression Images Using Penalized Preimage Learning

MSE	λ or	Laplacian Penalty				Ridge Penalty				Weakly Supervised Penalty			
	η^-	s=3	s=5	s=10	s=15	s=3	s=5	s=10	s=15	s=3	s=5	s=10	s=15
RBF	10^{-3}	12.657	15.901	19.066	20.988	12.515	13.382	14.481	15.22	**12.383**	12.818	12.774	13.032
	10^{-4}	12.57	14.377	16.16	17.205	12.452	13.251	14.271	14.973	12.011	11.592	**11.417**	11.456
	10^{-5}	12.446	13.381	14.492	15.23	12.424	13.212	14.226	14.924	**12.399**	13.073	13.959	14.576
	10^{-6}	12.423	13.225	14.248	14.95	12.42	13.207	14.221	14.918	**12.418**	13.193	14.194	14.884
Polyn.	10^{-3}	12.739	13.491	14.558	15.499	12.703	13.187	14.074	14.94	**12.651**	12.921	13.558	14.26
	10^{-4}	12.701	13.211	14.115	14.988	12.697	13.178	14.063	14.928	**12.692**	13.152	14.013	14.861
	10^{-5}	12.697	13.18	14.067	14.933	**12.696**	13.177	14.062	14.927	**12.696**	13.174	14.057	14.921
	10^{-6}	**12.696**	13.177	14.062	14.928	**12.696**	13.177	14.062	14.927	**12.696**	13.177	14.061	14.926

TABLE VI
MSEs of Denoised YaleB Images Using Penalized Preimage Learning

MSE	λ or	Laplacian Penalty				Ridge Penalty				Weakly Supervised Penalty			
	η^-	s=3	s=5	s=10	s=15	s=3	s=5	s=10	s=15	s=3	s=5	s=10	s=15
RBF	10^{-3}	41.473	41.169	43.389	45.094	40.341	**38.947**	39.158	39.532	53.053	52.434	53.44	54.592
	10^{-4}	40.687	39.662	40.486	41.214	40.3	38.969	39.145	39.481	40.925	38.323	36.536	**35.886**
	10^{-5}	40.322	39.038	39.305	39.686	40.3	38.978	39.148	39.479	40.183	38.73	**38.71**	38.935
	10^{-6}	40.302	38.984	39.164	39.5	40.3	38.979	39.149	39.479	40.288	**38.953**	39.104	39.425
Polyn.	10^{-3}	40.97	39.831	39.943	40.21	40.96	39.768	39.758	39.968	40.788	39.422	**39.146**	39.189
	10^{-4}	40.959	39.773	39.776	39.991	40.96	39.769	39.758	39.967	40.942	39.733	**39.696**	39.887
	10^{-5}	40.96	39.769	39.76	39.969	40.96	39.769	39.758	39.967	40.959	39.765	**39.752**	39.959
	10^{-6}	40.96	39.769	39.758	39.967	40.96	39.769	39.758	39.967	40.96	39.769	**39.758**	39.966

TABLE VII
MSEs of Reconstructed Occluded Images Using Penalized Preimage Learning

MSE	λ or	Laplacian Penalty				Ridge Penalty				Weakly Supervised Penalty			
	η^-	s=3	s=5	s=10	s=15	s=3	s=5	s=10	s=15	s=3	s=5	s=10	s=15
RBF	10^{-3}	37.857	38.857	42.203	44.61	34.623	33.243	**32.785**	32.966	47.375	47.205	47.996	47.182
	10^{-4}	35.631	34.954	35.447	36.051	34.618	33.246	32.725	32.857	34.401	31.573	28.506	**27.378**
	10^{-5}	34.726	33.442	33.033	33.216	34.629	33.256	32.726	32.851	34.466	32.99	**32.285**	32.308
	10^{-6}	34.639	33.275	32.757	32.887	34.63	33.258	32.727	32.851	34.614	33.231	**32.683**	32.798
Polyn.	10^{-3}	28.225	27.3	27.391	27.723	28.136	27.06	26.983	27.246	27.869	26.593	**26.226**	26.346
	10^{-4}	28.145	27.084	27.023	27.293	28.138	27.061	26.983	27.245	28.111	27.015	**26.909**	27.157
	10^{-5}	28.139	27.064	26.987	27.25	28.138	27.062	26.983	27.245	28.135	27.057	**26.976**	27.236
	10^{-6}	28.138	27.062	26.984	27.246	28.138	27.062	26.983	27.245	28.138	27.061	**26.982**	27.244

TABLE VIII
MSEs of Normalized Illuminated Images Using Penalized Preimage Learning

MSE	λ or	Laplacian Penalty				Ridge Penalty				Weakly Supervised Penalty			
	η^-	s=3	s=5	s=10	s=15	s=3	s=5	s=10	s=15	s=3	s=5	s=10	s=15
RBF	10^{-3}	63.9	**62.053**	63.533	65.214	64.674	63.455	63.9	64.387	69.353	66.502	65.666	66.6
	10^{-4}	64.289	63.094	64.134	64.987	65.212	64.019	64.311	64.731	64.025	60.605	58.131	**57.346**
	10^{-5}	65.115	63.942	64.324	64.792	65.275	64.081	64.353	64.765	64.898	63.448	**63.369**	63.614
	10^{-6}	65.265	64.073	64.354	64.771	65.282	64.087	64.357	64.768	65.243	**64.023**	64.259	64.653
Polyn.	10^{-3}	61.927	61.163	61.446	61.915	62.142	61.348	61.464	61.867	61.743	60.698	**60.423**	60.631
	10^{-4}	62.129	61.336	61.467	61.876	62.15	61.355	61.469	61.871	62.106	**61.285**	61.356	61.739
	10^{-5}	62.149	61.354	61.469	61.872	62.151	61.356	61.469	61.872	62.147	**61.349**	61.458	61.858
	10^{-6}	62.151	61.356	61.469	61.872	62.151	61.356	61.469	61.872	62.151	**61.355**	61.468	61.87

implicitly imposed to alleviate this potential problem. Hence, the convexity constraint is not only able to ensure a well-defined preimage, but also help regularize the effect of weakly supervised information.

3) The Effect of the Number of Nearest Neighbors: This part shows that using local information could be already satisfactory in the two proposed criteria (15) and (16) by extensively evaluating its effect.

We first address the case when the penalty term in the optimization function G does not take effect, i.e., λ or η^- equals to zero. Table XIV reports the results and the lowest MSE value in each row is in boldface. It can be seen that $\mathrm{P}^2\mathrm{L}$ obtains the lowest MSE when s is small (e.g., $s \leq 10$) and degrades as s increases, for both RBF kernel and polynomial kernel. Note that the derivation of criterion (15) is partially inspired by the assumption in LLE, i.e., locality preserving. Without any effect

TABLE IX
MSE: CRITERION (15) VERSUS CRITERION (16) IN $\mathrm{P^2L}$ FOR NORMALIZED FACIAL EXPRESSION IMAGES

Penalty Function	RBF				Polyn.			
	s=3	*s*=5	*s*=10	*s*=15	*s*=3	*s*=5	*s*=10	*s*=15
Criterion (15)	12.42	13.206	14.22	14.917	12.696	13.177	14.062	14.927
Criterion (16), Laplacian Penalty	12.423	13.225	14.248	14.95	12.696	13.177	14.062	14.928
Criterion (16), Ridge Penalty	12.42	13.207	14.221	14.918	12.696	13.177	14.062	14.927
Criterion (16), Weakly Supervised Penalty	**12.011**	**11.592**	**11.417**	**11.456**	**12.651**	**12.921**	**13.558**	**14.26**

TABLE X
MSE: CRITERION (15) VERSUS CRITERION (16) IN $\mathrm{P^2L}$ FOR IMAGE DENOISING (YALEB)

Penalty Function	RBF				Polyn.			
	s=3	*s*=5	*s*=10	*s*=15	*s*=3	*s*=5	*s*=10	*s*=15
Criterion (15)	40.3	38.979	39.149	39.479	40.96	39.769	39.758	39.967
Criterion (16), Laplacian Penalty	40.302	38.984	39.164	39.5	40.959	39.769	39.758	39.967
Criterion (16), Ridge Penalty	40.3	38.947	39.145	39.479	40.96	39.768	39.758	39.967
Criterion (16), Weakly Supervised Penalty	**40.183**	**38.323**	**36.536**	**35.886**	**40.788**	**39.422**	**39.146**	**39.189**

TABLE XI
MSE: CRITERION (15) VERSUS CRITERION (16) IN $\mathrm{P^2L}$ FOR RECONSTRUCTION OF OCCLUDED IMAGES (YALEB)

Penalty Function	RBF				Polyn.			
	s=3	*s*=5	*s*=10	*s*=15	*s*=3	*s*=5	*s*=10	*s*=15
Criterion (15)	34.63	33.258	32.727	32.851	28.138	27.062	26.983	27.245
Criterion (16), Laplacian Penalty	34.639	33.275	32.757	32.887	28.138	27.062	26.984	27.246
Criterion (16), Ridge Penalty	34.618	33.243	32.725	32.851	28.136	27.06	26.983	27.245
Criterion (16), Weakly Supervised Penalty	**34.401**	**31.573**	**28.506**	**27.378**	**27.869**	**26.593**	**26.226**	**26.346**

TABLE XII
MSE: CRITERION (15) VERSUS CRITERION (16) IN $\mathrm{P^2L}$ FOR ILLUMINATION NORMALIZATION (YALEB)

Penalty Function	RBF				Polyn.			
	s=3	*s*=5	*s*=10	*s*=15	*s*=3	*s*=5	*s*=10	*s*=15
Criterion (15)	65.283	64.088	64.358	64.769	62.151	61.356	61.469	61.872
Criterion (16), Laplacian Penalty	**63.9**	62.053	63.533	64.771	61.927	61.163	61.446	61.872
Criterion (16), Ridge Penalty	64.674	63.455	63.9	64.387	62.142	61.348	61.464	61.867
Criterion (16), Weakly Supervised Penalty	64.025	**60.605**	**58.131**	**57.346**	**61.743**	**60.698**	**60.423**	**60.631**

of any penalty function, this case does exactly evaluate the feasibility of using local but not global information for learning preimage.

Consider criterion (16) where penalty function is used. We consider the unsupervised penalty functions, i.e., the ridge and Laplacian penalties. As these two penalty functions currently perform better for illumination normalization, we mainly present their results in this case. As shown in Tables XV and Table XVI, when quite fewer neighbors are used, the best results are obtained.

The above two cases indicate that when only unsupervised information is used, using appropriately quite limited local information is sufficient. This will also coincide with the case of distance constraint reported later, which also only uses unsupervised information.

Now, consider the case using weakly supervised penalty ($\eta^- = 10^{-4}$). Results are shown in Table XVII. Since some weakly supervised information is integrated, the scenario could be different. For polynomial kernel, a small s is still preferred; for RBF kernel, the scenario becomes a bit complicated. However, we may still get some experimental observations that if more facial structures are preserved, less local information may be probably sufficient when weakly supervised information is used for RBF kernel. The facial expression variation may destroy facial structures relatively slightly, as much more holistic facial structures are still preserved. In contrast, noise, occlusion, and illumination would make some structures missed. However, it should be noted that even though more neighbors are used for RBF in this case, s is still much smaller than the amount of training samples.

Therefore, when the model becomes less unsupervised, a bit more neighbors may be needed sometimes. It may be because the weakly supervised penalty uses some negative information to regularize the preimage value, but from another point of view, this might drive the preimage away from some ideal positive neighbors (training samples here) at the same time. So, more neighbors from training samples may be needed to alleviate this possible side effect. Currently, it would be difficult to fully interpret why the case of using polynomial kernel is less sensitive to weakly supervised penalty. One interpretation might be that the polynomial kernel with factional power used in this paper may be already discriminative itself as explored by Liu [22].

Moreover, taking the performance of $\mathrm{P^2L}$ with weakly supervised penalty ($\eta = 10^{-4}$) when $s = 5$ as the reference, Table XVIII shows that the computational expense of quadratic program will increase a lot for each testing image if s is large

TABLE XIII
MSE: Criterion (16) Versus Nonconvex Criterion (16), Using Weakly Supervised Penalty ($\eta^{-} = 10^{-4}$)

Applications	Method	RBF				Polyn.			
		s=3	s=5	s=10	s=15	s=3	s=5	s=10	s=15
Expression Normalization	Non-convex Criterion (16)	13.833	15.281	21.177	27.359	13.396	14.736	17.149	19.442
	Non-convex Criterion (16), CUT	13.699	14.984	20.05	25.182	13.378	14.714	17.113	19.391
	Criterion (16)	**12.011**	**11.592**	**11.417**	**11.456**	**12.692**	**13.152**	**14.013**	**14.861**
Face Image Denoising	Non-convex Criterion (16)	40.581	39.075	41.228	45.813	**39.859**	39.122	40.433	41.695
	Non-convex Criterion (16), CUT	**40.52**	38.934	40.874	45.225	**39.859**	**39.121**	40.43	41.691
	Criterion (16)	40.925	**38.323**	**36.536**	**35.886**	40.942	39.733	**39.696**	**39.887**
Recovery of Occluded Face Image	Non-convex Criterion (16)	35.746	35.191	40.774	47.814	30.764	29.29	29.798	31.227
	Non-convex Criterion (16), CUT	35.65	34.896	39.767	45.962	30.748	29.257	29.724	31.107
	Criterion (16)	**34.401**	**31.573**	**28.506**	**27.378**	**28.111**	**27.015**	**26.909**	**27.157**
Illumination Normalization	Non-convex Criterion (16)	88.178	108.934	154.71	186.468	69.271	74.915	87.981	96.587
	Non-convex Criterion (16), CUT	85.854	103.17	140.55	164.82	69.067	74.457	86.82	94.914
	Criterion (16)	**64.025**	**60.605**	**58.131**	**57.346**	**62.106**	**61.285**	**61.356**	**61.739**

TABLE XIV
MSE Versus Number of Neighbors in $\mathrm{P}^2\mathrm{L}$, Criterion (15)

Applications	kernel	MSE												
		s=3	s=5	s=10	s=15	s=20	s=30	s=40	s=50	s=60	s=70	s=80	s=90	s=100
Expression Normalization	RBF	**12.42**	13.206	14.22	14.917	15.966	16.88	17.387	17.709	17.942	18.135	18.282	18.421	18.541
	Polyn.	**12.696**	13.177	14.062	14.927	16.007	16.894	17.38	17.71	17.978	18.161	18.326	18.451	18.559
Face Image Denoising	RBF	40.3	**38.979**	39.149	39.479	39.799	40.145	40.247	40.412	40.467	40.527	40.621	40.695	40.783
	Polyn.	40.96	39.769	**39.758**	39.967	40.2	40.4	40.529	40.604	40.729	40.774	40.745	40.761	40.799
Recovery of Occlusion	RBF	34.63	33.258	**32.727**	32.851	32.926	33.119	33.243	33.344	33.378	33.447	33.538	33.564	33.678
	Polyn.	28.138	27.062	**26.983**	27.245	27.472	27.74	27.931	28.047	28.167	28.258	28.275	28.35	28.386
Illumination Normalization	RBF	65.283	**64.088**	64.358	64.769	64.928	65.392	65.824	66.211	66.477	66.767	66.939	67.13	67.264
	Polyn.	62.151	**61.356**	61.469	61.872	62.145	62.684	63.071	63.376	63.559	63.723	63.908	64.154	64.325

TABLE XV
MSE Versus Number of Neighbors, Criterion (16), Ridge Penalty ($\lambda = 10^{-4}$)

Applications	kernel	MSE												
		s=3	s=5	s=10	s=15	s=20	s=30	s=40	s=50	s=60	s=70	s=80	s=90	s=100
Illumination Normalization	RBF	65.212	**64.019**	64.311	64.731	64.895	65.363	65.798	66.188	66.455	66.745	66.918	67.109	67.243
	Polyn.	62.15	**61.355**	61.469	61.871	62.144	62.684	63.07	63.376	63.559	63.723	63.908	64.154	64.325

TABLE XVI
MSE Versus Number of Neighbors, Criterion (16), Laplacian Penalty ($\lambda = 10^{-4}$)

Applications	kernel	MSE												
		s=3	s=5	s=10	s=15	s=20	s=30	s=40	s=50	s=60	s=70	s=80	s=90	s=100
Illumination Normalization	RBF	64.289	**63.094**	64.134	64.987	65.407	66.066	66.582	67.027	67.335	67.63	67.822	68.026	68.182
	Polyn.	62.129	**61.336**	61.467	61.876	62.152	62.694	63.083	63.39	63.574	63.739	63.924	64.17	64.342

(e.g., $s \geq 100$) but with little significant improvement. Like some preimage algorithms, P^2L is learned for each testing sample. So, it would be reasonable for $\mathrm{P}^2\mathrm{L}$ to use an appropriate small number of neighbors to infer preimage in this aspect. As shown later, even though s is set appropriately small (e.g., $s = 15$), $\mathrm{P}^2\mathrm{L}$ still obtains (much) lower MSE and better visual results as compared to existing methods.

In summary, the results show that in the two proposed criteria (15) and (16), learning the preimage of a feature vector using its nearby neighbors rather than a large amount of training samples would be already satisfactory and get better performances than existing methods as shown later. For the completely unsupervised case, quite a limited number of neighbors are already enough. When weakly supervised information is used, sometimes, a bit more neighboring information would be preferred for some kernel, such as RBF, and this may also depend on how challenging the application is. In such a case, a tradeoff between reconstruction performance and computational complexity needs to be considered. In practice, some model selection techniques could be used. Section VI will provide additional discussion on this issue.

C. *Comparison With Other Methods*

This section conducts a comparison between the proposed penalized preimage learning model and three representative methods, namely, Mika's method [5], the distance-constraint-based scheme proposed by Kwok and Tsang [6], and preimage map [7]. As Mika's method was originally developed based on RBF kernel [5], so for polynomial kernel, we used the gradient update scheme for acquiring a solution. In our experiments, we set $\rho = 0.1$ in (6), set the training sample mean as the initialized value, and set the maximum iteration count to

TABLE XVII
MSE VERSUS NUMBER OF NEIGHBORS, CRITERION (16), WEAKLY SUPERVISED PENALTY ($\eta^- = 10^{-4}$)

Applications	kernel	MSE												
		s=3	*s*=5	*s*=10	*s*=15	*s*=20	*s*=30	*s*=40	*s*=50	*s*=60	*s*=70	*s*=80	*s*=90	*s*=100
Expression Normalization	RBF	12.011	11.592	**11.417**	11.456	11.605	11.873	12.158	12.412	12.66	12.868	13.052	13.259	13.47
	Polyn.	**12.692**	13.152	14.013	14.861	15.916	16.78	17.253	17.574	17.834	18.013	18.173	18.295	18.398
Face Image Denoising	RBF	40.925	38.323	36.536	35.886	**35.774**	36.047	36.379	36.426	36.261	36.202	36.214	36.033	36.011
	Polyn.	40.942	39.733	**39.696**	39.887	40.111	40.298	40.419	40.49	40.611	40.651	40.618	40.629	40.663
Recovery of Occlusion	RBF	34.401	31.573	28.506	27.378	26.859	26.436	26.265	**26.248**	26.274	26.297	26.416	26.476	26.717
	Polyn.	28.111	27.015	**26.909**	27.157	27.375	27.632	27.815	27.927	28.044	28.132	28.146	28.22	28.254
Illumination Normalization	RBF	64.025	60.605	58.131	57.346	56.625	55.976	**55.701**	55.747	55.936	56.355	56.586	57.2	57.957
	Polyn.	62.106	**61.285**	61.356	61.739	61.999	62.518	62.891	63.184	63.358	63.514	63.693	63.931	64.098

TABLE XVIII
PERFORMANCE IMPROVEMENT VERSUS INCREASE OF COMPUTATIONAL EXPENSE [TAKE $\mathrm{P^2L}$ AT $s = 5$ AS THE REFERENCE PERFORMANCE, WEAKLY SUPERVISED PENALTY, RBF ($\eta^- = 10^{-4}$)]

Applications	Items	Neighbors					
		s=5	*s*=15	*s*=40	*s*=60	*s*=80	*s*=100
Face Image Denoising	Performance Improvement	N/A	6.36%	5.07%	5.38%	5.50%	6.03%
	Increase of Computational Expense	N/A	16.63%	98.33%	204.85%	358.92%	552.73%
Recovery of Occlusion	Performance Improvement	N/A	13.29%	16.81%	16.78%	16.33%	15.38%
	Increase of Computational Expense	N/A	15.15%	94.65%	202.01%	353.12%	548.47%
Illumination Normalization	Performance Improvement	N/A	5.38%	8.09%	7.70%	6.63%	4.37%
	Increase of Computational Expense	N/A	17.37%	84.44%	166.56%	309.50%	502.43%

TABLE XIX
MSES OF NORMALIZED FACIAL EXPRESSION IMAGES USING COMPARED METHODS

kernel	Distance Constraint				Mika's Method	Pre-image Map	PCA	$\mathrm{P^2L}$ (Weakly Supervised Penalty), $\eta^-=10^{-4}$			
	s=3	*s*=5	*s*=10	*s*=15				*s*=3	*s*=5	*s*=10	*s*=15
RBF	8.68e+005	14.297	16.626	18.506	45.854	42.781	48.76	12.011	11.592	**11.417**	11.456
Polyn.	4.73e+005	14.367	16.784	19.109	79.427	50.28		**12.692**	13.152	14.013	14.861

TABLE XX
MSES OF DENOISED YALEB IMAGES USING COMPARED METHODS

kernel	Distance Constraint				Mika's Method	Pre-image Map	PCA	$\mathrm{P^2L}$ (Weakly Supervised Penalty), $\eta^-=10^{-4}$			
	s=3	*s*=5	*s*=10	*s*=15				*s*=3	*s*=5	*s*=10	*s*=15
RBF	3857.4	480.43	39.415	40.297	54.235	59.3	53.381	40.925	38.323	36.536	**35.886**
Polyn.	77.637	42.386	40.119	40.825	61.368	50.998		40.942	39.733	**39.696**	39.887

TABLE XXI
MSES OF RECONSTRUCTED OCCLUDED IMAGES USING COMPARED METHODS

kernel	Distance Constraint				Mika's Method	Pre-image Map	PCA	$\mathrm{P^2L}$ (Weakly Supervised Penalty), $\eta^-=10^{-4}$			
	s=3	*s*=5	*s*=10	*s*=15				*s*=3	*s*=5	*s*=10	*s*=15
RBF	343.99	54.909	36.349	38.788	72.451	120.91	78.924	34.401	31.573	28.506	**27.378**
Polyn.	702.56	83.593	34.51	38.12	54.86	163.23		28.111	27.015	**26.909**	27.157

TABLE XXII
MSES OF NORMALIZED ILLUMINATED IMAGES USING COMPARED METHODS

kernel	Distance Constraint				Mika's Method	Pre-image Map	PCA	$\mathrm{P^2L}$ (Weakly Supervised Penalty), $\eta^-=10^{-4}$			
	s=3	*s*=5	*s*=10	*s*=15				*s*=3	*s*=5	*s*=10	*s*=15
RBF	233.32	132.86	90.7	93.943	121.2	126.109	128.67	64.025	60.605	58.131	**57.346**
Polyn.	799.45	320.42	105.64	99.64	62.832	142.028		62.106	**61.285**	61.356	61.739

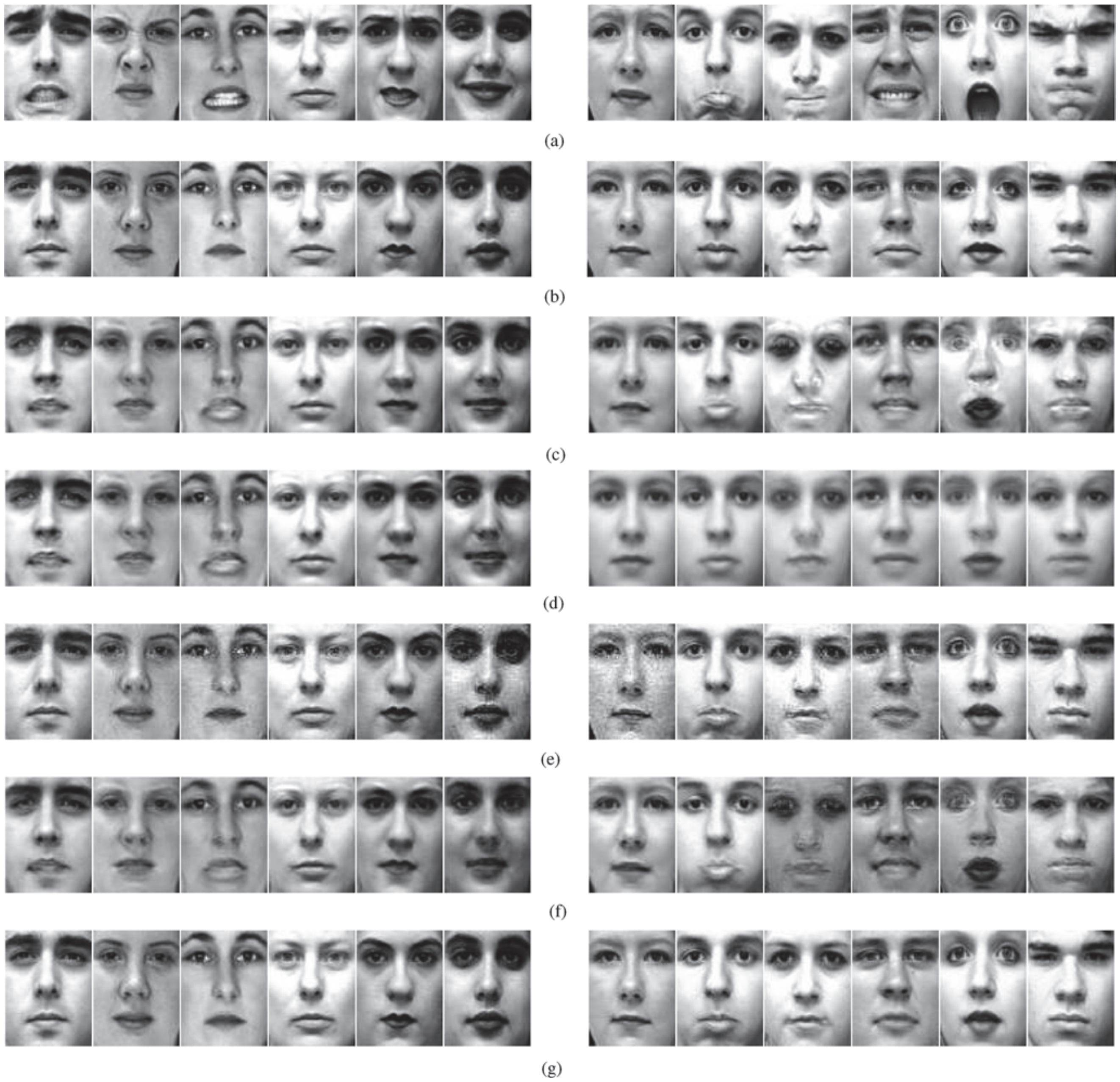


Fig. 2. Illustration of normalized facial expression images. In each row of (d)–(g), the left six images are results based on RBF kernel and the right six images are results based on polynomial kernel. Five neighbors are used in both the distance-constraint-based method and $\mathrm{P^2L}$. The importance weight η in $\mathrm{P^2L}$ is 10^{-4}. Noting that there is no unique ground-truth natural facial expression image for each person in the data set, we show the one nearest to the mean of training samples for each person in (b) for reference. (a) Facial expression images. (b) References of natural facial expression images. (c) PCA. (d) Mika's method. (e) Distance constraint. (f) Preimage map. (g) Penalized preimage learning (weakly supervised).

be 200. In preimage map, we know that a different kernel κ' is additionally used for kernel regression [7]. However, as far as we know, how to determine this kind of kernel function is still an unsolved or at least a not well-investigated problem. In this paper, we selected the RBF kernel and set

$$\kappa'(P_\kappa\varphi(\mathbf{x}), P_\kappa\varphi(\mathbf{y})) = \exp(-\|P_\kappa\varphi(\mathbf{x}) - P_\kappa\varphi(\mathbf{y})\|^2/c')$$

where c' was commonly set as

$$c' = \frac{1}{N(N-1)}\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N}\|P_\kappa\varphi(\mathbf{x}_i) - P_\kappa\varphi(\mathbf{x}_j)\|^2.$$

In our experiments, the parameter υ in (8) was selected from $\{100, 10, 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$, and the best result would be reported. In addition, results of PCA will also be reported for reference.

For convenience of the comparison, in Tables XIX–XXII, the results of the proposed model $\mathrm{P^2L}$ with weakly supervised penalty ($\eta^- = 10^{-4}$) are included. The results show that the proposed model overall achieves significant improvements against the compared methods. This scenario could be more evidently observed in the experiments of facial expression normalization, recovery of occluded images, and illumination normalization. Note that, in most cases, $\mathrm{P^2L}$ using other penalty functions that are not listed in Tables XIX–XXII or

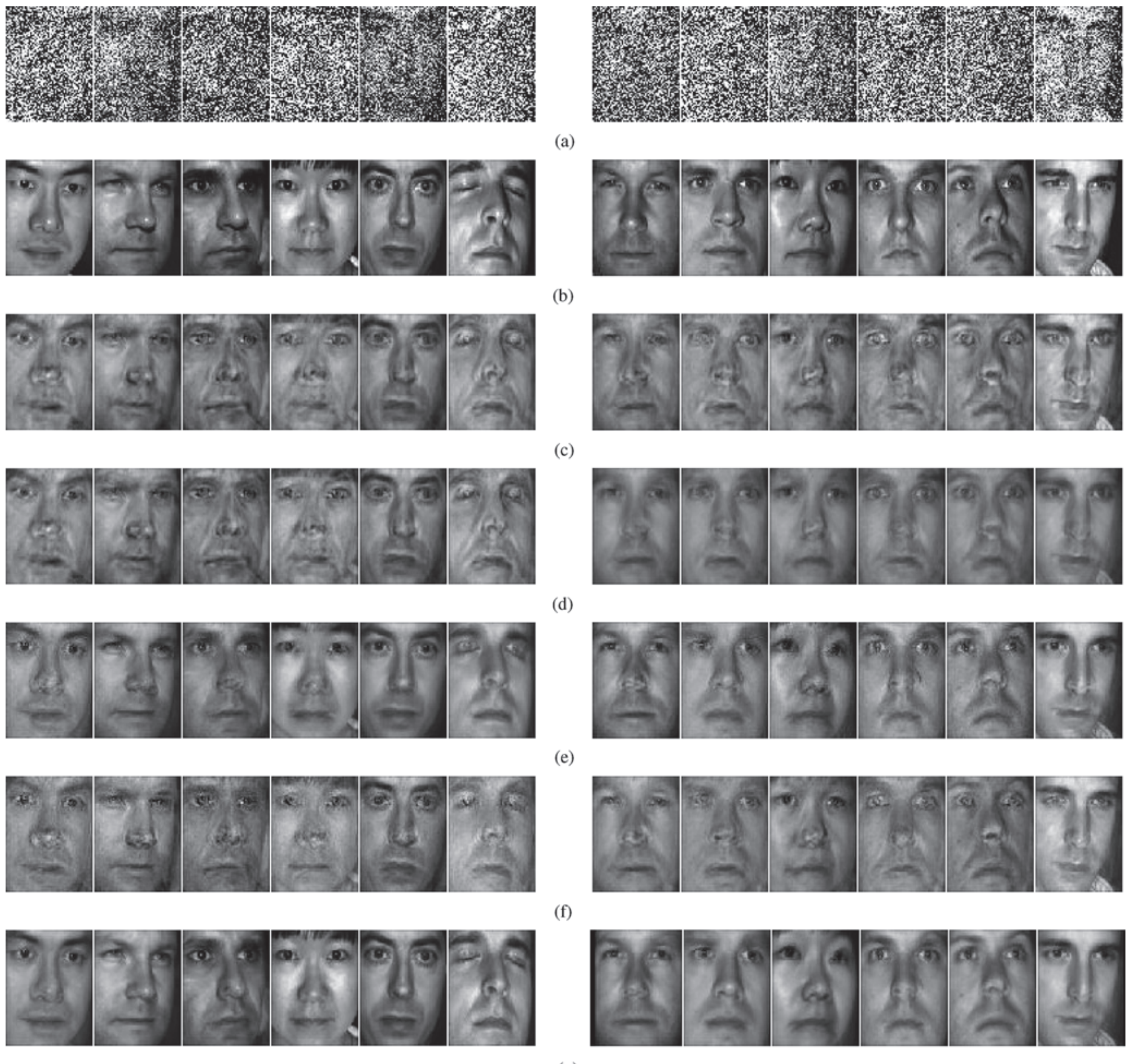


Fig. 3. Illustration of denoised face images. In each row of (d)–(g), the left six images are results based on RBF kernel and the right six images are based on polynomial kernel. Ten neighbors are used in the distance-constraint-based method and $\mathrm{P^2L}$. The importance weight η^- in $\mathrm{P^2L}$ is 10^{-4}. (a) Noisy images. (b) Original images. (c) PCA. (d) Mika's method. (e) Distance constraint. (f) Preimage map. (g) Penalized preimage learning (weakly supervised).

just in form of the proposed criterion (15) can also get superior results to the ones obtained by existing methods. Though Mika's method also performs well for illumination normalization when using polynomial kernel, it is computationally expensive. The preimage map does not always get much lower MSE than Mika's method except the cases for facial expression normalization and image denoising using polynomial kernel. Though it can obtain better visual quality than Mika's method for digit image denoising in our inner test which coincides with the finding in [7], it is somewhat surprising to us that the performance of preimage map is still overall unsatisfactory over the applications presented here. This may be at least due to its two weaknesses. First, as discussed in Section II-C, preimage map currently cannot be learned by taking use of negative samples; second, the selections of kernel function for kernel regression and the regularization parameter υ are so far still difficult to be determined in literatures.

In addition, we have an experimental finding that the distance constraint scheme would not perform well for high-dimensional data when fewer neighbors are used, and as seen in Tables XIX–XXII, the MSEs of distance constraint are high when $s = 3$. Interpretation of this scenario has been given in the previous section. Though the number of neighbors of the kernel feature should be properly designated in both $\mathrm{P^2L}$ and the distance-constraint-based method, the performance of $\mathrm{P^2L}$ would be affected insignificantly when different numbers of neighbors are used. This is particularly indicated by the results shown in Tables XIV–XVII.

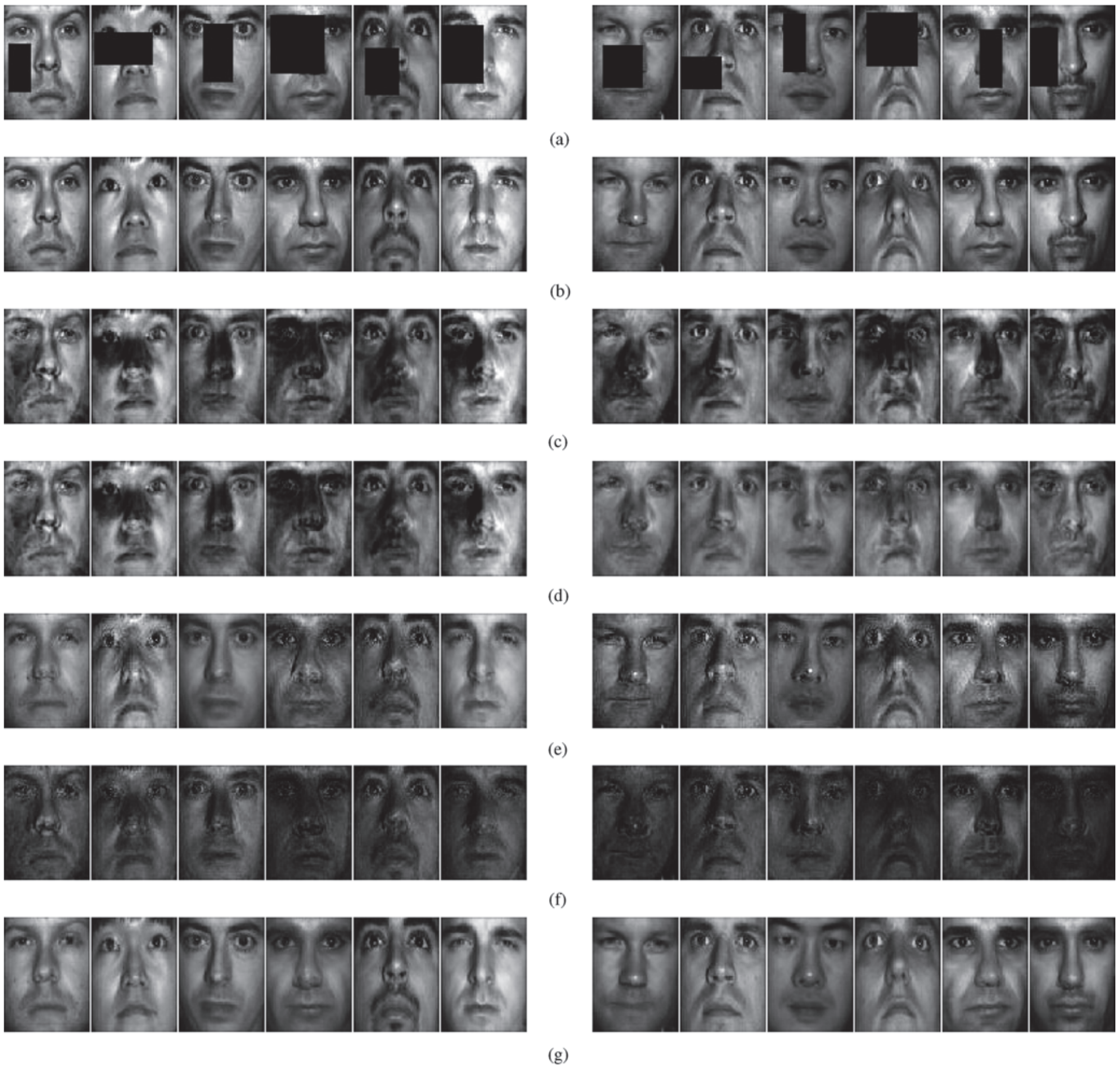


Fig. 4. Illustration of reconstructed occluded face images. In each row of (d)–(g), the left six images are results based on RBF kernel and the right six images are results based on polynomial kernel. Ten neighbors are used in the distance-constraint-based method and $\mathrm{P}^2\mathrm{L}$. The importance weight η^- in $\mathrm{P}^2\mathrm{L}$ is 10^{-4}. (a) Occluded images. (b) Original images. (c) PCA. (d) Mika's method. (e) Distance constraint. (f) Preimage map. (g) Penalized preimage learning (weakly supervised).

Finally, we compare the visual results of all methods which are shown in Figs. 2–5. For illustration, the numbers of neighbors used in the distance constraint method and $\mathrm{P}^2\mathrm{L}$ are indicated in the caption of each figure. They were selected as the tradeoff between better visual results and smaller MSE values obtained in our experiments. For preimage map, we mainly illustrate the results corresponding to the lowest MSE. For visualization, we have implemented the Matlab function "mat2gray," in which the intensity of image was set within the range from 0 to 1. It should be noted that the MSE values reported in Tables I–XXII were computed based on pure data, which were purely obtained by preimage learning algorithms without any additional preprocessing.

The comparison results show that the distance constraint scheme and $\mathrm{P}^2\mathrm{L}$ always perform better than the iterative method and the preimage map. However, $\mathrm{P}^2\mathrm{L}$ gets better visual quality of reconstructed face images, especially by comparing the local portions of images near noses, eyes, mouths, etc. Although Mika's method also gets low MSE in illumination normalization when polynomial kernel is used as reported in Table XXII, the reconstructed images shown in Fig. 5(d) are smoother than the ones obtained by the proposed method. In addition, the results also show that KPCA with preimage learning performs better than PCA. In summary, the experimental results show that $\mathrm{P}^2\mathrm{L}$ algorithm gives lower MSE and better visual results.

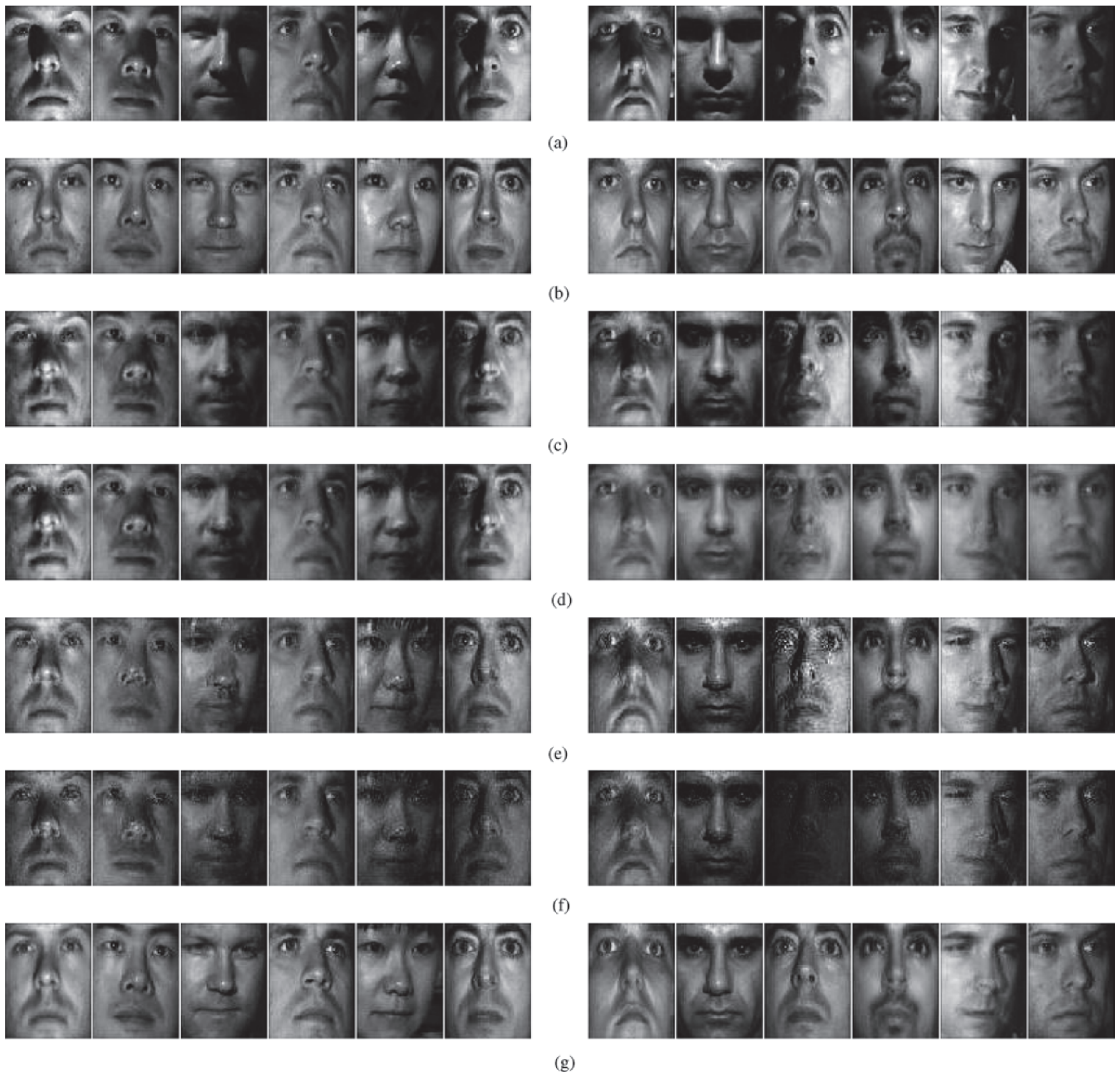


Fig. 5. Illustration of normalized illuminated face images. In each row of (d)–(g), the left six images are results based on RBF kernel and the right six images are results based on polynomial kernel. Ten neighbors are used in the distance-constraint-based method and five neighbors are used in $\mathrm{P^2L}$. The importance weight η^- in $\mathrm{P^2L}$ is 10^{-4}. (a) Illuminated images. (b) Original images. (c) PCA. (d) Mika's method. (e) Distance constraint. (f) Preimage map. (g) Penalized preimage learning (weakly supervised).

VI. Conclusion and Discussion

In this paper, we address the preimage problem in kernel PCA. The contributions are as follows.

1) An efficient penalized preimage learning $(\mathrm{P^2L})$ methodology has been developed based on a two-step general framework.
2) For penalization, the weakly supervised penalty is proposed and extensively evaluated along with Laplacian penalty and ridge penalty. The convexity constraint on the combination weights is also introduced, evaluated, and discussed in the proposed methodology.

Moreover, a comprehensive comparison between different preimage learning methods in KPCA has also been reported by conducting extensive experiments in four different applications, namely facial expression normalization, face image denoising, recovery of occluded face image, and illumination normalization of face image.

Compared with existing preimage learning models, the major differences are that the preimage learned by $\mathrm{P^2L}$ is directly modeled by convex combination of training samples and further learned by the guidance of penalized function, so that the preimage is well defined and preserves the penalized pointwise conditional mutual information. In existing algorithms, the preimage values are indirectly learned as the combination of training samples, since their preimage algorithms are not directly turned to the optimization of the combination weights as demonstrated in this paper. Existing methods do not guar-

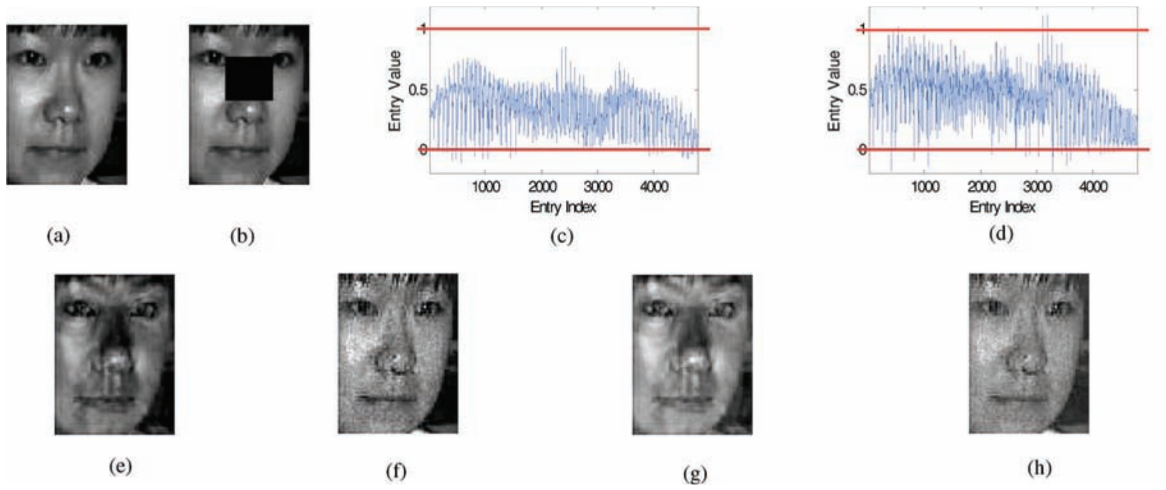


Fig. 6. Examples of preimage values extracted by Mika's method and Kwok and Tang's method, respectively. In our experiments, each pixel in a gray image is ranging from 0 to 1. (a) Original image. (b) Occluded image. (c) Preimage vector obtained by method [5]. (d) Preimage vector obtained by method [6]. (e) Reconstructed image obtained by method [5], using normalization-I. (f) Reconstructed image obtained by method [6], using normalization-I. (g) Reconstructed image obtained by method [5], using normalization-II. (h) Reconstructed image obtained by method [6], using normalization-II. Two types of normalization, namely, normalization-I and normalization-II are used for showing the visual results. For normalization-I, threshold is made as normalization such that the entries of the preimage vector are set to be 1 if they exceed 1 and 0 if they are negative; for normalization-II, linear scaling is directly done such that the entry values are stretched, ranging from 0 to 1. All results are based on RBF kernel with the parameter specified in the experiment.

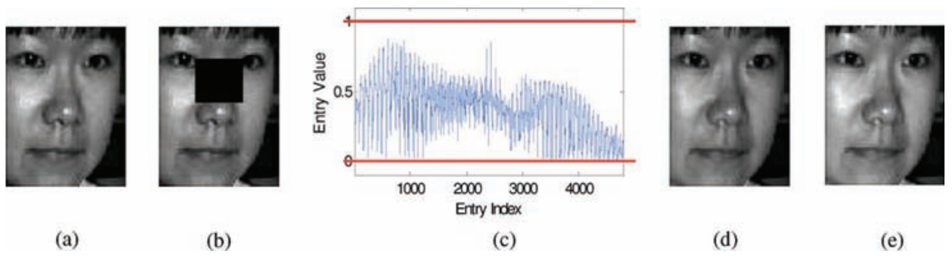


Fig. 7. Examples of the preimage values obtained by $\mathrm{P}^2\mathrm{L}$. (a) Original image. (b) Occluded image. (c) Preimage vector obtained by the proposed $\mathrm{P}^2\mathrm{L}$ (weakly supervised penalty, $\eta^- = 10^{-4}$). (d) Reconstructed image obtained by $\mathrm{P}^2\mathrm{L}$, using normalization-I. (e) Reconstructed image obtained by $\mathrm{P}^2\mathrm{L}$, using normalization-II. All results are based on RBF kernel with the parameter specified in the experiment.

antee to learn a well-defined preimage and the penalization framework is not modeled and widely discussed for alleviating the ill-posed problem in preimage learning as well. Besides these main differences, $\mathrm{P}^2\mathrm{L}$ is noniterative as compared to Mika's method [5], learns the preimage that preserves local nonnegative penalized reconstruction information as compared to the distance-constraint-based algorithm that preserves the local information in line with MDS [6], and is able to use (local) negative information as compared to preimage map [7]. In addition, it might be possible to apply convexity constraint or the proposed penalty function by modifying some existing models; however, the advantage of realizing these ideas in criteria (15) and (16) is that the combination weights of samples are directly modeled for quadratic optimization and this would benefit for developing efficient algorithms for preimage learning. Experimental results show $\mathrm{P}^2\mathrm{L}$ can get lower MSE values and better visual quality of reconstructed images.

Among the penalty functions discussed, the weakly supervised penalty outperforms the others. Currently, one concern about this penalty is how to get positive and negative information. In current experiments, negative data were obtained by additional simulation for applications of image denoising and recovery of occlusion. Even for facial expression and illumination normalization, simulation of facial expression and illuminated images could also be possible by using active appearance models [23] and Lambertian model [24], respectively. If this information, especially the negative information, is sometimes not immediately available or difficult to be simulated in practice, we could consider an incremental manner for implementation. It can be motivated that some positive or negative information may be able to be explored from some testing samples that have already been observed. Assume there is a filter that can distinguish any of these testing images as positive or negative information well. In this way, the knowledge of positive/negative information can grow out of nothing as more testing images are observed. It should be noted that such filter and the details of incremental technique need to be well designed in future. Specially, if it is assumed that the images to be preprocessed are corrupted, then they can be treated as negative samples and the incremental learning may be implemented more easily without a filter.

A possible future research issue for weakly supervised penalty (as well as other related methods) could be on the selection of useful neighbors. So far, we demonstrated experimentally that local information could be already satisfactory for preimage learning under criterion (16). It would currently be still difficult to design an elegant and efficient strategy to determine the optimal number of neighbors by balancing the reconstruction performance and computational expense, though a practical and simple way could be to employ a validation set for some model selection, conditioned that the maximum number (e.g., 100) of nearest neighbors is restricted.[8] Moreover, it may also be useful to discuss some more general topic about the selection of useful neighbors in future.

Finally, the extensive evaluations of different preimage learning algorithms in a wide range of applications also show that KPCA with preimage learning could be widely applied to many applications and appealing results can be obtained, though KPCA is implemented without incorporating any special physical model. However, KPCA is still an unsupervised technique and one can find that even using the proposed preimage learning model, a few processed images are likely different from the reference images. In this aspect, applying preimage learning to some other kernel algorithms such as support vector data description as done in [25], [26] or some supervised kernel algorithms may possibly be useful for tackling this problem in future. Also, how to combine "KPCA + preimage learning" with existing models for solving specific computer vision problem could be an interesting research issue, as specific model is designed to solve each application and "KPCA + preimage learning" may perform some compensation. For example, in a recently reported work for face illumination normalization [27], there were still some artifacts appearing in the normalized images such as occlusions and noises, which appeared unpredictably in the normalized image. In this case, "KPCA + preimage learning" could be used for denoising and recovery process. However, how to use it properly still needs further investigations in future.

Appendix I

A sketch of KPCA is given here. Similar contents can be referred to [2], [5], and [6]. The goal of this presentation is mainly to acquire a simple formula for projecting a feature vector onto a kernel principal subspace as shown by (26).

Define $\mathbf{O}_t^\varphi = (1/\sqrt{N})(\varphi(\mathbf{x}_1)-\boldsymbol{\mu}^\varphi, \ldots, \varphi(\mathbf{x}_N)-\boldsymbol{\mu}^\varphi)$. Then, KPCA in fact performs linear PCA in the feature space and is formulated as the following eigenvalue problem:

$$\mathbf{S}_t^\varphi \mathbf{U}^\varphi = \mathbf{U}^\varphi \boldsymbol{\Lambda}^\varphi \tag{25}$$

where $\mathbf{S}_t^\varphi = \mathbf{O}_t^\varphi \mathbf{O}_t^{\varphi T}$, $\mathbf{U}^\varphi = (\mathbf{u}_1^\varphi, \ldots, \mathbf{u}_q^\varphi)$, $\boldsymbol{\Lambda}^\varphi = \mathrm{diag}(\lambda_1^\varphi, \ldots, \lambda_q^\varphi)$, $\lambda_1^\varphi \geq \cdots \geq \lambda_q^\varphi > 0$. As

$$\mathbf{O}_t^\varphi = N^{-0.5}(\boldsymbol{\Psi}_\varphi - \boldsymbol{\mu}^\varphi \mathbf{e}_N^T) = N^{-0.5}\boldsymbol{\Psi}_\varphi(\mathbf{I}_N - N^{-1}\mathbf{e}_N\mathbf{e}_N^T)$$

and

$$(\mathbf{I}_N - N^{-1}\mathbf{e}_N\mathbf{e}_N^T)(\mathbf{I}_N - N^{-1}\mathbf{e}_N\mathbf{e}_N^T)^T = (\mathbf{I}_N - N^{-1}\mathbf{e}_N\mathbf{e}_N^T)$$

we then have

$$\mathbf{S}_t^\varphi = \mathbf{O}_t^\varphi \mathbf{O}_t^{\varphi T} = N^{-1}\boldsymbol{\Psi}_\varphi(\mathbf{I}_N - N^{-1}\mathbf{e}_N\mathbf{e}_N^T)\boldsymbol{\Psi}_\varphi^T.$$

Also, from (25), for each $\mathbf{u}_i^\varphi$, we could have some $\boldsymbol{\alpha}_i = (\alpha_1^i, \ldots, \alpha_N^i)^T$ [2], [3] such that

$$\mathbf{u}_i^\varphi = \sum_{j=1}^N \alpha_j^i \frac{1}{\sqrt{N}}(\varphi(\mathbf{x}_j) - \boldsymbol{\mu}^\varphi) = \mathbf{O}_t^\varphi \boldsymbol{\alpha}_i = \boldsymbol{\Psi}_\varphi \mathbf{p}_i$$

where $\mathbf{p}_i = N^{-0.5}(\mathbf{I}_N - N^{-1}\mathbf{e}_N\mathbf{e}_N^T)\boldsymbol{\alpha}_i$. So $\mathbf{U}^\varphi = \boldsymbol{\Psi}_\varphi \mathbf{P}$, $\mathbf{P} = (\mathbf{p}_1, \ldots, \mathbf{p}_q)$. Denote the subspace spanned by the first q_0 largest kernel principal components by $\mathbf{U}_{q_0}^\varphi = \boldsymbol{\Psi}_\varphi \mathbf{P}_{q_0}$, $\mathbf{P}_{q_0} = (\mathbf{p}_1, \ldots, \mathbf{p}_{q_0})$. Then, for a given pattern $\mathbf{x}$, in KPCA, the projection $P_\kappa \varphi(\mathbf{x})$ of $\varphi(\mathbf{x})$ onto such subspace is

$$\begin{aligned} P_\kappa \varphi(\mathbf{x}) &= \mathbf{U}_{q_0}^\varphi \mathbf{U}_{q_0}^{\varphi T}(\varphi(\mathbf{x}) - \boldsymbol{\mu}^\varphi) + \boldsymbol{\mu}^\varphi \\ &= \boldsymbol{\Psi}_\varphi \mathbf{P}_{q_0}\mathbf{P}_{q_0}^T \boldsymbol{\Psi}_\varphi^T(\varphi(\mathbf{x}) - N^{-1}\boldsymbol{\Psi}_\varphi \mathbf{e}_N) + N^{-1}\boldsymbol{\Psi}_\varphi \mathbf{e}_N \\ &= \boldsymbol{\Psi}_\varphi \boldsymbol{\gamma}^{\mathbf{x}} \end{aligned} \tag{26}$$

where $\boldsymbol{\gamma}^{\mathbf{x}} = (\gamma_1^{\mathbf{x}}, \ldots, \gamma_N^{\mathbf{x}})^T = \mathbf{P}_{q_0}\mathbf{P}_{q_0}^T\boldsymbol{\Psi}_\varphi^T\varphi(\mathbf{x}) - N^{-1}\mathbf{P}_{q_0}\mathbf{P}_{q_0}^T\mathbf{K}\mathbf{e}_N + N^{-1}\mathbf{e}_N$ and $\mathbf{K} = \boldsymbol{\Psi}_\varphi^T\boldsymbol{\Psi}_\varphi$.

Appendix II

This appendix shows an example that the proposed model performs more robust, as compared to the state-of-the-art methods, even though popular normalization technique is not applied to the learned preimage. For this purpose, we here compare the results of Mika's [5] and Kwok and Tsang's methods[9] [6]. Two examples are first shown in Fig. 6. As shown in Fig. 6(c) and (d), the learned preimages do not lie in the image domain $[0,1]^n$, since some pixel value can be negative or larger than 1. In Fig. 6(e)–(h), the visual results produced by two commonly used normalization techniques are also shown. It is shown that the results may not still be good even if a normalization step is additionally adopted after preimage learning. For comparison, we show the corresponding results of the proposed model in Fig. 7. Note that the preimage learned by $\mathrm{P^2L}$ can be well defined without any normalization. So the result shows that even though the normalization is not processed, the reconstructed image learned by $\mathrm{P^2L}$ can be much more similar to the original reference image. This indicates that with convexity constraint, the proposed method could perform more robust. More extensive justifications could be found in Section V-B1.

Acknowledgment

The authors would like to thank all reviewers and the associate editor for their constructive comments that improved this paper.

References

[1] M. Kirby and L. Sirovich, "Application of the Karhunen-Loeve procedure for the characterization of human faces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 1, pp. 103–108, Jan. 1990.

[8]This is only a possible way that may not be optimal to control the computational expense of the model.

[9]Preimage map would also have similar scenarios with Mika's method and Kwok and Tsang's method here. It is not illustrated due to the limited room.

[2] B. Schölkopf, A. Smola, and K. R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, pp. 1299–1319, 1998.

[3] B. Schölkopf and A. Smola, *Learning With Kernels*. Cambridge, MA: MIT Press, 2002.

[4] V. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

[5] 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*. Cambridge, MA: MIT Press, 1998.

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

[7] G. H. Bakır, J. Weston, and B. Schölkopf, "Learning to find pre-images," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2004.

[8] W.-S. Zheng and J. H. Lai, "Regularized locality preserving learning of pre-image problem in kernel principal component analysis," in *Proc. 18th Int. Conf. Pattern Recognit.*, 2006, vol. 2, pp. 456–459.

[9] W.-S. Zheng, J. H. Lai, and P. C. Yuen, "Weakly supervised learning on pre-image problem in kernel methods," in *Proc. 18th Int. Conf. Pattern Recognit.*, 2006, vol. 2, pp. 711–715.

[10] P. Arias, G. Randall, and G. Sapiro, "Connecting the out-of-sample and pre-image problems in kernel methods," in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007, DOI: 10.1109/CVPR.2007.383038.

[11] K. I. Kim, M. O. Franz, and B. Schölkopf, "Iterative kernel principal component analysis for image modeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 9, pp. 1351–1366, Sep. 2005.

[12] R. Herbrich, *Learning Kernel Classifiers Theory and Algorithms*. Cambridge, MA: MIT Press, 2002.

[13] T. F. Cox and M. A. A. Cox, *Multidimensional Scaling, Monographs on Statistics and Applied Probability*, 2nd ed. London, U.K.: Chapman & Hall, 2001.

[14] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2326, 2000.

[15] J. B. Tenenbaum, V. D. Silvam, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 2000.

[16] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, pp. 1373–1396, 2003.

[17] S. Dambreville, Y. Rathi, and A. Tannenbaum, "Statistical shape analysis using kernel PCA," in *Proc. IS&T/SPIE Symp. Electron. Imag.*, 2006.

[18] S. Z. Li, "On discontinuity-adaptive smoothness priors in computer vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 6, pp. 576–586, Jun. 1995.

[19] T. Hastie, A. Buja, and R. Tibshirani, "Penalized discriminant analysis," *Ann. Stat.*, vol. 23, pp. 73–10, 1995.

[20] Y.-l. Tian, T. Kanade, and J. F. Cohn, "Recognizing action units for facial expression analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 2, pp. 97–115, Feb. 2001.

[21] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 643–660, Jun. 2001.

[22] C. Liu, "Gabor-based kernel PCA with fractional power polynomial models for face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 5, pp. 572–581, May 2004.

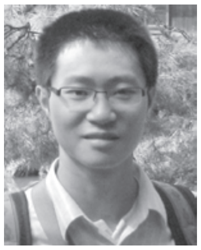
[23] T. Cootes, G. Edwards, and C. Taylor, "Active appearance models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 681–685, Jun. 2001.

[24] A. Shashua and T. Riklin-Raviv, "The quotient image: Class-based re-rendering and recognition with varying illuminations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 2, pp. 129–139, Feb. 2001.

[25] J. Park, D. Kang, J. Kim, J. T. Kwok, and I. W. Tsang, "Pattern de-noising based on support vector data description," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2005, vol. 2, pp. 949–953.

[26] J. Park, D. Kang, J. Kim, J. T. Kwok, and I. W. Tsang, "SVDD-based pattern denoising," *Neural Comput.*, vol. 19, no. 7, pp. 1919–1938, 2007.

[27] X. Xie, W.-S. Zheng, J. H. Lai, and P. C. Yuen, "Face illumination normalization on large and small scale features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2008, DOI: 10.1109/CVPR.2008.4587811.



Wei-Shi Zheng (M'08) received the B.S. degree in science with specialties in mathematics and computer science and the Ph.D. degree in applied mathematics from Sun Yat-Sen University, Guangzhou, China, in 2003 and 2008, respectively.

He is a Postdoctoral Researcher at the Department of Computer Science, Queen Mary University of London, London, U.K. He is now working on the European SAMURAI Research Project with Prof. S. Gong and Dr. T. Xiang. He has been a visiting student working with Prof. Z. Stan Li at the Institute of Automation, Chinese Academy of Sciences, Beijing, China, and an exchanged research student working with Prof. Yuen Pong C. at Hong Kong Baptist University, Hong Kong. His current research interests are in object association and categorization. He is also interested in discriminant/sparse feature extraction, dimension reduction, kernel methods in machine learning, and face image analysis.

Dr. Zheng was awarded the HP Chinese Excellent Student Scholarship 2008.

JianHuang Lai received the M.Sc. degree in applied mathematics and the Ph.D. degree in mathematics from Sun Yat-sen University, Guangzhou, China, in 1989 and 1999, respectively.

He joined Sun Yat-sen University as an Assistant Professor in 1989. Currently, he is a Professor of the Department of Automation, School of Information Science and Technology, and also the Vice-Dean of the school. He has published over 80 scientific papers in the national and international journals and conferences on image processing and pattern recognition. His current research interests are in the areas of digital image processing, pattern recognition, multimedia communication, and wavelet and its applications.

Dr. Lai had successfully organized the 2004 International Conference on Advances in Biometric Personal Authentication, which was also the Fifth Chinese Conference on Biometric Recognition (Sinobiometrics'04), Guangzhou, China, in December 2004. He has taken charge of more than five national research projects, including NSF-Guangdong (U0835005), NSFC (60144001, 60373082, 60675016), the Key (Keygrant) Project of Chinese Ministry of Education (105134), and NSF of Guangdong, China (021766, 06023194). He serves as a standing director of the Image and Graphics Association of China and also serves as a standing director and secretary-general of the Image and Graphics Association of Guangdong.

Pong C. Yuen (S'92–M'93) received the B.Sc. degree in electronic engineering with first class honors from City Polytechnic of Hong Kong, Hong Kong, in 1989 and the Ph.D. degree in electrical and electronic engineering from The University of Hong Kong, Hong Kong, in 1993.

He joined the Department of Computer Science, Hong Kong Baptist University, Hong Kong, in 1993, as an Assistant Professor and currently is a Professor. He was a recipient of the University Fellowship to visit The University of Sydney, Sydney, Australia, in 1996. He was associated with the Laboratory of Imaging Science and Engineering, Department of Electrical Engineering. In 1998, he spent a six-month sabbatical leave at The University of Maryland Institute for Advanced Computer Studies (UMIACS), University of Maryland, College Park. He was associated with the Computer Vision Laboratory, CFAR. From June 2005 to January 2006, he was a Visiting Professor in GRAVIR laboratory (GRAphics, VIsion and Robotics) of INRIA, Rhone Alpes, France. He was associated with PRIMA Group. He was the Director of Croucher Advanced Study Institute (ASI) on biometric authentication in 2004 and the Director of Croucher ASI on Biometric Security and Privacy in 2007. His current research interests include human face processing and recognition, biometric security and privacy, and human activity recognition.

Dr. Yuen has been actively involved in many international conferences as an organizing committee and/or technical program committee member. Recently, he was the track Co-Chair of International Conference on Pattern Recognition 2006. He is an editorial board member of *Pattern Recognition*.