Iterative Kernel Principal Component Analysis for Image Modeling

Kwang In Kim, Matthias O. Franz, and Bernhard Schölkopf

Max Planck Institute für biologische Kybernetik
Spemannstr. 38, D-72076 Tübingen, Germany.
E-mail:kimki;mof;bs@tuebingen.mpg.de

Abstract - In recent years, *Kernel Principal Component Analysis* (KPCA) has been suggested for various image processing tasks requiring an image model such as, e.g., denoising or compression. The original form of KPCA, however, can be only applied to strongly restricted image classes due to the limited number of training examples that can be processed. We therefore propose a new iterative method for performing KPCA, the *Kernel Hebbian Algorithm* which iteratively estimates the Kernel Principal Components with only linear order memory complexity.

In our experiments, we compute models for complex image classes such as faces and natural images which require a large number of training examples. The resulting image models are tested in single-frame super-resolution and denoising applications. The KPCA model is not specifically tailored to these tasks; in fact, the same model can be used in super-resolution with variable input resolution, or denoising with unknown noise characteristics. In spite of this, both super-resolution and denoising performance are comparable to existing methods.

Index terms–Principal component analysis, Kernel methods, Image models, Image enhancement, Unsupervised learning.

1 Introduction

Prior knowledge about the statistics of specific image classes affords numerous applications in image processing such as super-resolution [14, 23], denoising [51, 44, 32], segmentation [30], or compression [5]. The prior can be coded either *implicitly* by directly learning the mapping between input and desired output (as in [14, 23]), or *explicitly* by finding a suitable image model. In image modeling, we can roughly distinguish between approaches that try to estimate aspects of the underlying probability distribution using a fixed set of basis elements such as wavelets [5, 44], projected profiles of objects [18] or geometrical primitives [40, 29], and approaches that try to find basis sets with certain optimality properties ranging from Principal Component Analysis (PCA) [41], Independent Component Analysis (ICA) [4, 24] to sparse coding [36].

Interestingly, all of the latter approaches model images as linear combinations of transparent basis images. Many researchers have pointed out, however, that this does not reflect the generation process of natural images (e.g., [40]). Here, one of the main contributing factors is *occlusion* which is highly nonlinear. This suggests the use of techniques that can cope with nonlinear combinations of basis images. One of these techniques is *Kernel Principal Component Analysis* (KPCA) [43]. In contrast to linear PCA, KPCA is capable of capturing part of the higher-order statistics which are particularly important for encoding image structure [11].

Capturing these higher-order statistics can require a large number of training examples, particularly for larger image sizes and complex image classes such as patches taken from natural images. This causes problems for KPCA, since KPCA requires to store and manipulate the *kernel matrix* the size of which is the square of the number of examples. To overcome this problem, a new iterative algorithm for KPCA, the *Kernel Hebbian Algorithm* (KHA) is introduced. It is based on the generalized Hebbian algorithm (GHA) which was introduced as an online algorithm for linear PCA [34, 41]. The resulting algorithm estimates the kernel principal components with linear order memory complexity, making it applicable to large problems.

In a previous application, KPCA was used for the denoising of handwritten digits [32], a relatively restricted class of images requiring a smaller number of training examples. With the KHA, the application domain can be extended to more complex image classes such as faces or natural images. Since KPCA is an *unsupervised* learning technique, the obtained image model can be used for other tasks as well. In our case, we apply the same image model in a single-frame super-resolution task where the details of a high-resolution image are restored from a single low-resolution image. This problem could previously be solved only in a supervised setting by encoding a fixed relationship between pairs of high- and low-resolution images [14, 23]. The results presented here indicate that a generic KPCA model can achieve a comparable performance to other, more specialized computer vision algorithms.

The remainder of this paper is organized as follows: Section 2 briefly introduces PCA, GHA, and KPCA. Section 3 formulates the KHA. Experimental results are presented in Section 4 and conclusions are drawn in Section 5.

2 Principal component models

2.1 Linear principal component analysis and Generalized Hebbian Algorithm

Given a set of l centered observations $\mathbf{x}_k = \mathbb{R}^N$, $k = 1, \dots, l$, and $\sum_{k=1}^{l} \mathbf{x}_k = 0$, PCA diagonalizes the covariance matrix[1]

$$\overline{C} = \frac{1}{l} \sum_{j=1}^{l} \mathbf{x}_j \mathbf{x}_j^{\top}. \tag{1}$$

For lower-dimensional data, this is readily performed by solving the eigenvalue equation

$$\lambda \mathbf{v} = \overline{C} \mathbf{v} \tag{2}$$

for eigenvalues $\lambda \geq 0$ and eigenvectors $\mathbf{v}_i \in \mathbb{R}^N \setminus \{0\}$ (cf., e.g. [21]). The resulting set of mutually orthogonal eigenvectors defines a new basis along the directions of maximum variance in the data. The pairwise decorrelated expansion coefficients in this new basis are called the *principal components* (PCs) of the dataset. From the point of view of image modeling, the PCA basis has the interesting property that, among all basis expansions, it minimizes the reconstruction error when the expansion is truncated to a smaller number of basis vectors. Thus, a class of high-dimensional images can be described by a low-dimensional model containing only a few PCs.

Computationally, it can be advantageous to solve the eigenvalue problem by iterative methods which do not need to compute and store $\overline{C}$ directly. This is particulary useful when the size of $\overline{C}$ is large such that the memory complexity becomes prohibitive. One widely used iterative PCA method is the so-called *Generalized Hebbian Algorithm* (GHA) by Sanger [41]. The GHA is a training algorithm for a linear, single-layer feedforward neural network of the form $\mathbf{y} = \mathbf{W}\mathbf{x}$ acting on the training examples $\mathbf{x}_k$. After training, each output of the network represents the projection on one eigenvector $\mathbf{v}_i$ of $\overline{C}$, ordered by decreasing eigenvalue. If we are interested in the eigenvectors corresponding to the r largest eigenvalues then $\mathbf{W}$ is an $r \times N$ weight matrix which is modified according to the update rule

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \eta(t)(\mathbf{y}(t)\mathbf{x}(t)^{\top} - \mathrm{LT}[\mathbf{y}(t)\mathbf{y}(t)^{\top}]\mathbf{W}(t)). \tag{3}$$

Here, the argument t denotes a discrete moment in time when an example $\mathbf{x}(t)$ is selected randomly from the $\mathbf{x}_k$ (with uniform probability over all $\mathbf{x}_k$) and presented to the network. $\eta(t)$ is a learning rate parameter, and LT$[\cdot]$ sets all elements above the diagonal of its matrix argument to zero, thereby making it lower triangular. In a local stability analysis, Oja [34] (for $r = 1$) and Sanger [41] (for $r \geq 1$) showed that the rows of $\mathbf{W}$ tend to the eigenvectors $\mathbf{v}_i$ of $\overline{C}$ as $t \to \infty$ for properly chosen initialization and learning rate (see Sect. 3.1). Intuitively, this can be seen by looking at (3): the first, Hebbian term $\mathbf{y}(t)\mathbf{x}(t)^{\top}$ in the brackets tries to maximize the output variance and thus will orient each row of $\mathbf{W}$ towards the largest eigenvector, whereas the diagonal elements in the second term LT$[\mathbf{y}(t)\mathbf{y}(t)^{\top}]$ prevent $\mathbf{W}$ from growing infinitely. The off-diagonal elements remove the contribution of the respective eigenvectors with larger eigenvalues in each row of (3).

The GHA has been applied in several studies to compute the PCs of natural images [41, 20, 36]. PCA image models have been used, for instance, for image coding and texture segmentation [41], and for explaining psychophysically derived orientation tuning curves [2].

[1]More precisely, the covariance matrix is defined as the expectation $E[\mathbf{x}\mathbf{x}^{\top}]$; $\overline{C}$ is an estimate based on a finite set of examples.

2.2 Kernel principal component analysis

Linear PCA is an appropriate model for data that are generated by a Gaussian distribution, or data that are best described by second-order correlations. In fact, PCA is based only on second-order correlations (cf. Eq. 1) such that no higher-order statistics can influence its result. It is well known, however, that the distribution of natural images is highly non-gaussian, and that all the "interesting" structures in images such as edges or corners cannot be described by second-order correlations [11]. This motivates the use of a nonlinear analysis technique that can capture higher-order dependencies in the data.

In KPCA, this nonlinearity is introduced by first mapping the data into another space F using a *nonlinear* map $\Phi : \mathbb{R}^N \rightarrow F$, before a standard linear PCA is carried out in F using the mapped examples $\Phi(\mathbf{x}_k)$ [43]. The map Φ and the space F are determined implicitly by the choice of a *kernel function* k which computes the dot product between two input examples $\mathbf{x}$ and $\mathbf{y}$ mapped into F via

$$k(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}). \tag{4}$$

One can show that if k is a so-called *positive definite kernel*, then there exists a map Φ into a dot product space F such that (4) holds. The space F then has the structure of a so-called *Reproducing Kernel Hilbert Space* (RKHS) [42].

The identity (4) is important for KPCA since PCA in F can be formulated entirely in terms of inner products of the mapped examples[2]. Thus, we can replace all inner products by evaluations of the kernel function. This has two important consequences: first, inner products in F can be evaluated without computing $\Phi(\mathbf{x})$ explicitly. This allows us to work with a very high-dimensional, possibly infinite-dimensional RKHS F. Second, if a positive definite kernel function is specified we need to know neither Φ nor F explicitly to perform KPCA since only inner products are used in the computations.

Commonly used examples of such positive definite kernel functions are the *polynomial kernel* of degree $d \in \mathbb{N}$, $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^d$ or the *Gaussian kernel* of width $\sigma > 0$, $k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2 / 2\sigma^2)$, each of them implying a different map and RKHS. The corresponding RKHSs are $\frac{(N+d-1)!}{d!(N-1)!}$ -dimensional for the polynomial kernel [42], and infinite-dimensional for the Gaussian kernel. For more examples of classes of kernels and detailed description of the properties of each kernel class, readers are referred to [42, 22].

Coming back to computing PCA in F, we can rewrite the covariance matrix of the mapped examples (Eq. 2) by stacking them into the matrix $\mathbf{\Phi} = \left(\Phi(\mathbf{x}_1)^\top, \ldots, \Phi(\mathbf{x}_l)^\top\right)^\top$ as

$$\overline{C} = \frac{1}{l} \mathbf{\Phi}^\top \mathbf{\Phi}, \tag{5}$$

assuming that the data are centered in F (i.e., $\sum_{k=1}^{l} \Phi(\mathbf{x}_k) = 0$).[3] We now have to find the eigenvalues $\lambda \geq 0$ and eigenvectors $\mathbf{v} \in F \setminus \{0\}$ satisfying

$$\lambda \mathbf{v} = \overline{C} \mathbf{v}. \tag{6}$$

Since all solutions $\mathbf{v}$ with $\lambda \neq 0$ lie within the span of $\{\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_l)\}$ (cf. footnote 2), we may consider the following equivalent problem

$$\lambda \mathbf{\Phi} \mathbf{v} = \mathbf{\Phi} \overline{C} \mathbf{v}, \tag{7}$$

and represent $\mathbf{v}$ in terms of an l-dimensional vector $\mathbf{q}$ as $\mathbf{v} = \mathbf{\Phi}^\top \mathbf{q}$. Combining this with (5) and (7) and defining an $l \times l$ kernel matrix $\mathbf{K}$ by $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^\top$ leads to $l\lambda \mathbf{K}\mathbf{q} = \mathbf{K}^2 \mathbf{q}$. The solution can be obtained by solving the *kernel eigenvalue problem* [43]

$$l\lambda \mathbf{q} = \mathbf{K}\mathbf{q}. \tag{8}$$

As we said before, the resulting kernel PCs (KPCs) are linear combinations of inner products of the data points, i.e., there is no need to compute Φ explicitly since everything can be expressed in terms of kernel functions. In

[2]This can be verified by replacing $\mathbf{x}$ in Eq. 1 with $\Phi(\mathbf{x})$ and substituting the right side of Eq. 1 for $\overline{C}$ in Eq. 2. Then, all solutions $\mathbf{v}$ with $\lambda \neq 0$ must lie in the span of $\{\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_l)\}$ in F; hence, Eq. 2 is equivalent to $\lambda(\Phi(\mathbf{x}_k) \cdot \mathbf{v}) = \Phi(\mathbf{x}_k) \cdot \overline{C}\mathbf{v}$ for all $k = 1, \ldots, l$.

[3]The centering issue will be dealt with later.

contrast to PCA, ICA or sparse coding, KPCs consist of nonlinear interactions between the data points. In terms of image modeling, this means that images are modeled as nonlinear combinations of the input images by using the kernel function. As a consequence, KPCA becomes also sensitive to higher-order statistical properties of the input, i.e., the obtained KPCs depend also on interactions between more than two pixels. The exact nature of the interactions that can be modelled depends to a large degree on the chosen kernel and is unknown in most cases. Certain polynomial kernels of degree d, for instance, are capable of modeling all multiplicative interactions between up to d pixels [12]. Note that the modeling capability of PCA is retained by KPCA: It allows for a truncated expansion in only a few KPCs. However, the truncated expansion minimizes the reconstruction error in the RKHS, not in the input space as in linear PCA. This seems like an odd optimization principle, but it is not clear from the outset whether the Euclidian error norm is a better error measure for such complex objects as images. In fact, several applications have shown that KPCA can lead to better models than PCA [32, 42, 27].

The large variety of used kernel functions [42, 22] indicates that there is no single best kernel for all possible applications. Accordingly, the choice of a good kernel depends on the problem of interest. While there exist several methods for choosing optimal kernels for a given supervised learning problem [48, 7, 10, 49], the best choice of the kernel in unsupervised learning remains elusive because of the lack of proper evaluation criteria. In our experiments, we used the Gaussian kernel. Since there exists no principled method for finding the kernel width σ in unsupervised learning, we chose the default value $\sigma = 1$ which lead to visually satisfying results. For a number of reasons, the Gaussian is considered a default "general purpose kernel" in the kernel methods community. Among these are its universal approximation capabilities (the associated RKHS is dense in the space of continuous functions, cf. [45]), its translation invariance, and its desirable regularization properties: it can be shown to correspond to a smoothness regularizer which penalizes derivatives of all orders (see e.g., [16, 42]). To see this, first note that KPCA can be rewritten as an optimization problem. For centered data, the first principal component is the minimal norm vector $\mathbf{v}$ subject to the constraint that the variance of $\mathbf{v}^\top \Phi(\mathbf{x}_i)$ is 1 on the training set. Second, note that the minimal norm can be interpreted as a smoothness regularizer. We have $\|\mathbf{v}\|^2 = \left(\mathbf{\Phi}^\top \mathbf{q}\right)^\top \mathbf{\Phi}^\top \mathbf{q} = \sum_{i,j} q_i q_j k(\mathbf{x}_i, \mathbf{x}_j)$. For the Gaussian kernel, we can rewrite this as

$$\sum_{i,j} q_i q_j k(\mathbf{x}_i, \mathbf{x}_j) = \|Pf\|^2, \tag{9}$$

where $f(\mathbf{x}) = \sum_i q_i k(\mathbf{x}_i, \mathbf{x})$ is the projection onto $\mathbf{v}$ in F, i.e., the feature extraction function, and P is a derivative operator of all orders [38] — therefore, KPCA with the Gaussian kernel maximizes the smoothness of the feature extractor f, subject to a variance constraint.

3 Iterative KPCA

3.1 Kernel Hebbian Algorithm

The size of the kernel matrix scales with the square of the number of examples. Thus, it becomes computationally infeasible to directly solve the kernel eigenvalue problem for a large number of examples. As noted in the introduction, a similar problem occurs with linear PCA when the covariance matrix becomes large. This motivated the introduction of the GHA which does not require the storage and inversion of the covariance matrix. Here, we propose a similar approach by reformulating the GHA in a RKHS to obtain a memory-efficient approximation of KPCA.

The GHA update rule of Eq. (3) is represented in the RKHS F as

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \eta(t)(\mathbf{y}(t)\Phi(\mathbf{x}(t))^\top - \mathrm{LT}[\mathbf{y}(t)\mathbf{y}(t)^\top]\mathbf{W}(t)), \tag{10}$$

where the rows of $\mathbf{W}(t)$ are now vectors in F and $\mathbf{y}(t) = \mathbf{W}(t)\Phi(\mathbf{x}(t))$. $\Phi(\mathbf{x}(t))$ is a pattern presented at time t which is randomly selected from the mapped data points $\{\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_l)\}$. For notational convenience we assume that there is a function $J(t)$ which maps t to $i \in \{1, \ldots, l\}$ ensuring $\Phi(\mathbf{x}(J(t))) = \Phi(\mathbf{x}_i)$ and denote

$\Phi(\mathbf{x}(J(t)))$ simply by $\Phi(\mathbf{x}(t))$[4]. From the direct KPCA solution, it is known that $\mathbf{w}(t)$ can be expanded in the mapped data points $\Phi(\mathbf{x}_i)$. This restricts the search space to linear combinations of the $\Phi(\mathbf{x}_i)$ such that $\mathbf{W}(t)$ can be expressed as

$$\mathbf{W}(t) = \mathbf{A}(t)\mathbf{\Phi} \tag{11}$$

with an $r \times l$ matrix $\mathbf{A}(t) = (\mathbf{a}_1(t)^\top, \ldots, \mathbf{a}_r(t)^\top)^\top$ of expansion coefficients. The ith row $\mathbf{a}_i = (a_{i1}, \ldots, a_{il})$ of $\mathbf{A}(t)$ corresponds to the expansion coefficients of the ith eigenvector of $\mathbf{K}$ in the $\Phi(\mathbf{x}_i)$, i.e., $\mathbf{w}_i(t) = \mathbf{\Phi}^\top \mathbf{a}_i(t)$. Using this representation, the update rule becomes

$$\mathbf{A}(t+1)\mathbf{\Phi} = \mathbf{A}(t)\mathbf{\Phi} + \eta(t)\left(\mathbf{y}(t)\Phi(\mathbf{x}(t))^\top - \mathrm{LT}[\mathbf{y}(t)\mathbf{y}(t)^\top]\mathbf{A}(t)\mathbf{\Phi}\right). \tag{12}$$

The mapped data points $\Phi(\mathbf{x}(t))$ can be represented as $\Phi(\mathbf{x}(t)) = \mathbf{\Phi}^\top \mathbf{b}(t)$ with a canonical unit vector $\mathbf{b}(t) = (0, \ldots, 1, \ldots, 0)^\top$ in $\mathbb{R}^l$ (only the $J(t)$-th element is 1). Using this notation, the update rule can be written solely in terms of the expansion coefficients as

$$\mathbf{A}(t+1) = \mathbf{A}(t) + \eta(t)\left(\mathbf{y}(t)\mathbf{b}(t)^\top - \mathrm{LT}[\mathbf{y}(t)\mathbf{y}(t)^\top]\mathbf{A}(t)\right). \tag{13}$$

Representing (13) in component-wise form gives

$$a_{ij}(t+1) = \begin{cases} a_{ij}(t) + \eta y_i(t) - \eta y_i(t)\sum_{k=1}^{i} a_{kj}(t) y_k(t) & \text{if} \quad J(t) = j \\ a_{ij}(t) - \eta y_i(t)\sum_{k=1}^{i} a_{kj}(t) y_k(t) & \text{otherwise,} \end{cases} \tag{14}$$

where

$$y_i(t) = \sum_{k=1}^{l} a_{ik}(t)\Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}(t)) = \sum_{k=1}^{l} a_{ik}(t) k(\mathbf{x}_k, \mathbf{x}(t)). \tag{15}$$

This does not require $\Phi(\mathbf{x})$ in explicit form, thus providing a practical implementation of the GHA in F. For stationary input distributions, the learning rate is typically chosen as $\eta(t) = \frac{1}{t}$ since in this case local convergence can be guaranteed (see paragraph on convergence below). In many practical applications [21], especially for nonstationary input in an online setting, η can be set to a small constant value to keep the solution adaptive. $\mathbf{A}$ should be randomly initialized to avoid possible convergence problems (see below).

During the derivation of (13), it was assumed that the data are centered in F which is not true in general unless explicit centering is performed. Centering can be done by subtracting the mean of the data from each pattern. Then each pattern $\Phi(\mathbf{x}(t))$ is replaced by $\widetilde{\Phi}(\mathbf{x}(t)) = \Phi(\mathbf{x}(t)) - \overline{\Phi}(\mathbf{x})$, where $\overline{\Phi}(\mathbf{x})$ is the sample mean $\overline{\Phi}(\mathbf{x}) = \frac{1}{l}\sum_{k=1}^{l} \Phi(\mathbf{x}_k)$. The centered algorithm remains the same as in (14) except that Eq. (15) has to be replaced by the more complicated expression

$$y_i(t) = \sum_{k=1}^{l} a_{ik}(t)(k(\mathbf{x}(t), \mathbf{x}_k) - \bar{k}(\mathbf{x}_k)) - \overline{a}_i(t)\sum_{k=1}^{l}(k(\mathbf{x}(t), \mathbf{x}_k) - \bar{k}(\mathbf{x}_k)). \tag{16}$$

with $\bar{k}(\mathbf{x}_k) = \frac{1}{l}\sum_{m=1}^{l} k(\mathbf{x}_m, \mathbf{x}_k)$ and $\overline{a}_i(t) = \frac{1}{l}\sum_{m=1}^{l} a_{im}(t)$.[5] $\bar{k}(\mathbf{x}_k)$ for each $k, i = 1, \ldots, l$ can be calculated once at the beginning of the whole procedure. This is directly applicable in a batch setting (i.e., the patterns are fixed and known in advance); in an online setting, one should instead use a sliding mean, in order to be able to adapt to changes in the distribution. For the details of the online algorithm, readers are referred to [26]. It should be noted that not only in training but also in testing, each pattern should be centered using the training mean.

[4]The original development of the GHA relies on infinitely many examples to achieve convergence. Since we are dealing with finite sample sizes we construct the sequence $\{\mathbf{x}(t)\}$ by using a $J(t)$ that concatenates random permutations of $\{\mathbf{x}_i\}$ until a sufficient degree of convergence is reached. Typically, this will require repeated sweeps through the entire dataset.

[5]A Matlab example implementation can be downloaded at
http://www.kyb.tuebingen.mpg.de/prjs/comp_vision_robotics/nat_image/kha/kha.htm.

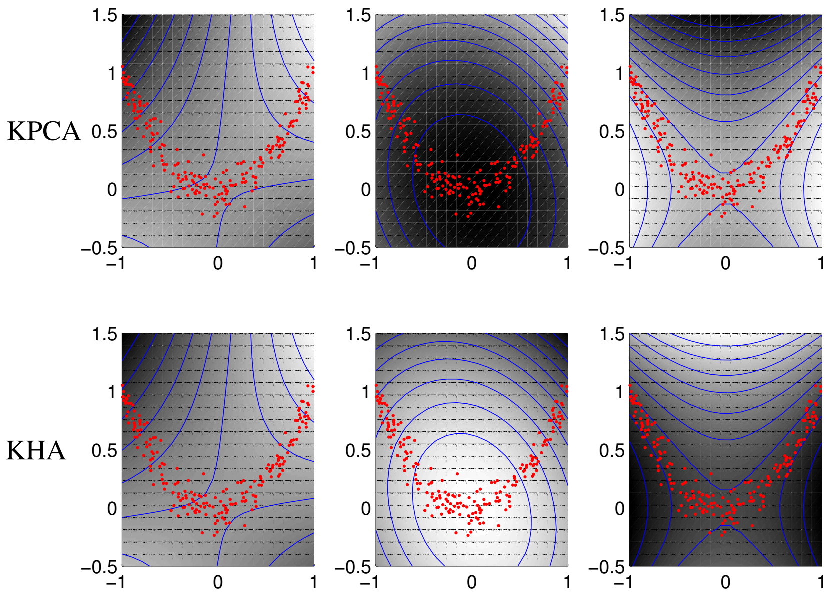


Figure 1: Two dimensional toy example, with data generated in the following way: x-values have uniform distribution in $[-1, 1]$, y-values are generated from $y_i = -x_i^2 + \xi$, where ξ is normal noise with standard deviation 0.2. From left to right, contour lines of constant value of the first three PCs for 150 data points obtained from KPCA and KHA with degree-2 polynomial kernel. The KHA results were obtained after 150,000 updates. The learning rate was constant and set to 0.05.

Complexity. At each update of Eq. (13), we need to store the $r \times l$ coefficient matrix $\mathbf{A}$ (r is the number of PCs to be computed, l the number of examples) and the training set requiring $\mathbf{O}(l \times N)$ where N is the dimensionality of the input space. The entire memory complexity of the KHA is then $\mathbf{O}(r \times l + l \times N)$ which, for large sample sizes, compares favorably with the $\mathbf{O}(l^2)$ complexity of standard KPCA. However, this comes at the price that now time complexity depends on the dimensionality of the input: if we assume that the evaluation of the kernel function typically is of time complexity $\mathbf{O}(N)$ (this is the case for the Gaussian and the polynomial Kernel), then the time complexity of the KHA for each presentation of a pattern is $\mathbf{O}(r \times l \times N + r^2 \times l)$. For high-dimensional input, time complexity can be lowered by precomputing and storing part of the kernel matrix. If we store a $l' \times l'$-size portion of the kernel matrix, the time and the memory complexity become $\mathbf{O}(r \times (l - l') \times N + r^2 \times l)$ and $\mathbf{O}(r \times l + l \times N + l' \times l')$, respectively. The time complexity reduces to $\mathbf{O}(r \times l + r^2 \times l)$ when we store the entire kernel matrix, as KPCA does, but this, of course, would make the primary motivation for using the KHA obsolete. Clearly, the entire time complexity of the KHA strongly depends on the number of updates. This, in turn, will depend to a large degree on the problem at hand and the required accuracy of the KPCs. Complicated problems with only a small number of examples may require a large number of repeated sweeps through all data points before sufficient accuracy is achieved. In our experiments, we had to sweep between 40 to 800 times through the entire dataset to obtain visually satisfying results.

Convergence. Unfortunately, there are no results in the literature about the convergence speed of the GHA or its global convergence properties for general initial conditions. There is, however, a theorem by Oja (for $r = 1$) and Sanger (for $r \geq 1$) on the local convergence of the GHA [34, 41][6] that directly carries over to the KHA. Their results are based on the stability analysis of an associated ordinary differential equation. They could show

[6]In [41], Sanger actually claimed to have found a proof for global convergence. Later authors [35, 50] noticed that his convergence proof is only valid locally around the equilibrium points. There are, however, global convergence proofs for closely related learning rules [8, 50].

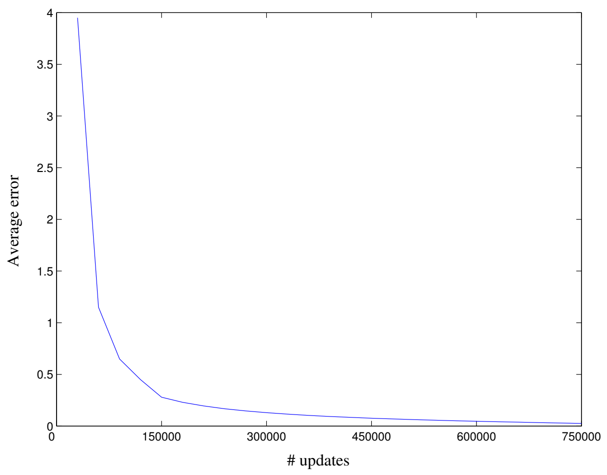


Figure 2: The average l^2 distance in the RKHS of the first three eigenvectors and their estimates obtained by the KHA. The sign of each PC has been corrected for the computation.

that under certain conditions the asymptotically stable limits of this differential equation are possible limits of the GHA. Since then, extensive simulations confirmed that the limits of the GHA and its associated differential equation are “usually” the same, i.e., the conditions of Oja’s and Sanger’s theorems usually hold in practice [35].

The convergence properties of the GHA are equally valid for the KHA. This can be stated formally as

Theorem 1 *For a finite set of centered data, $\mathbf{A}$ initially in general position, and learning rate $\eta(t) = \frac{1}{t}$, the rows of $\mathbf{W}$ in (10) (and equivalently the rows of $\mathbf{A}$ in (13)) will approach the first r normalized eigenvectors of the correlation matrix $\overline{C}$ in the RKHS, ordered by decreasing eigenvalue.*

Proof: Requiring the data and $\mathbf{A}$ to be initially in general position excludes pathological configurations in which the KHA cannot converge. If $\mathbf{A}$ were initialized such that its rows are the zero vector or orthogonal to the eigenvectors, then it would never change irrespective of the data. Similarly, if the input data was always orthogonal to the initialization of $\mathbf{A}$, we would run into problems. For the proof, we note that for a finite set of data $\{\mathbf{x}_1, \ldots, \mathbf{x}_l\}$, we can induce from a given kernel k a *kernel PCA map* into $\mathbb{R}^l$ [42]

$$\Phi_l : \mathbf{x} \rightarrow \mathbf{K}^{-\frac{1}{2}}(k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_l)),$$

satisfying

$$\Phi_l(\mathbf{x}_i)^\top \Phi_l(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j).$$

By applying the GHA in the space spanned by the kernel PCA map, (i.e., replacing each occurrence of $\Phi(\mathbf{x})$ with $\Phi_l(\mathbf{x})$ in Eq. 10, and noting that this time, $\mathbf{W}$ lies in $\mathbb{R}^l$ rather than in F), we obtain an algorithm in $\mathbb{R}^l$ which is exactly equivalent to the KHA in F. The local convergence of the KHA then follows from the local convergence of the GHA for $\eta(t) = \frac{1}{t}$ in $\mathbb{R}^l$ as stated in Theorem 1 in [41]. $\square$

It should be noted that, in practice, this approach cannot be taken to *construct* an iterative algorithm since it involves the computation of $\mathbf{K}^{-\frac{1}{2}}$. Fig. 1 shows the first three PCs of a toy data set, extracted by both KPCA and KHA with a polynomial kernel of degree 2. The visual similarity of the PCs (ignoring the sign difference) obtained from the KPCA and KHA illustrates the capability of KHA to approximate the full KPCA solution. The convergence behavior of the KHA in this example is shown in Fig. 2.

3.2 Applications

The KHA extends the range of possible KPCA applications considerably since now larger datasets can be processed. We will demonstrate the potential of the KHA in two image reconstruction tasks, single-frame super-resolution and denoising.

Single-frame super-resolution refers to the task of constructing a high-resolution enlargement of a *single*[7] low-resolution, pixel-based image. In contrast to the usual interpolation and sharpening (e.g. [25]), new high-resolution details are added to the reconstruction. This can only be done by relying on prior knowledge about the image class to be processed.

In previous work, single-frame super-resolution was mainly done in a *supervised* learning setting [23, 13]: During the training phase, pairs of low-resolution patches and the corresponding high-resolution patches are collected. In the super-resolution phase, each low-resolution patch of the input image is compared to the stored low-resolution patches and the high-resolution patch corresponding to the nearest low-resolution patch is selected. Here, we propose an alternative approach to super-resolution based on KPCA which is an *unsupervised* learning method. Instead of encoding a fixed relationship between pairs of high- and low-resolution image patches, we rely on the generic model of the high-resolution images obtained from KPCA. To reconstruct a super-resolution image from a low-resolution image which was *not* contained in the training set, we first scale up the image $\mathbf{x}$ to the same size as the high-resolution training images, map the scaled image $S\mathbf{x}$ into the RKHS F using Φ, then perform centering, and project it into the KPCA subspace corresponding to a limited number r of KPCs to get $P\Phi(S\mathbf{x})$:

$$P\Phi(S\mathbf{x}) = \sum_{i=1}^{r} \left(\mathbf{w}_i \cdot (\Phi(S\mathbf{x}) - \overline{\Phi}(\mathbf{x}))\right) \mathbf{w}_i \tag{17}$$

where $\mathbf{w}_i$ is the KHA estimate of the i-th eigenvector of $\overline{C}$

$$\mathbf{w}_i = \sum_k a_{ik} \Phi(\mathbf{x}_k)$$

and $\overline{\Phi}(\mathbf{x}) = \frac{1}{l}\sum_{k=1}^{l} \Phi(\mathbf{x}_k)$ is the sample average. In terms of the kernel function, this expression can be written as

$$P\Phi(S\mathbf{x}) = \sum_{i=1}^{r} \left(\sum_{m=1}^{l} q_{im} k(\mathbf{x}_m, S\mathbf{x}) - \frac{1}{l} \sum_{m,n=1}^{l} q_{im} k(\mathbf{x}_m, \mathbf{x}_n) \right) \mathbf{w}_i. \tag{18}$$

Via the projection P, the scaled image is mapped to an image which is consistent with the statistics of the high-resolution training images. However, at that point the projection is still centered and lives in F which can be infinite-dimensional. To obtain the reconstruction in the input space, the projection $P\Phi(S\mathbf{x})$ is first decentered by adding the sample average $\overline{\Phi}(\mathbf{x})$

$$\begin{aligned} P\Phi(S\mathbf{x}) + \overline{\Phi}(\mathbf{x}) &= \sum_{k=1}^{l} g_k \Phi(\mathbf{x}_k) + \frac{1}{l}\sum_{k=1}^{l} \Phi(\mathbf{x}_k) \\ &= \sum_{k=1}^{l} (g_k + \frac{1}{l}) \Phi(\mathbf{x}_k) \end{aligned} \tag{19}$$

[7]This should not be confused with *aggregation from multiple frames* where a single high-resolution frame is extracted from a sequence of low resolution images (cf., e.g. [3]).

where

$$\begin{aligned} g_k &= \sum_{i=1}^{r} \left(\mathbf{w}_i \cdot (\Phi(S\mathbf{x}) - \overline{\Phi}(\mathbf{x}))\right) a_{ik} \\ &= \sum_{i=1}^{r} \left(\sum_{m=1}^{l} a_{im} k(\mathbf{x}_m, S\mathbf{x}) - \frac{1}{l} \sum_{m,n=1}^{l} a_{im} k(\mathbf{x}_m, \mathbf{x}_n) \right) a_{ik}. \end{aligned} \tag{20}$$

In the next step, we need to find a corresponding point in $\mathbb{R}^N$ — this is a preimage problem. To solve it, we minimize $\|P\Phi(S\mathbf{x}) + \overline{\Phi}(\mathbf{x}) - \Phi(\mathbf{z})\|^2$ over $\mathbf{z} \in \mathbb{R}^N$.[8] Note that this objective function can be computed in terms of inner products and thus in terms of the kernel (4). For the minimization, we use gradient descent [6] with starting points obtained from the method of [28].

Image denoising refers to the task of constructing a noise-free image from a noisy input image. Since image denoising is a standard problem in the image processing community, the readers are referred to [17] for a brief survey. From the point of view of a KPCA model, image denoising can be regarded as the same problem as image super-resolution: the projection method from the super-resolution task can be applied to image denoising as well. The only difference is that the scaling of input image is omitted which is not necessary for denoising.

We consider two scenarios in our experiments. In the first scenario, the KPCA model is trained on clean image patterns which are distinct from the test images, as it was done before in the super-resolution task. KPCA has already been applied in this scenario to digit image denoising and demonstrated promising results [32]. This was possible because the class of digit images is small enough for the direct computation of KPCA. Presently, we focus on more complex image classes such as faces that require a larger number of training examples. In these cases, the direct computation of KPCA is no more feasible but requires the use of the KHA.

In the second scenario, no clean training images are required. Instead, it is assumed that the noise mainly contaminates KPCs corresponding to small eigenvalues. Thus, a truncated KPC expansion of the noisy image as obtained from the KHA leads automatically to a denoising effect. In this scenario, the KHA is trained and tested on the same dataset. This approach is similar to wavelet-based methods [44, 37] and linear PCA-based methods [1] for image denoising in the sense that the objective is to find a good feature space in which the noise shows low power or is concentrated on a small subspace. The main difference, however, is that the KPCA model finds features that are data-dependent (in contrast to wavelet-based methods) and nonlinear (in contrast to linear PCA-based methods).

4 Experiments

From a machine learning point of view, an image is simply a point in an image space whose dimensionality is equal to the number of pixels in the image. If the class of images is moderately constrained (e.g. face images) or if the image size is small enough, the learning of a KPCA model poses no serious problems since the image space can be sampled densely enough. However, for rather large images with arbitrarily high complexity (e.g., natural images) the necessarily limited amount of training data leads to *overfitting*, where one obtains a model which explains the training data perfectly but fails to generalize to unknown data. In these cases, we adopt a *patch-based* approach where a large image is regarded as a composition of patches (small sub-images). Accordingly, this section describes two distinct sets of experiments according to the classes of images considered: the *single-patch* case regards a small image as a single pattern and the *multi-patch* case regards a large image as a set of small patches.

The best choice of the learning parameter η and the number of updates for the convergence of the KHA depends on the data set. In our case, the updates finished when the squared distance between two solutions from consecutive updates was smaller than a given threshold. We used a fixed learning rate η of 0.05 throughout the experiments.

[8]Note that *exact* preimages will usually not exist [42].

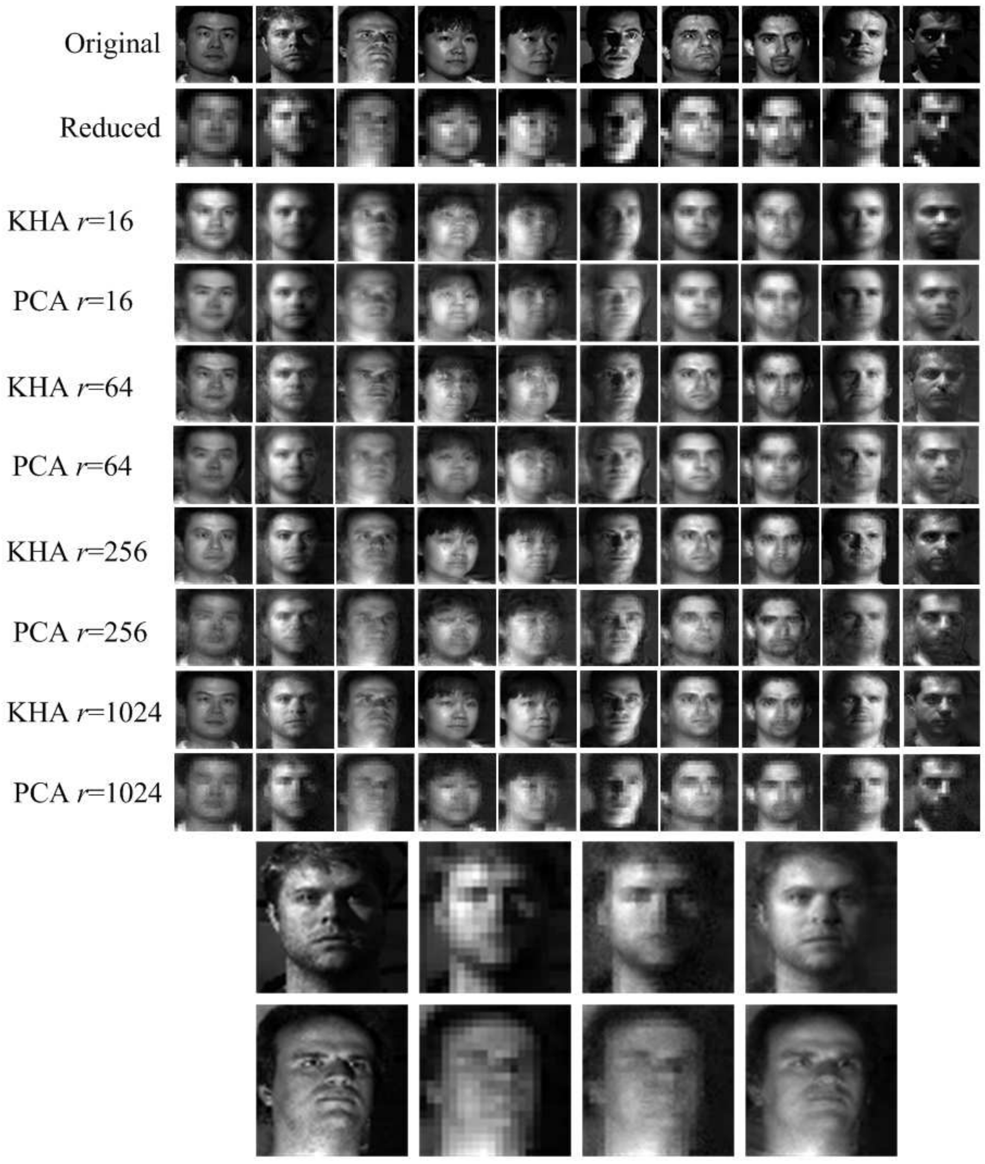


Figure 3: Face reconstruction based on PCA and KHA for varying number r of PCs. Below, enlarged reconstruction examples are shown for the second face (top) and third face (bottom) in the order 1. original image, 2. reduced resolution image, 3. PCA reconstruction with 1024 components, 4. KHA reconstruction with 1024 components. Note that, in contrast to the KHA, PCA tries to approximate the *reduced* resolution image whereas the KHA tries to come up with a high-resolution image that is consistent with the statistics of the original image.

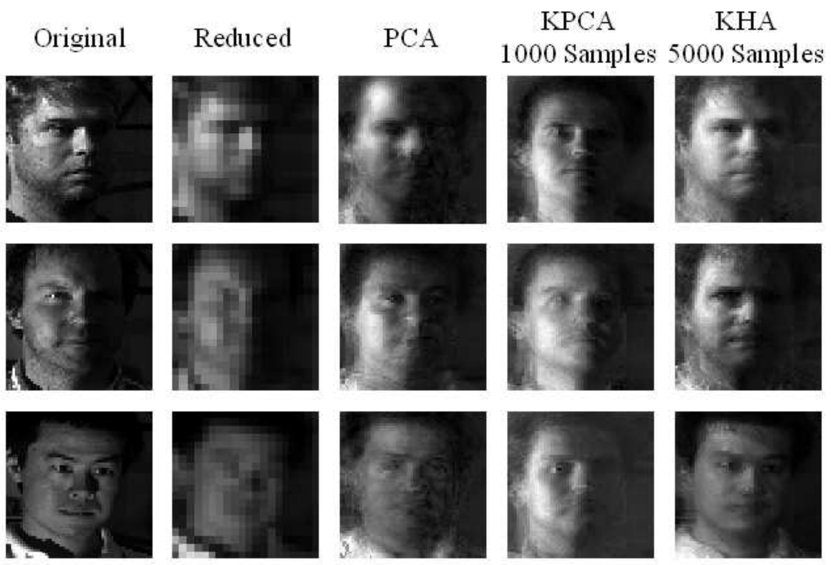


Figure 4: Face reconstruction examples obtained from KPCA and KHA with 256 PCs trained on 1,000 and 5,000 examples, respectively. Occasional erroneous reconstruction of images indicates that KPCA requires a large amount of data to properly sample the underlying structure.

Single-patch case: Super-resolution and de-noising of face images. Here we consider a database of face images. The Yale Face Database B contains 5,760 images of 10 persons [15]. 5,000 images were used for training while 10 randomly selected images which are disjoint from the training set were used to test the method (note, however, as there are only 10 persons in the database, the same person, in different views, is likely to occur in training and test set). Since the direct computation of KPCA for this dataset is not practical on standard hardware, the KHA was utilized. In training, (60×60)-sized face images were fed into the KHA using a Gaussian kernel with $\sigma = 1$.

For the super-resolution experiments, the test images were blurred and subsampled to a 20×20 grid and scaled up to the original scale (60×60) by turning each pixel into a 3×3 square of identical pixels, before doing the reconstruction. To avoid overflow during the computation of exponential term in the kernel, each pixel value is normalized into the interval around zero ($[-0.05, 0.05]$). Fig. 3 shows reconstruction examples obtained using different numbers of PCs. For comparison, reconstructions obtained by linear PCA are also displayed. While the images obtained from linear PCA look like somewhat uncontrolled superpositions of different face images, the images obtained from its nonlinear counterpart (KHA) are more face-like. In spite of its less realistic results, linear PCA was slightly better than the KHA in terms of the mean squared error (average 9.20 and 8.48 for KHA and PCA, respectively for 100 PCs). This stems from the characteristics of PCA which is constructed to minimize the MSE while KHA is not concerned with the MSE in the input space. Instead, it seems to force the images to be contained in the manifold of face images. Similar observations have been reported in [31]. Interestingly, a small number of examples and a sparse sampling of this manifold can have the consequence that the KHA reconstruction looks like the face of person different from the one used to generate the test image. In a sense, this means that the errors performed by KPCA are errors *along* the manifold of faces. Fig. 4 demonstrates this effect by comparing results from KPCA on 1,000 example images (corresponding to a sparse sampling of the face manifold) and KHA on 5,000 training images (denser sampling). As the examples show, some of the misreconstructions that are made by KPCA due to the lack of training examples were corrected by the KHA using a larger training set.

To see the effect of varying input image resolution on the reconstruction result, another set of experiments

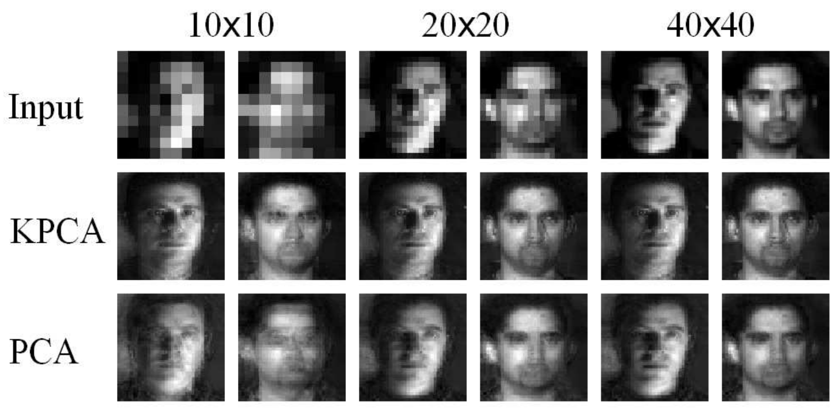


Figure 5: Face reconstruction based on PCA and KHA with 256 PCs with different input image resolutions.

has been performed with different resolutions of 10×10, 20×20, and 40×40 as shown in Fig. 5. A graceful degradation of reconstruction performance was observed from both PCA and KPCA models as the input image resolution decreases. However, the KHA results look uniformly better than those of PCA, especially when the input image is very small (10×10).

For denoising of face images, the first scenario is considered (c.f. 3.2). Gaussian noise was added to the test images of Fig. 3 with an average SNR of 3.12dB. We applied the KPCA model used for face image super-resolution with no additional training and no modification of the experimental setting, except for the omission of the smoothing and resizing steps. For comparison, linear PCA, Wiener filters,[9] and wavelet-based methods[10] were also applied. The results (Fig. 6) indicate that the proposed model is general enough to be applied to more than one specific application, while still having an acceptable performance in each of them. Compared to other methods, the SNR of the KHA reconstruction was not the best. However, visual inspection shows that the KHA solutions are more realistic. For example, the KHA reconstruction of the first face image in Fig. 6 is much brighter than the original which results in a deteriorated SNR mainly because of the difference in average brightness. However, it is visually more face-like than the other reconstructions.

By moving along the directions of the eigenvectors in the RKHS and computing the corresponding preimages one can directly visualize what the KPCA model has learned. As an example, Fig. 7 shows the preimages along the principal axes corresponding to the three largest eigenvalues. In contrast to linear eigenfaces (i.e., eigenvectors of face images) [46], the nonlinear eigenfaces obtained from the KHA are more face-like. This can be explained if we assume that the data have a cluster structure and that the Gaussian kernel parameter σ is small compared to the distances between the clusters, but large compared to the distances within the clusters. Then KPCA becomes similar to linear PCA performed on each cluster of similar (close in terms of the distance in input space) images: in this case, the kernel matrix is almost block diagonal, and accordingly, the eigenvectors of the kernel matrix become similar to the eigenvectors of each cluster. As a result, the eigenvectors are superpositions of similar images (cf. Eq. (7)) such that their preimages do not show the superposition artifacts usually encountered in linear eigenfaces. On the other hand, the distance structure within a block (a set of patterns close to each other) is similar to the Euclidean distance in the input space if the block size is small enough.[11]

[9]We applied Matlab's spatially adaptive Wiener filter with window size (3×3), (5×5), and (7×7). The best result (in terms of SNR) was obtained for window size (5×5).

[10]Matlab's wavelet toolbox was utilized. Denoising was done by thresholding in each wavelet basis. The wavelet bases investigated include Haar and Symmlets4, 6, and 8. The threshold value varied from 20 to 80 with an interval of 5. The best result was obtained with Symmlet4 and threshold 45.

[11]This becomes evident when the function $f(z) = e^z$ is expanded in the Taylor series about zero. When the absolute values of z approaches to zero, the high-order terms in the series tend to vanish such that f becomes linear. Thus the kernel is approximately $1 - \|\mathbf{x} - \mathbf{y}\|^2$, a conditionally positive definite kernel which for KPCA is equal to $\mathbf{x} \cdot \mathbf{y}$ [42].

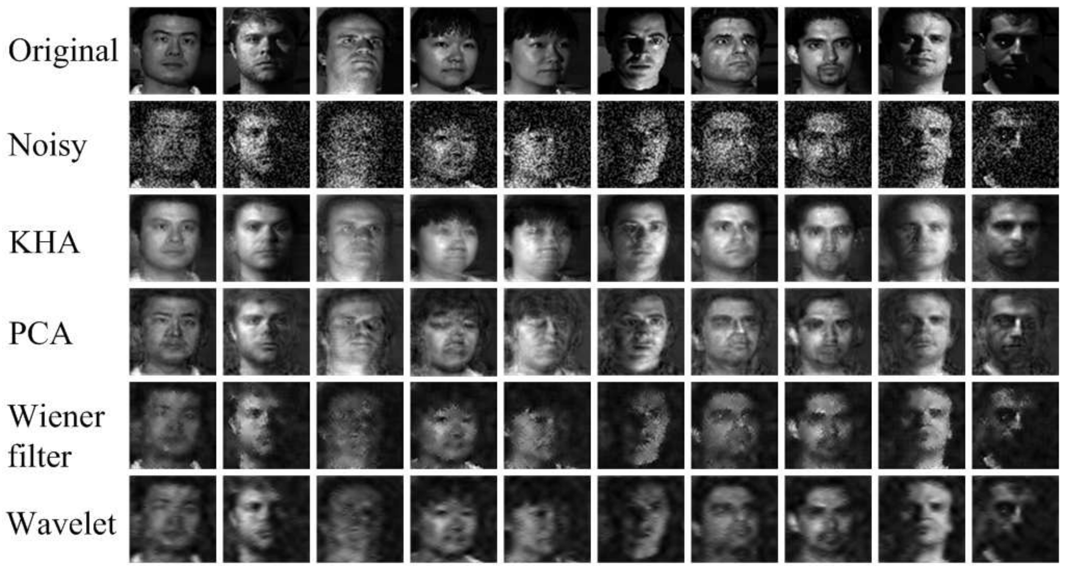


Figure 6: Face image denoising based on PCA and KHA with 256 PCs, Wiener filter, and wavelet from Matlab. The SNR value was 7.58dB, 8.29dB, 8.45dB, and 8.31dB, respectively.

In addition, within the projection interval defined by the standard deviation of sampling points along the eigenvector (Fig. 7), the eigenfaces show a morphing behavior from one face class to another as we move along the principal axis. The corresponding trajectory is not entirely contained in the training set, but is actually *learned* from the training samples.[12] Outside this region the sampling of patterns becomes very sparse as indicated by the increasing distance to the nearest training pattern in Fig. 8. Here, we observe a rather uncontrolled superposition of face images similar to linear PCA. This supports the previously mentioned manifold interpretation: assuming that the data are sampled densely enough, the KPCA image model reconstructs the manifold defined by the image class. Note that it has recently been pointed out that for certain kernels, KPCA corresponds to several known manifold dimensionality reduction algorithms [19].

Multi-patch super-resolution of natural images. For a realistic natural image super-resolution, we adopt the method of [13], where the large image is decomposed into its low-frequency components and a set of small patches containing the local high-frequency information. Whereas Freeman et. al. [13] use a nearest neighbor classifier to select appropriate high-frequency patches in the super-resolution phase, we replace this classifier by the projection step described above. During the training stage, images are high-pass filtered and a set of image patches are collected from the resulting high-frequency images. These image patches are contrast-normalized [13] and then fed into the KHA. In the super-resolution phase, the input image is rescaled to the original resolution using bicubic interpolation and band-pass filtered to remove the low-frequency components. Then, the resulting high-frequency component image is divided into a set of small image patches each of which is reconstructed in the same way as in single patch super-resolution. The resulting image contains only high-frequency components which are then superimposed on the bicubic interpolation to give the final reconstruction.

The KHA was trained on a set of 10,000 (12×12)-sized image patches obtained from the images in Fig. 9. As above, the σ parameter was set to a rather small value (1) to capture the nonlinear structure of the images. The reconstruction of the high-frequency image is then obtained based on the first 200 KPCs. This choice reflects a good compromise between performance and computational cost. Using more PCs, we obtained results which are as good or slightly better. At present, we have no principled method for choosing the optimal number of PCs.

[12]Similar observations have been reported in [39] where KPCA was used to model 3D objects from 2D views.

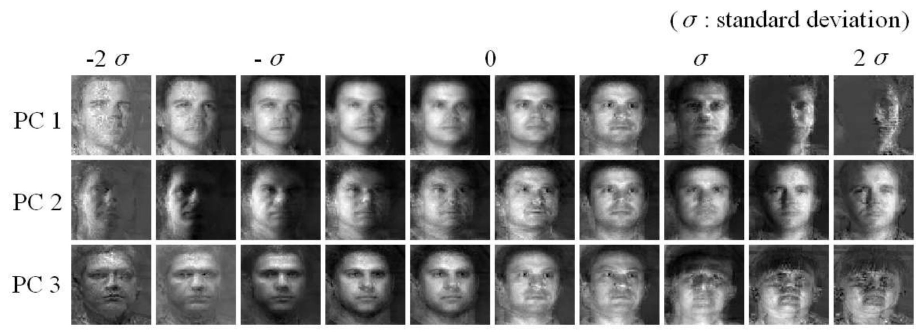


Figure 7: Preimages of the first three eigenvectors with RKHS projections varying from -2σ to 2σ (where σ is the standard deviation along each principal axis). Note that moving from $-\sigma$ to σ generates a morph between two faces.

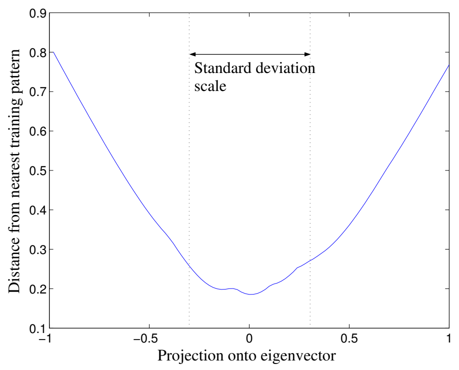


Figure 8: RKHS distance to the nearest training pattern when moving along the third principal axis.

The number 200 is motivated by the fact that in the face image super-resolution experiments, the performance of KPCA appeared to saturate around that number. In general, however, the optimal number to choose can be expected to depend on the problem at hand and on the amount of computational resources one is able to invest. When applied to non-overlapping patches, the resulting image as a whole shows a block structure since each patch is reconstructed independently of its neighborhood. To reduce this effect, the patches are chosen to overlap into their neighbors such that the overlapping regions can be averaged.

A (396×528)-size image not contained in the training set was used for testing. The (198×264)-sized low-resolution image was obtained by blurring and subsampling. Fig. 10 shows the super-resolution result. The final reconstruction was post-processed using high-boost filtering [17] to enhance the edges that become slightly blurred since only the first 200 KPCs are used in the reconstruction. It should be noted that the original KHA reconstruction of the high-frequency components still contains blocking artifacts even with the use of overlapping patches. This, however, does not severely degrade the final result since the overall structure is contained in the low frequency input image and the KHA reconstruction only adds the missing high-frequency information. Regarding more advanced techniques for the removal of blocking artifacts, readers are referred to [13] where the spatial relationship between patches is modeled based on Markov random fields (MRFs).

Fig. 11 shows more super-resolution results. The low resolution image is obtained in the same way as in Fig. 10. For comparison, bicubic interpolation and a supervised learning technique based on a nearest neighbor classifier such as the one used by [13] have also been applied. Note, however, that we did not implement the additional processing stage of [13] that takes into account spatial context in the matching of high- and low-resolution

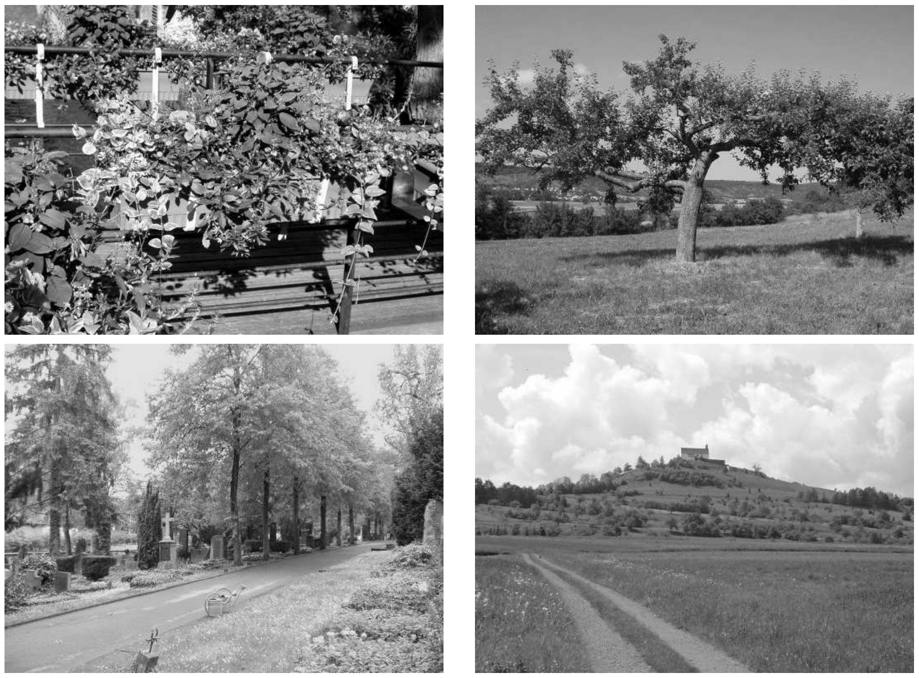


Figure 9: Training images of size 396×528. The training patterns are obtained by sampling 2,500 points at random from each image.

image patches. Again, for all the methods the final reconstructions are high-boost filtered. In comparison to image stretching (Fig. 11.b), bicubic interpolation (Fig. 11.c) produces far better results. However simple edge enhancement without any prior knowledge failed to completely remove the blurring effect. The two learning-based methods show a better capability in recovering the complex local structure. This, however, comes at the price of a certain specialization to the encoded image class: whereas interpolation can be applied to any image, we found in our experiments that a KPCA model of faces performed poorly in reconstructing natural images and vice versa.

As shown in Fig. 11 the KHA and the nearest neighbor-based method showed a comparable performance. Some insight into the differences in the modeling capabilities of the two methods can be gained by enlarging the size of reconstruction patch and performing super-resolution directly on the raw images. For this, a new KHA image model was trained on raw image patches without high-pass filtering. Then, during the super-resolution phase, the input image was decomposed into (16×16) patches which were reconstructed independently of the neighbouring patches. For comparison, the nearest neighbor-based model was trained in the same way. As can be seen in Fig. 12.e, the two learning-based approaches again show better reconstruction results than bicubic interpolation. Although the KHA reconstruction in Fig. 12.e is more noisy, it better preserves the overall tree structure when compared to the nearest neighbor reconstruction (Fig. 12.c). This difference can be attributed to the better generalization capability of KPCA which can be seen by applying the nearest neighbor method to the face image super-resolution problem (i.e., single patch application) (Fig. 13). In the simple nearest neighbor reconstruction which replaces the input with the nearest stored pattern based on the Euclidean distance in the input image space, three faces were erroneously reconstructed while the other reconstructions are far better than those of the KHA as they happen to be near to one of the stored patterns. The high-frequency restoration approach used for the natural images failed to capture any details. This mainly stems from the high dimensionality of the input images (60×60), since in this case the training patterns do not form a sufficiently dense sampling of the

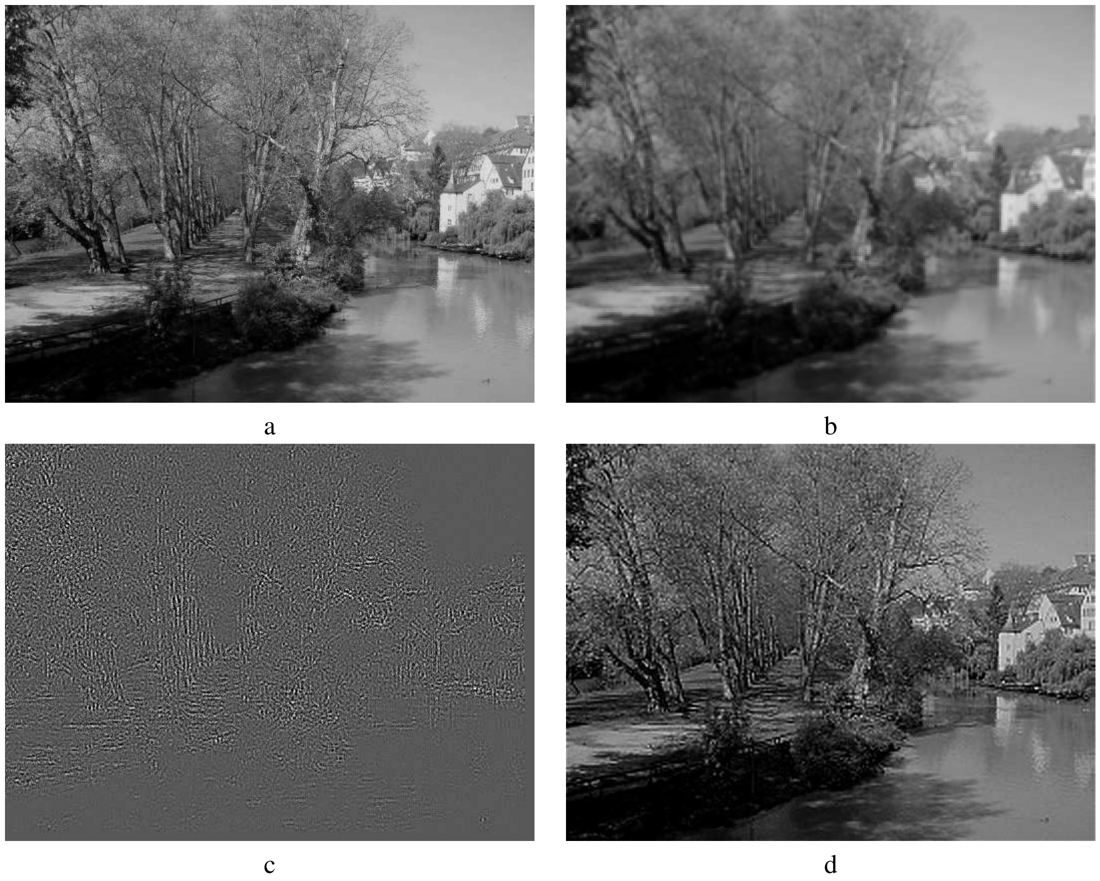
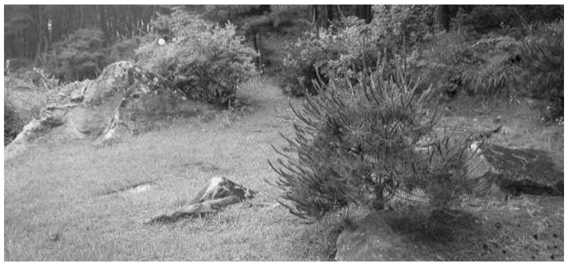


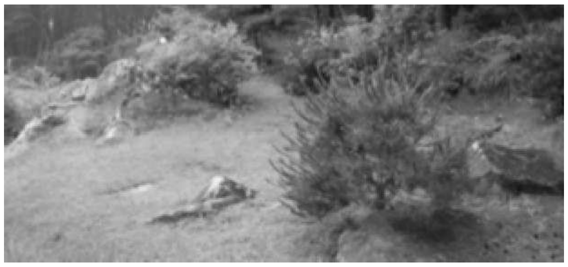
Figure 10: Example of natural image super-resolution: a. original image of resolution 396×528, b. low resolution image (198×264) stretched to the original scale, c. reconstruction of the high-frequency component (contrast enhanced for better visibility), and d. final KHA reconstruction.

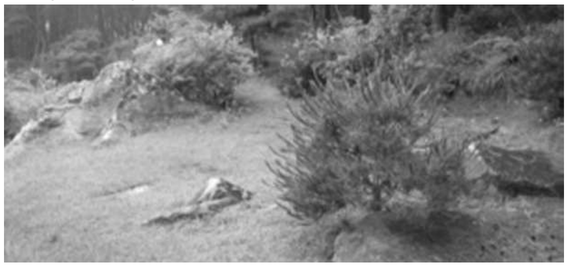
input space. As a result, high- and low-frequency components from different images are mixed together in the reconstruction. Overall, the results indicate a better generalization capability of KPCA as compared to the nearest neighbour classifier. This agrees with the finding of Freeman et al. [13] that, due to its limited generalization capability, the nearest neighbour method alone was not sufficient to obtain satisfactory super-resolution results.

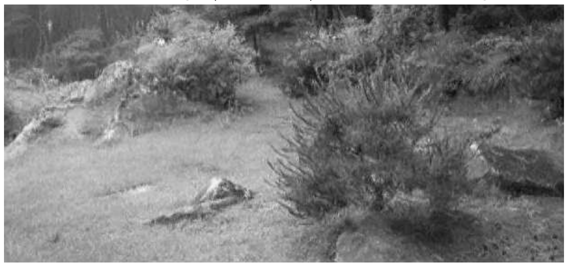
Multi-patch image denoising. Here, we adopt the second scenario of Sec. 3.2 where the noise was assumed to mainly contaminate the KPC subspace corresponding to smaller eigenvalues. As a consequence, training and test set are the same.

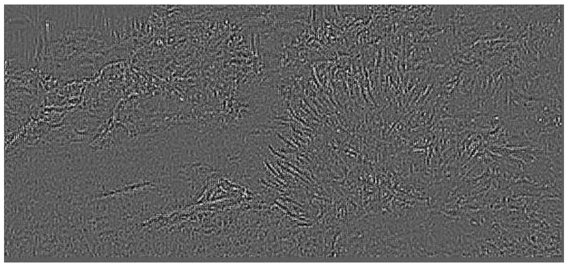
In the experiment, two different noisy images were constructed by adding white Gaussian noise (SNR 7.72dB) and salt and pepper noise (SNR 4.94dB) to the 256×256-sized Lena image [33]. From each image, 12×12 overlapping image patches were sampled at a regular interval of 2 pixels. The KPCA image model for the Gaussian kernel ($\sigma = 1$) was obtained by training the KHA on each training set with a learning rate η of 0.05 for around 800 sweeps through the dataset. The denoised images were then obtained by reconstructing the input image using the first r PCs from each KPCA model. For comparison, the median filter, Matlab's Wiener filter, and wavelet-based methods were applied. The parameters for the Wiener filter and wavelet-based methods were chosen in the same way as in the single-patch denoising experiments. Two state-of-the-art methods [9, 37]

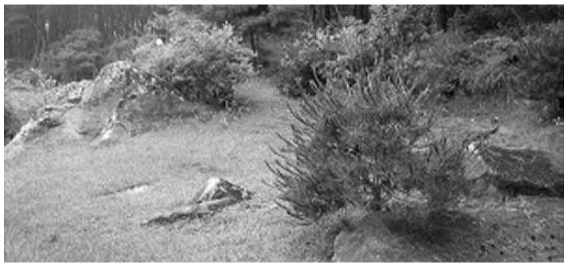

a. original image of resolution 284×618


b. low resolution image (142×309) stretched to the original scale


c. bicubic interpolation


d. supervised example-based learning based on nearest neighbor classifier


e. unsupervised KHA reconstruction of high-frequency component (contrast enhanced for better visibility)


f. unsupervised KHA reconstruction

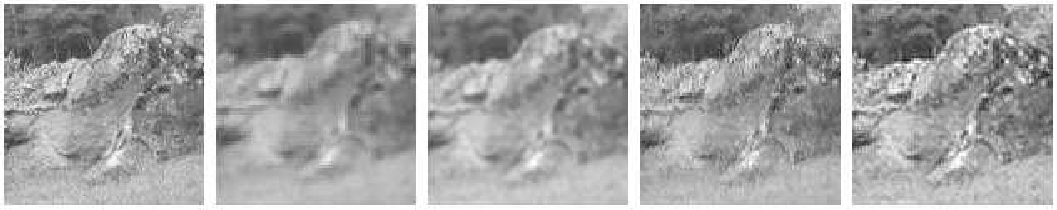
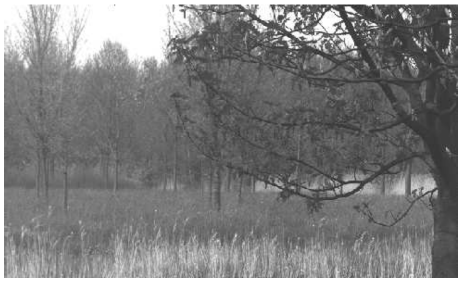
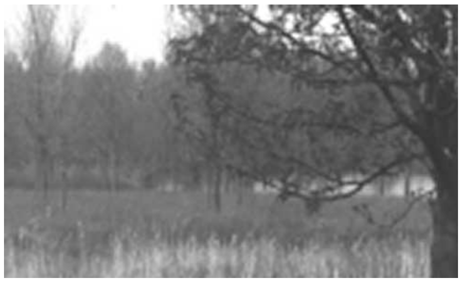
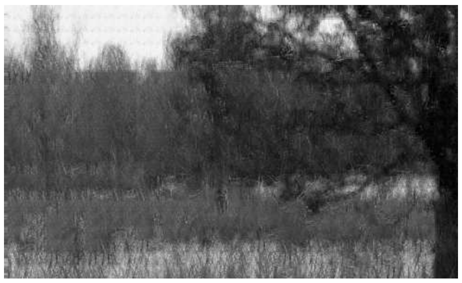
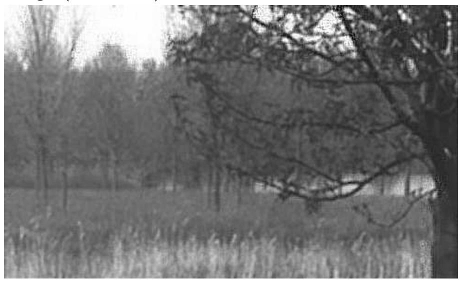

g. enlarged portions of a-d, and f (from left to right)

Figure 11: Comparison between different super-resolution methods.


a. original image of resolution 500×300


b. bicubic interpolation obtained from low resolution image (150×90)


c. nearest neighbor reconstruction


d. KHA reconstruction

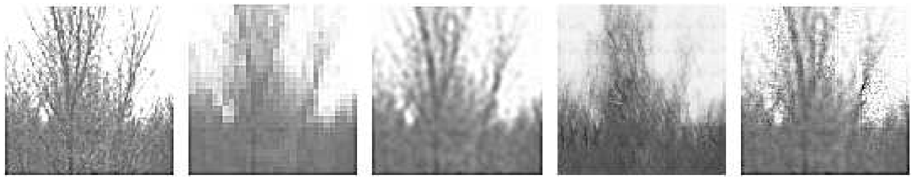

e. enlarged portions of a, low resolution image, b, c, and d (from left to right)

Figure 12: Image super-resolution without high-frequency decomposition.

Table 1: Performance of different denoising methods

Denoising method \ Noise type		Gaussian noise	Salt and pepper noise
Median filter		11.74dB	15.65dB
Matlab's wavelet-based methods		12.13dB	10.54dB
Matlab's Wiener filter		14.06dB	11.48dB
Choi and Baraniuk's method [9]	$d = 2.0$	14.14dB	11.91dB
	$d = 2.5$	13.84dB	11.48dB
	$d = 3.0$	13.20dB	10.94dB
Pižurica and Philips's method [37]	Symmlets2	15.54dB	7.85dB
	Symmlets4	15.42dB	7.93dB
	Symmlets8	15.19dB	7.96dB
	Daubechies wavelet2	15.54dB	7.85dB
	Daubechies wavelet4	15.11dB	8.00dB
	Daubechies wavelet8	14.84dB	8.00dB
PCA | KHA	$r = 10$	12.13dB | 12.42dB	11.44dB | 11.89dB
	$r = 20$	13.22dB | 14.09dB	11.76dB | 12.71dB
	$r = 30$	13.05dB | 14.25dB	11.05dB | 12.67dB
	$r = 40$	12.46dB | 14.27dB	10.15dB | 12.44dB
	$r = 50$	12.29dB | 14.11dB	9.31dB | 12.12dB
	$r = 60$	11.32dB | 13.88dB	8.97dB | 11.45dB

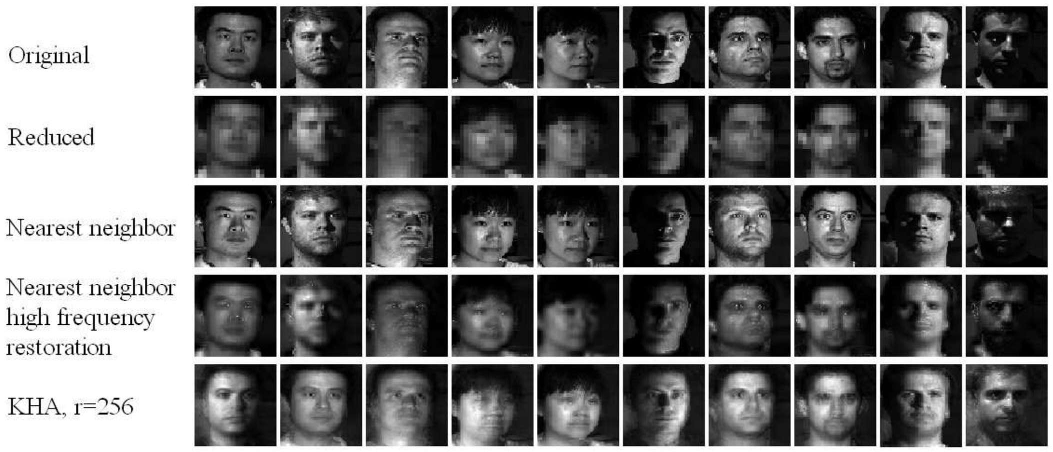


Figure 13: Face image super-resolution based on nearest neighbor methods. First and second rows: original and low-resolution input images (20×20), respectively. Third row: reconstructions obtained by replacing the KPCA with a nearest neighbor classifier in the single patch reconstruction experiments. Fourth row: reconstructions obtained by the multi-patch reconstruction method applied to the input image as a single patch. Fifth row: reconstructions obtained by the KPCA.

were also applied: Assuming Gaussian noise, Piẑurica and Philips [37] estimated the probability that a given coefficient in the wavelet subspace contains a noise-free component. For our simulation, different wavelet bases were utilized with Symmlets2, 4, and 8 and Daubechies wavelet2, 4, and 8. In [9], Choi and Baraniuk estimated the original signal by projecting the noisy signal into Besov spaces in the wavelet domain. For this, the factor d for estimating the Besov norm varied from 2.0 to 3.0 (as recommended in the Matlab code obtained from the author's website: http://www.dsp.rice.edu/$\sim$choi/) with an interval of 0.5. Table 1 summarizes the results. The superior performance of Piẑurica and Philips's method and the median filter for each type of noise are based on prior knowledge of the noise source. On the other hand, their reduced performance on the other noise type demonstrates the risk of relying on such an assumption. In contrast, the KHA performs well for both noise types indicating that, if nothing is known about the noise characteristics, the KHA can be considered as an acceptable alternative to existing methods. Figs. 14 and 15 show denoised images for all methods with the best parameter setting, respectively. Similar to linear PCA, KPCA would be expected to work best if the noise characteristics are Gaussian, however, not in the input domain but in the feature space associated with the kernel. The authors are not aware of any general insights into when this can be the case, although it has been observed that projections in feature spaces sometimes look more Gaussian than in the input space (e.g., p. 464 in [42]).

5 Conclusion

In this article, we proposed a generative image model based on a new method for iterative KPCA. In contrast to other patch-based modeling approaches, KPCA allows for nonlinear interactions between its basis images. Moreover, KPCA is capable of capturing part of the higher-order statistics which are particularly important for encoding image structure. To overcome the memory complexity of KPCA, the KHA was proposed as a method for the efficient estimation of kernel PCs. As a kernelization of the GHA, the KHA allows for computing KPCA without storing the kernel matrix, such that large datasets of high dimensionality can be processed. The presented

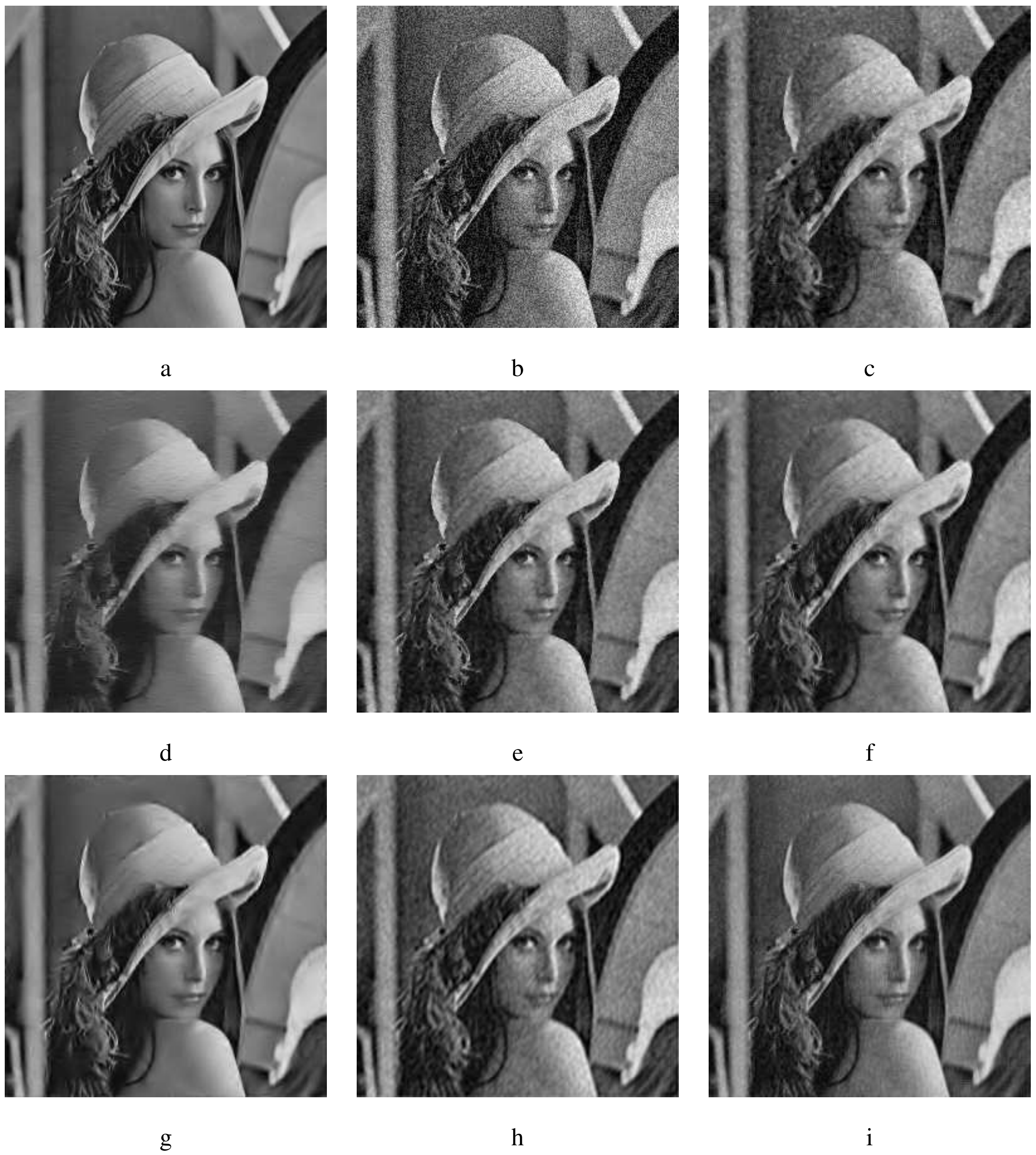


Figure 14: Denoising Gaussian noise: a. original image, b. input noisy image, c. median filter, d. Matlab's wavelet denoising, e. Matlab's Wiener filter, f. Choi and Baraniuk's method [9] ($d = 2.0$), g. Piẑurica and Philips's method [37] (Symmlets2), h. PCA ($r = 20$), and i. KHA ($r = 40$).

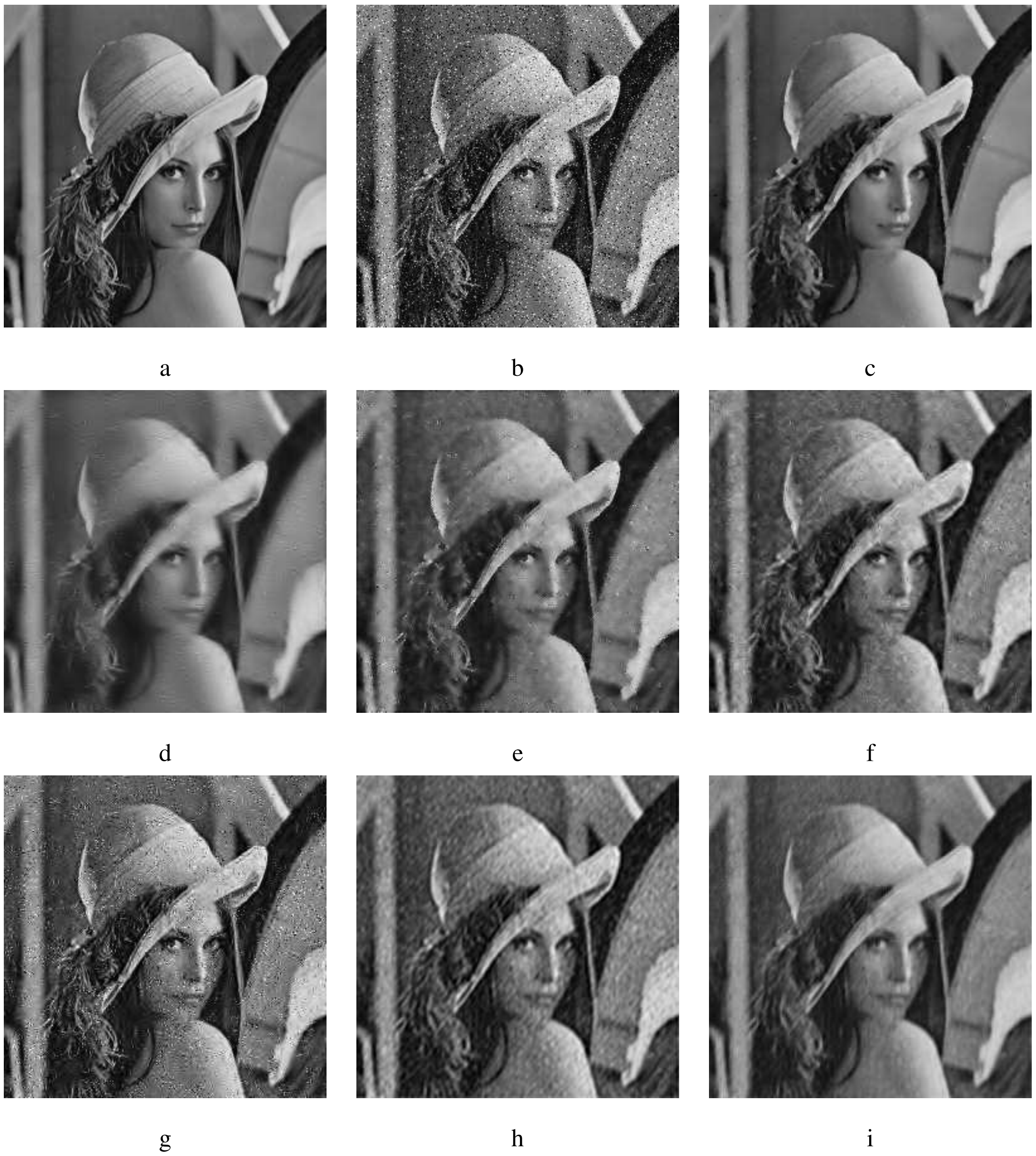


Figure 15: Denoising salt and pepper type noise: a. original image, b. input noisy image, c. median filter, d. Matlab's wavelet denoising, e. Matlab's Wiener filter, f. Choi and Baraniuk's method [9] ($d = 2.0$), g. Piẑurica and Philips's method [37] (Daubechies wavelet8), h. PCA ($r = 20$), and i. KHA ($r = 20$).

super-resolution and denoising experiments show that the generic image models obtained from the KHA lead to a comparable performance to existing computer vision and image processing approaches that are specifically tailored to these tasks.

Compared to existing super-resolution and denoising methods, the experimental results obtained using the KHA are promising. In terms of reconstruction quality, however, it is difficult to compare our approach with the previous ones in [23] and [13] since this largely depends on subjective assessment, not on objective quantities such as the MSE (which is - as the results on PCA indicate - a poor quality measure for images). The main difference lies in the applied learning method: the methods proposed by Hertzmann et al. [23] and Freeman et al. [13] are based on *supervised* learning. In machine learning, it is generally believed that when feasible, a supervised learning approach often leads to the best results. However, at the same time, supervised algorithms can have shortcomings in that the data may be more expensive to obtain (since they require inputs *and* outputs), and the solution can be less flexible in that it is only useful for the exact task considered. This means that once trained on a training set containing labeled data, the above methods can only be used for the one image super-resolution task it was trained for. In contrast, our image model, which is only trained on high-resolution images, can be directly applied to a variety of image restoration tasks, including denoising and image super-resolution using inputs of various resolutions, without retraining.

There are various directions for further work. The KHA, as a general iterative algorithm of KPCA applicable to large datasets, can significantly enlarge the application area of KPCA, which as a generic machine learning technique also enjoys some popularity in other fields including speech processing [27], character recognition [42], object shape modeling [47], etc. With respect to image modeling, the best choice of the kernel remains elusive. We still do not know which higher-order statistics are important for coding image content. It is also unclear to what extent the different available kernels are capable of modeling the occlusion and superposition phenomena that contribute to the generation of an image. A further investigation of these questions could lead to the design of new kernels that specifically incorporate the generation principles of natural images. Finally, our current multi-patch applications do not explicitly enforce spatial consistency between patches except by averaging in overlapping regions. Similar to the approache of [13], the explicit use of spatial context in the KPCA reconstruction process might improve its modeling capabilities considerably.

Acknowledgments

The ideas presented in this study have greatly profited from discussions with C. Szepesvari, A. Gretton, M. Hein, and G. Bakır. We are grateful to Y. Kwon for helping us with the experiments, and to the anonymous reviewers whose comments helped to improve the quality of this manuscript.

References

[1] S. Akkarakaran and P. P. Vaidyanathan. The role of principal component filter banks in noise reduction. In *Wavelet Applications in Signal and Image Processing VII*, pages 346–357, vol. 3813. SPIE, The International Society for Optical Engineering, 1999.

[2] R. J. Baddeley and P. J. B. Hancock. A statistical analysis of natural images predicts psychophysically derived orientation tuning curves. *Proc. R. Soc. B*, 246(1317):219 – 223, 1991.

[3] S. Baker and T. Kanade. Limits on super-resolution and how to break them. *IEEE Trans. Pattern Analalysis and Machine Intelligence*, 24(9):1167–1183, 2002.

[4] A. J. Bell and T. J. Sejnowski. The 'independent components' of natural scenes are edge filters. *Vision Research*, 37:3327–3338, 1997.

[5] R. W. Buccigrossi and E. P. Simoncelli. Image compression via joint statistical characterization in the wavelet domain. *IEEE Trans. Image. Proc.*, 8(12):1688 – 1701, 1999.

[6] C. J. C. Burges. Simplified support vector decision rules. In L. Saitta, editor, *Proceedings of the 13th International Conference on Machine Learning*, pages 71–77, San Mateo, CA, 1996. Morgan Kaufmann.

[7] O. Chapelle and V. Vapnik. Model selection for support vector machines. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*. MIT Press, 2000.

[8] T. Chen, Y. Hua, and W-Y. Yan. Global convergence of oja's subspace algorithm for principal component extraction. *IEEE Trans. Neural Networks*, 9(1):58–67, 1998.

[9] H. Choi and R. G. Baraniuk. Multiple basis wavelet denoising using besov projections. In *Proceedings of IEEE International Conference on Image Processing*, pages 595–599, 1999.

[10] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel-target alignment. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[11] D. J. Field. What is the goal of sensory coding? *Neural Computation*, 6:559–601, 1994.

[12] M.O. Franz and B. Schölkopf. Implicit estimation of Wiener series. In *Proc. IEEE MLSP 2004*, 2004 (in press).

[13] W. T. Freeman, T. R. Jones, and E. C. Pasztor. Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22(2):56–65, 2002.

[14] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *Intl. J. Comp. Vis.*, 40(1):25 – 47, 2000.

[15] 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 Analalysis and Machine Intelligence*, 23(6):643–660, 2001.

[16] F. Girosi. An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10(6):1455–1480, 1998.

[17] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison Wesley, Massachusetts, 1992.

[18] U. Grenander and A. Srivastava. Probability models for clutter in natural images. *IEEE Trans. PAMI*, 23(4):424 – 429, 2001.

[19] J. Ham, D. D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. Technical Report 110, Max-Planck-Insitut für biologische Kybernetik, Tübingen, July 2003.

[20] P. J. B. Hangcock, R. J.Baddely, and L. S Smith. The principal components of natural images. *Network*, 3:61 – 70, 1992.

[21] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, New Jersey, 2nd edition, 1999.

[22] R. Herbrich. *Learning Kernel Classifiers: Theory and Algorithms*. MIT Press, Cambridge, MA, 2001.

[23] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Computer Graphics (Proc. Siggraph 2001)*, pages 327–340, NY, 2001. ACM Press.

[24] J. Hurri, A. Hyvärinen, J. Karhunen, and E. Oja. Image feature extraction using independent component analysis. In *Proc. IEEE 1996 Nordic Conference of Signal Processing (NORSIG'96)*, 1996.

[25] R. Keys. Cubic convolution interpolation for digital image processing. *IEEE Trans. Acoustics, Speech, Signal Processing*, 29(6):1153–1160, 1981.

[26] K. I. Kim, M. O. Franz, and B. Schölkopf. Kernel Hebbian algorithm for iterative kernel principal component analysis. Technical Report 109, Max-Planck-Insitut für biologische Kybernetik, Tübingen, June 2003.

[27] J. T. Kwok, B. Mak, and S. Ho. Eigenvoice speaker adaptation via composite kernel principal component analysis. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, Cambridge, MA, To appear 2004. MIT Press.

[28] J. T. Kwok and I. W. Tsang. Finding the pre-images in kernel principal component analysis. *6th Annual Workshop On Kernel Machines*, Whistler, Canada, 2002, poster available at http://www.cs.ust.hk/~jamesk/kernels.html.

[29] A. B. Lee, D. Mumford, and J. Huang. Occlusion models for natural images: A statistical study of a scale-invariant dead leaves model. *Intl. J. Comp. Vis.*, 41(1/2):35 – 59, 2001.

[30] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *Intl. J. Comp. Vis.*, 43(1):7 – 27, 2001.

[31] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, editors, *Neural Networks for Signal Processing IX*, pages 41–48. IEEE, 1999.

[32] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and de-noising in feature spaces. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 536–542, Cambridge, MA, 1999. MIT Press.

[33] D. C. Munson. A note on lena. *IEEE Trans. Image Processing*, 5(1), 1996.

[34] E. Oja. A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273, 1982.

[35] E. Oja. Principal components, minor components, and linear neural networks. *Neural Networks*, 5:927–935, 1992.

[36] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

[37] A. Piẑurica and W. Philips. Estimating probability of presence of a signal of interest in multiresolution single- and multiband image denoising. *submitted to IEEE Trans. Image Processing*, 2004.

[38] T. Poggio and F. Girosi. Extension of a theory of networks for approximation and learning: dimensionality reduction and clustering. In *Proceedings Image Understanding Workshop*, pages 597–603, Pittsburgh, Pennsylvania, 1990. Morgan Kaufmann.

[39] S. Romdhani, S. Gong, and A. Psarrou. A multiview nonlinear active shape model using kernel PCA. In *Proceedings of BMVC*, pages 483–492, Nottingham, UK, 1999.

[40] D. L. Ruderman. Origins of scaling in natural images. *Vis. Res.*, 37(23):3385 – 3395, 1997.

[41] T. D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural netowork. *Neural Networks*, 12:459–473, 1989.

[42] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[43] B. Schölkopf, A. Smola, and K. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[44] E. P. Simoncelli. Bayesian denoising of visual images in the wavelet domain. In P. Müller and B. Vidakovic, editors, *Bayesian inference in wavelet based models*, pages 291 – 308. Springer, New York, 1999.

[45] I. Steinwart. On the generalization ability of support vector machines. Technical report, University of Jena, 2001.

[46] A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–591, 1991.

[47] C. J. Twining and C. J. Taylor. Kernel principal component analysis and the construction of non-linear active shape models. In *Proceedings of British Machine Vision Conference*, pages 23–32, 2001.

[48] V. Vapnik and O. Chapelle. Bounds on error expectation for SVM. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 261–280, Cambridge, MA, 2000. MIT Press.

[49] U. von Luxburg, O. Bousquet, and B. Schlkopf. A compression approach to support vector model selection. *The Journal of Machine Learning Research*, 5:293–323, 2004.

[50] S. Yoshizawa, U. Helmke, and K. Starkov. Convergence analysis for principal component flows. *Int. J. Appl. Math. Comput. Sci.*, 11(1):223–236, 2001.

[51] S. Zhu and D. Mumford. Prior learning and Gibbs reaction-diffusion. *IEEE Trans. PAMI*, 19(11):1236 – 1250, 1997.