A. Leonardis, H. Bischof (Eds):
Statistical Learning in Computer Vision, ECCV 2004 Workshop, Prague, Czech Republic, May 2004, pp. 135-149.

Kernel Hebbian Algorithm for Single-Frame Super-Resolution

Kwang In Kim[1], Matthias O. Franz[1], and Bernhard Schölkopf[1]

Max Planck Institute für biologische Kybernetik
Spemannstr. 38, D-72076 Tübingen, Germany
{kimki, mof, bs}@tuebingen.mpg.de
http://www.kyb.tuebingen.mpg.de/

Abstract. This paper presents a method for single-frame image super-resolution using an *unsupervised learning technique*. The required prior knowledge about the high-resolution images is obtained from *Kernel Principal Component Analysis* (KPCA). 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*. By kernelizing the Generalized Hebbian Algorithm, one can iteratively estimate the Kernel Principal Components with only linear order memory complexity. The resulting super-resolution algorithm shows a comparable performance to the existing supervised methods on images containing faces and natural scenes.

1 Introduction

The problem of image super-resolution, where one achieves high-resolution enlargements of pixel-based images [1], has a lot of potential applications in graphics, image processing, and computer vision. Methods of image super-resolution can be conveniently divided into three complimentary classes [1]: 1) *interpolation and sharpening* enlarges the low resolution image using generic image interpolation techniques and sharpen the resulting image for better visibility; 2) *aggregation from multiple frames* extracts a single high-resolution frame from a sequence of low-resolution images; 3) *single-frame super-resolution* extracts high-resolution image details from a single low-resolution image, which cannot be achieved by simple sharpening. All three approaches rely on a certain type of prior knowledge about the image class to be reconstructed. The third approach, in particular, needs a specific characterization of its respective image class which is often available in the form of example patterns.

Whereas the first two methods have already been extensively studied [2], [3], the third method has been introduced only recently. Hertzmann, et. al [4] proposed a patch-wise reconstruction technique. 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 as reconstruction. In [1], Freeman et. al. presented a similar technique to extract high-frequency details from a low-resolution image. Both algorithms already demonstrated impressive super-resolution results.

Here, we propose an alternative approach to super-resolution based on an *unsupervised* learning technique. Instead of encoding a fixed relationship between pairs of high- and low-resolution image patches, we rely on a generic model of the high-resolution images that is obtained from *Kernel Principal Component Analysis* (KPCA) [5]. In contrast to linear Principal Component Analysis (PCA) which has already been widely used in image analysis, KPCA is capable of capturing part of the higher-order statistics which are particularly important for encoding image structure [6]. Recent success in related applications [7] further motivates the use of KPCA for image super-resolution. Capturing these higher-order statistics, however, 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 [8][9]. The resulting algorithm estimates the kernel principal components with linear order memory complexity, making it applicable to large problems.

This paper is organized as follows. The remainder of this section briefly introduces PCA, GHA, and KPCA. Section 2 present the proposed image super-

resolution method while Section 3 and formulates the KHA. Experimental results are presented in Section 4 and conclusions are drawn in Section 5.

Principal component analysis. Given a set of centered observations $\mathbf{x}_k = \mathbb{R}^N$, $k = 1, \ldots, l$, and $\sum_{k=1}^{l} \mathbf{x}_k = 0$, PCA diagonalizes the covariance matrix[1] $\overline{C} = \frac{1}{l} \sum_{j=1}^{l} \mathbf{x}_j \mathbf{x}_j{}^\top$. This is readily performed by solving the eigenvalue equation $\lambda \mathbf{v} = \overline{C}\mathbf{v}$ for eigenvalues $\lambda \geq 0$ and eigenvectors $\mathbf{v}_i \in \mathbb{R}^N \setminus 0$.

Generalized Hebbian algorithm. From a computational point of view, 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. Among the existing iterative methods for PCA, the generalized Hebbian algorithm (GHA) is of particular interest, since it does not only provide a memory-efficient implementation but also has the inherent capability to adapt to time-varying distributions.

Let us define a matrix $\mathbf{W}(t) = (\mathbf{w}_1(t)^\top, \ldots, \mathbf{w}_r(t)^\top)^\top$, where r is the number of eigenvectors considered and $\mathbf{w}_i(t) \in \mathbb{R}^N$. Given a random initialization of $\mathbf{W}(0)$, the GHA applies the following recursive rule

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

where $\mathbf{x}(t)$ is a randomly selected pattern from l input examples, presented at time t, $\mathbf{y}(t) = \mathbf{W}(t)\mathbf{x}(t)$, and $\mathrm{LT}[\cdot]$ sets all elements above the diagonal of its matrix argument to zero, thereby making it lower triangular. It was shown in [8] for $i = 1$ and in [9] for $i > 1$ that $\mathbf{W}(t) \to \mathbf{V}_i$ as $t \to \infty$.[2] For a detailed discussion of the GHA, readers are referred to [9].

Kernel principal component analysis. When the data of interest are highly nonlinear, linear PCA fails to capture the underlying structure. As a nonlinear extension of PCA, KPCA computes the principal components (PCs) in a possibly high-dimensional *Reproducing Kernel Hilbert Space* (RKHS) F which is related to the input space by a nonlinear map $\Phi : \mathbb{R}^N \to F$ [10]. An important property of a RKHS is that the inner product of two points mapped by Φ can be evaluated using *kernel functions*

$$k(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}), \tag{2}$$

which allows us to compute the value of the inner product without having to carry out the map Φ explicitly. Since PCA can be formulated in terms of inner products, we can compute it also implicitly in a RKHS. Assuming that the data are centered in F (i.e., $\sum_{k=1}^{l} \Phi(\mathbf{x}_k) = 0$)[3] the covariance matrix takes the form

$$C = \frac{1}{l} \mathbf{\Phi}^\top \mathbf{\Phi}, \tag{3}$$

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

[2] Originally it has been shown that $\mathbf{w}_i$ converges to the i-th eigenvector of $E[\mathbf{x}\mathbf{x}^\top]$, given an infinite sequence of examples. By replacing each $\mathbf{x}(t)$ with a random selection $\mathbf{x}_i$ from a finite training set, we obtain the above statement.

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

where $\mathbf{\Phi} = \left(\Phi(\mathbf{x}_1)^\top, \ldots, \Phi(\mathbf{x}_l)^\top\right)^\top$. We now have to find the eigenvalues $\lambda \geq 0$ and eigenvectors $\mathbf{v} \in F \setminus 0$ satisfying

$$\lambda \mathbf{v} = C\mathbf{v}. \tag{4}$$

Since all solutions $\mathbf{v}$ with $\lambda \neq 0$ lie within the span of $\{\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_l)\}$ [5], we may consider the following equivalent problem

$$\lambda \mathbf{\Phi v} = \mathbf{\Phi} C\mathbf{v}, \tag{5}$$

and we may represent $\mathbf{v}$ in terms of an l-dimensional vector $\mathbf{q}$ as $\mathbf{v} = \mathbf{\Phi}^\top \mathbf{q}$. Combining this with (3) and (5) and defining an $l \times l$ kernel matrix $\mathbf{K}$ by $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^\top$ leads to $l\lambda \mathbf{Kq} = \mathbf{K}^2\mathbf{q}$. The solution can be obtained by solving the *kernel eigenvalue problem* [5]

$$l\lambda \mathbf{q} = \mathbf{Kq}. \tag{6}$$

It should be noted that the size of the kernel matrix scales with the square of the number of examples. Thus, it becomes computationally infeasible to solve directly the kernel eigenvalue problem for large number of examples. This motivates the introduction of the Kernel Hebbian Algorithm presented in the next section. For more detailed discussions on PCA and KPCA, including the issue of computational complexity, readers are referred to [10].

2 Image Super-Resolution

Single-patch super-resolution. To reconstruct a super-resolution patch (or small image) from a low-resolution image patch which was *not* contained in the training set, we first scale up the image to the same size as the training images by nearest neighbour interpolation, then map the image (call it $\mathbf{x}$) into the RKHS F using Φ, and project it onto the KPCA subspace spanned by a limited number of KPCs to get $P\Phi(\mathbf{x})$. Via the projection P, the image is mapped to an image which is consistent with the statistics of the high-resolution training images. However, at that point, the projection still lives in F, which can be infinite-dimensional. We thus need to find its *preimage*, i.e., the corresponding point in $\mathbb{R}^N$. To find it, we minimize $\|P\Phi(\mathbf{x}) - \Phi(\mathbf{z})\|^2$ over $\mathbf{z} \in \mathbb{R}^N$. Note that this objective function can be computed in terms of inner products and thus in terms of the kernel (2). For the minimization, we use gradient descent [11] with starting points obtained using the method of [12].

Multi-patch image reconstruction. For the super-resolution of a rather large image, we adopt the method of [1], 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. use a nearest neighbour 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 [1] and then fed to KPCA.

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 based on the KPCA in the same way as in single patch super-resolution. The resulting image containing only high-frequency components is then superimposed on the bicubic interpolation to give the final reconstruction.

3 Kernel Hebbian Algorithm

As noted in the introduction, the advantage of the GHA over conventional PCA is that it does not require the storage of the covariance matrix. This is particularly useful when the size of covariance matrix is large. Similarly, the direct formulation of KPCA becomes impractical for large sample sizes since the associated kernel matrix is too large to be stored in memory. For this case, we reformulate the GHA in a RKHS to obtain a memory-efficient approximation of KPCA.

The GHA update rule of Eq. (1) is represented in the RKHS F as

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \eta(t)\left(\mathbf{y}(t)\Phi(\mathbf{x}(t))^\top - \mathrm{LT}[\mathbf{y}(t)\mathbf{y}(t)^\top]\mathbf{W}(t)\right), \tag{7}$$

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 $\Phi(\mathbf{x}(t)) = \Phi(\mathbf{x}_i)$. 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

$$\mathbf{W}(t) = \mathbf{A}(t)\mathbf{\Phi} \tag{8}$$

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

$$\mathbf{A}(t+1)\mathbf{\Phi} = \mathbf{A}(t)\mathbf{\Phi} + \eta(t)\left(\mathbf{y}(t)\Phi(\mathbf{x}(t))^\top - \mathrm{LT}[\mathbf{y}(t)\mathbf{y}(t)^\top]\mathbf{A}(t)\mathbf{\Phi}\right). \tag{9}$$

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

$$\mathbf{A}(t+1) = \mathbf{A}(t) + \eta(t)\left(\mathbf{y}(t)\mathbf{b}(t)^\top - \mathrm{LT}[\mathbf{y}(t)\mathbf{y}(t)^\top]\mathbf{A}(t)\right), \tag{10}$$

or equivalently in component-wise form

$$a_{ij}(t+1) = \begin{cases} a_{ij}(t) + \eta y_i(t) - \eta y_i(t) \sum_{k=1}^{i} a_{kj}(t) y_k(t) & \text{if} \quad J(t) = j \\ a_{ij}(t) - \eta y_i(t) \sum_{k=1}^{i} a_{kj}(t) y_k(t) & \text{otherwise,} \end{cases} \tag{11}$$

where

$$y_i(t) = \sum_{k=1}^{l} a_{ik}(t) \Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}(t)) = \sum_{k=1}^{l} a_{ik}(t) k(\mathbf{x}_k, \mathbf{x}(t)), \tag{12}$$

which does not require $\Phi(\mathbf{x})$ in explicit form and accordingly provides a practical implementation of the GHA in F.

It can be shown that the proposed algorithm (7) (and equivalently (10)) will converge, and that $\mathbf{W}$ will approach the matrix whose rows are the first r eigenvectors of the covariance matrix C, ordered by decreasing eigenvalue, provided that k is continuous with respect to both of its arguments, and $\mathbf{A}$ is randomly initialized [13].

During the derivation of (10), 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 $\widetilde{\Phi}(\mathbf{x}(t)) \doteq \Phi(\mathbf{x}(t)) - \overline{\Phi}(\mathbf{x})$, 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 (11) except that Eq. (12) has to be replaced by the more complicated expression

$$y_i(t) = \sum_{k=1}^{l} a_{ik}(t)(k(\mathbf{x}(t), \mathbf{x}_k) - \bar{k}(\mathbf{x}_k)) - \overline{a}_i(t) \sum_{k=1}^{l} (k(\mathbf{x}(t), \mathbf{x}_k) - \bar{k}(\mathbf{x}_k)). \tag{13}$$

with $\bar{k}(\mathbf{x}_k) = \frac{1}{l} \sum_{m=1}^{l} k(\mathbf{x}_m, \mathbf{x}_k)$ and $\overline{a}_i(t) = \frac{1}{l} \sum_{m=1}^{l} a_{im}(t)$. This is directly applicable in the batch setting where all the samples are 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. It should be noted that not only in training but also in testing, each pattern should be centered, using the training mean.

The time and memory complexity for each iteration of KHA is $\mathbf{O}(r \times l \times N)$ and $\mathbf{O}(r \times l + l \times N)$, respectively, where r, l, and N are the number of PCs to be computed, the number of examples, and the dimensionality of input space, respectively.[4] This rather high time complexity can be lowered by precomputing and storing the whole or part of the kernel matrix. When we store the entire kernel matrix, as KPCA does, the time complexity reduces to $\mathbf{O}(r \times l)$.

4 Experimental Results

4.1 Single-patch Case: Super-resolution of Face Images.

In order to investigate the single-patch super-resolution capabilities of KPCA, we performed face image super-resolution where the each single face image is

[4] $\bar{k}(\mathbf{x}_k)$ and $\overline{a}_i(t)$ in (13) for each $k, i = 1, \ldots, l$ are calculated only once at the beginning of each iteration.

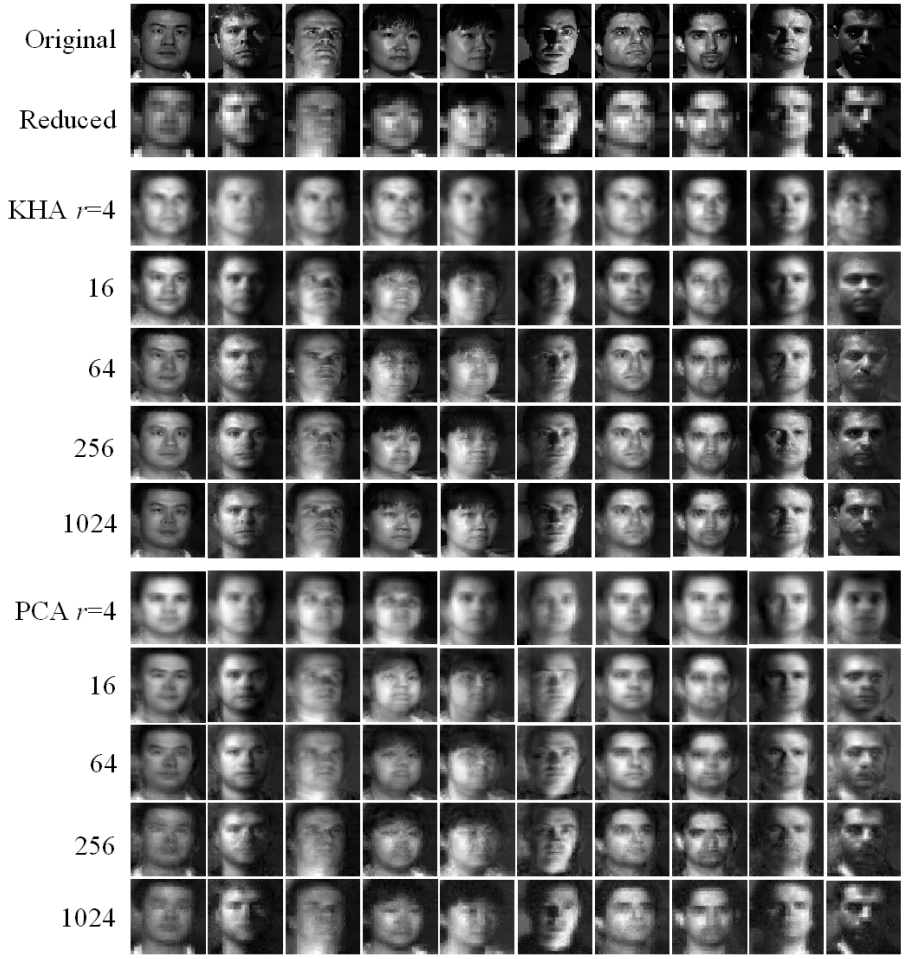

Fig. 1. Face reconstruction based on PCA and KHA using a Gaussian kernel with $\sigma = 1$ for varying numbers of PCs

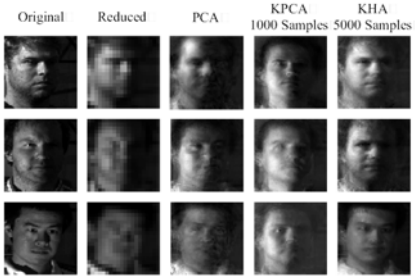

Fig. 2. Face reconstruction examples obtained from KPCA and KHA trained on 1,000 and 5,000 examples, respectively. Occasional erroneous reconstruction of images indicates that KPCA requires a large amount of data to properly sample the underlying structure

regarded as a single patch. The Yale Face Database B contains 5,760 images of 10 persons [14]. 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). For training, (60×60)-sized face images were fed into the KHA with a Gaussian kernel $(k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2/(2\sigma^2)))$. The learning rate η was fixed at 0.05 during training. Then, the test images were blurred and subsampled to a 20×20 grid and scaled up to the original resolution (60×60), 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 $([-0.05, 0.05])$. Figure 1 shows reconstruction examples obtained using different numbers of components. 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 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 by [15].

Interestingly, when the number of examples is small and the sampling of this manifold is sparse, this can have the consequence that the KPCA (or KHA) reconstruction looks like the face of another person than the one used to generate the test image. In a certain sense, this means that the errors performed by KPCA remain *within* the manifold of faces. Figure 2 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 shows, some of the misreconstructions that are made

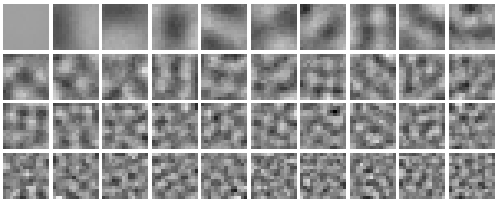

Fig. 3. The first 40 kernel PCs of 40,000 (14×14)-sized patches of natural images obtained from the KHA using a Gaussian kernel with $\sigma = 40$

by KPCA due to the lack of training examples were corrected by the KHA using a larger training set.

4.2 Super-resolution of natural images.

To examine the applicability of the KHA to larger scale problems, we trained our algorithm on more than 40,000 patterns obtained from natural images. Figure 3 shows the first 40 KPCs of the 12×12-sized image patches obtained from the KHA using a Gaussian kernel. The plausibility of the obtained PCs can be demonstrated by increasing the size of the Gaussian kernel such that the distance metric of the corresponding RKHS becomes more and more similar to that of the input space [10]. As can be seen in Fig. 3, the KPCs approach those of linear PCA [9] as expected.

Fig. 4. Training images of size 396×528. The training patterns are obtained by sampling 2,500 points at random from each image

For the multi-patch super-resolution, the KHA was trained on a set of 10,000 (12×12)-sized image patches obtained from the images in Fig. 4. This time, 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. 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 slightly overlap into their neighbours 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. 5 shows the super-resolution result. The final reconstruction was post-processed using high-boost filtering [16] to enhance the edges that become slightly blurred since only the first 200 KPCAs 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 [1] where the spatial relationship between patches is modeled based on Markov random fields.

Fig. 6 shows another super-resolution result. The low resolution image is obtained in the same way as in Fig. 5. For comparison, bicubic interpolation and the nearest neighbor technique also have been applied. Again, for all the methods the final reconstructions are high-boost filtered. In comparison to the image stretching (Fig. 6.b), bicubic interpolation (Fig. 6.c) produces far better results. However simple edge enhancement without any priori knowledge failed to completely remove the blurring effect. The two learning-based methods show better capability in recovering the complex local structure.

As shown in Fig. 6 the KHA and the nearest neighbor-based method showed comparable performances. However, the insight on the difference between the modeling capabilities of two methods can be gained by enlarging the size of reconstruction patch and performing super-resolution directly on raw images. For this, a new KHA image model was trained on raw image patches without high-pass filtering. Then, during the super-resolution, the input image was decomposed into (16×16) patches and reconstructed independently based on KHA. For comparison, the nearest neighbor-based model was trained in the same way. As highlighted in Fig. 7.e, two learning-based approaches again show better reconstruction results then bicubic reconstruction. Although the KHA reconstruction shown in Fig. 7.e is noisier, it preserves the overall tree structure which is lost by the nearest neighbor reconstruction (Fig. 6.d).

5 Discussion

This paper proposed an algorithm for running kernel PCA on large databases, leading to an unsupervised learning approach for image super-resolution. KPCA

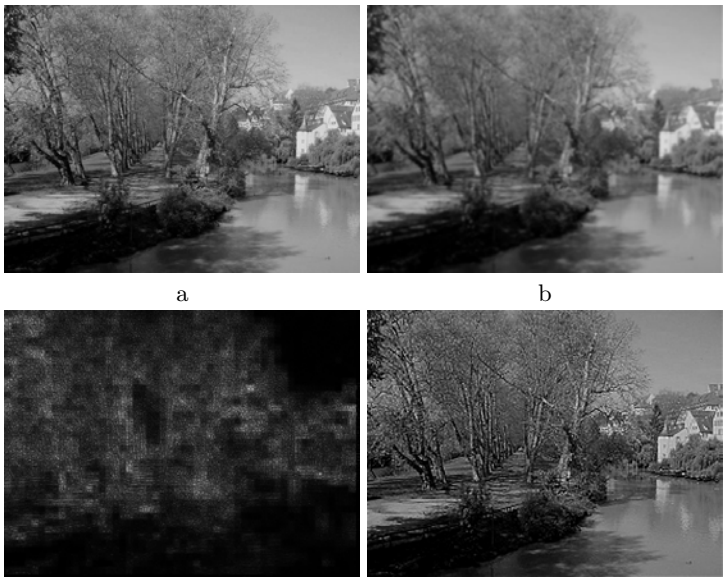

Fig. 5. Example of natural image super-resolution: a. original image of resolution 396×528, b. low resolution image (264×198) stretched to the original scale, c. reconstruction of the high-frequency component (contrast enhanced for better visibility), and d. final KHA reconstruction

Fig. 6. Comparison between different super-resolution methods: a. original image of resolution 396×528, b. low resolution image (264×198) stretched to the original scale, c. bicubic interpolation, d. example-based learning based on nearest neighbor classifier, e. KHA reconstruction of high-frequency component (contrast enhanced for better visibility), and f. KHA reconstruction, g. enlarged portions of b, c, f, and d (from left to right)

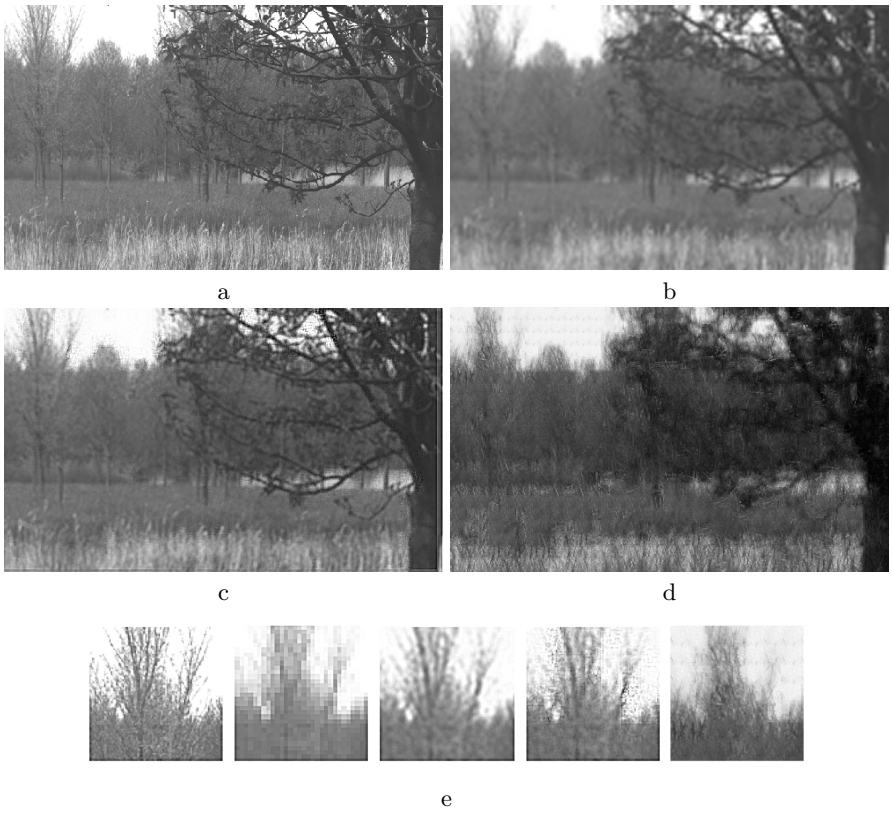

Fig. 7. Image super-resolution without high-frequency decomposition: a. original image of resolution 500 × 300, b. bicubic interpolation obtained from low resolution image (150 × 90), c. KHA reconstruction, d. nearest neighbor reconstruction, e. enlarged portions of a, low resolution image, b, c, and d (from left to right)

was used to learn the statistics contained in a set of the training patterns. By mapping the input image into an RKHS and kernelizing the problem, we capture part of the higher-order statistics which are particularly important for encoding image structure. To overcome the memory complexity of KPCA, KHA was proposed as a method for the efficient estimation of the PCs in an RKHS. As a kernelization of the GHA, the KHA allows for performing KPCA without storing the kernel matrix, such that large datasets of high dimensionality can be processed. This property makes the KHA particularly suitable for applications in statistical image processing applications, where large training sets are required to capture the statistics of an image class with sufficient accuracy.

Compared to existing methods, the experimental results obtained using our approach are promising. It should be stressed, however, that statistical super-resolution methods can be compared in several important respects, out of which accuracy is only one. For instance, the methods proposed by Hertzmann et . al. [4] and Freeman et. al. [1] are based on *supervised* learning, i.e., the training data are given as input-output pairs of low- and high-resolution images. 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, and the solution can be less flexible in that it is strictly only useful for the exact task considered. In our case, this means that once trained on a training set containing labeled data, the above methods can strictly speaking only be used for the one image super-resolution task it was trained for. In contrast, our unsupervised method,[5] which is only trained on high-resolution images, can directly be applied to a variety of image restoration tasks, including e.g. denoising, or image super-resolution using inputs of various resolutions, without retraining. We plan to experimentally corroborate this in future work.

These kinds of differences should be kept in mind when assessing the performance of the proposed method. We do not claim to have found the ultimate solution to the problem of image super-resolution—we even did not try to remove the blocking artifact to combine the patch-wise reconstruction, except boundary smoothing. The contribution of this paper lies in developing a specific nonlinear unsupervised machine learning technique that can be run on large databases and demonstrating its potential on image super-resolution.

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 is a generic machine learning technique that also enjoys some popularity in computer vision [10], [17]. For the image super-resolution, we plan to also explore supervised estimation methods in RKHS's (e.g., [18]), to study in more detail the benefits and shortcomings of a more task-specific solution to the problem.

[5] Our method is unsupervised in the sense that it is not given input-output pairs for training, but outputs only.

Acknowledgments The ideas presented in this study have greatly profited from discussions with A. Gretton, M. Hein, and G. Bakir.

References

1. Freeman, W.T., Jones, T.R., Pasztor, E.C.: Example-based super-resolution. IEEE Computer Graphics and Applications **22** (2002) 56–65
2. Keys, R.: Cubic convolution interpolation for digital image processing. IEEE Trans. Acoustics, Speech, Signal Processing **29** (1981) 1153–1160
3. Baker, S., Kanade, T.: Limits on super-resolution and how to break them. IEEE Trans. Pattern Analalysis and Machine Intelligence **24** (2002) 1167–1183
4. Hertzmann, A., Jacobs, C.E., Oliver, N., Curless, B., Salesin, D.H.: Image analogies. Computer Graphics (Proc. Siggraph 2001), NY, ACM Press (2001) 327–340
5. Schölkopf, B., Smola, A., Müller, K.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation **10** (1998) 1299–1319
6. Field, D.J.: What is the goal of sensory coding? Neural Computation **6** (1994) 559–601
7. Mika, S., Schölkopf, B., Smola, A.J., Müller, K.R., Scholz, M., Rätsch, G.: Kernel PCA and de-noising in feature spaces. In Kearns, M.S., Solla, S.A., Cohn, D.A., eds.: Advances in Neural Information Processing Systems 11, Cambridge, MA, MIT Press (1999) 536–542
8. Oja, E.: A simplified neuron model as a principal component analyzer. Journal of Mathematical Biology **15** (1982) 267–273
9. Sanger, T.D.: Optimal unsupervised learning in a single-layer linear feedforward neural netowork. Neural Networks **12** (1989) 459–473
10. Schölkopf, B., Smola, A.: Learning with Kernels. MIT Press, Cambridge, MA (2002)
11. Burges, C.J.C.: Simplified support vector decision rules. In Saitta, L., ed.: Proceedings of the 13th International Conference on Machine Learning, San Mateo, CA, Morgan Kaufmann (1996) 71–77
12. Kwok, J.T., Tsang, I.W.: (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.
13. Kim, K.I., Franz, M.O., Schölkopf, B.: Kernel Hebbian algorithm for iterative kernel principal component analysis. Technical Report 109, Max-Planck-Insitut für biologische Kybernetik, Tübingen (2003)
14. Georghiades, A.S., Belhumeur, P.N., Kriegman, D.J.: From few to many: illumination cone models for face recognition under variable lighting and pose. IEEE Trans. Pattern Analalysis and Machine Intelligence **23** (2001) 643–660
15. Mika, S., Rätsch, G., Weston, J., Schölkopf, B., Müller, K.R.: Fisher discriminant analysis with kernels. In Hu, Y.H., Larsen, J., Wilson, E., Douglas, S., eds.: Neural Networks for Signal Processing IX, IEEE (1999) 41–48
16. Gonzalez, R.C., Woods, R.E.: Digital Image Processing. Addison Wesley, Massachusetts (1992)
17. Mika, S., Schölkopf, B., Smola, A.J., Müller, K.R., Scholz, M., Rätsch, G.: Kernel PCA and de-noising in feature spaces. Advances in Neural Information Processing Systems, Cambridge, MA, MIT Press (1999) 536–542
18. Weston, J., Chapelle, O., Elisseeff, A., Schölkopf, B., Vapnik, V.: Kernel dependency estimation. Advances in Neural Information Processing Systems, Cambridge, MA, MIT Press (2003)