Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Aided Automation
Vienna University of Technology
Favoritenstr. 9/1832
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: {melzer,rei,bis}@prip.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-65 March 14, 2001

Kernel Canonical Correlation Analysis[1]

Thomas Melzer, Michael Reiter, and Horst Bischof

Abstract

This paper introduces a new non-linear feature extraction technique based on *Canonical Correlation Analysis* (CCA) with applications in regression and object recognition. The non-linear transformation of the input data is performed using kernel-methods. Although, in this respect, our approach is similar to other *generalized* linear methods like kernel-PCA, our method is especially well suited for relating two sets of measurements. The benefits of our method compared to standard feature extraction methods based on PCA will be illustrated with several experiments from the field of object recognition and pose estimation.

[1]This work was supported by the Austrian Science Foundation (FWF) under grant no. P13981-INF.

Contents

1 Introduction

When dealing with high-dimensional observations, linear mappings are often used to reduce the dimensionality of the data by extracting a small (compared to the superficial dimensionality of the data) number of linear features, thus alleviating subsequent computations. A prominent example of a linear feature extractor is *Principal Component Analysis* (PCA [7]). Among all linear, orthonormal transformations, PCA is optimal in the sense that it minimizes, in the mean square sense, the reconstruction error between the original signal $\mathbf{x}$ and the signal $\hat{\mathbf{x}}$ reconstructed from its low-dimensional representation $\mathbf{f}(\mathbf{x})$. During the recent years, PCA has been especially popular in the object recognition community, where it has successfully been employed in various applications such as face recognition [17], illumination planning [12], visual inspection and even visual servoing [14].

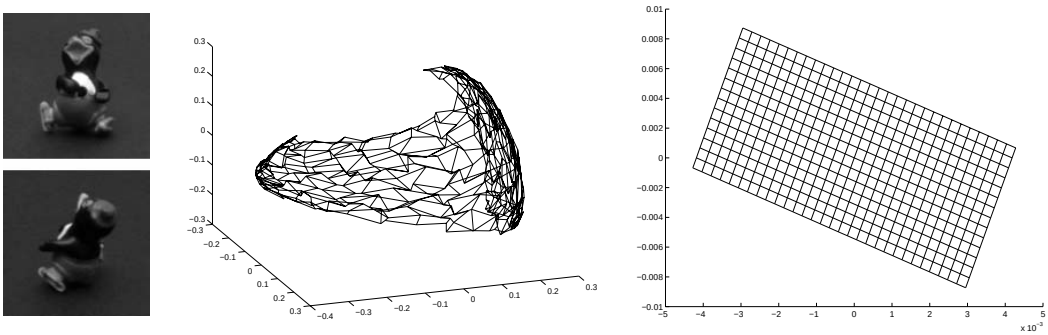
Although this demonstrates the broad applicability of PCA, one has to bear in mind that the goal of PCA is minimization of the reconstruction error; in particular, PCA-features are not well suited for regression tasks. Consider a mapping $\phi : \mathbf{x} \mapsto \mathbf{y}$. There is no reason to believe that the features extracted by PCA on the variable $\mathbf{x}$ will reflect the functional relation between $\mathbf{x}$ and $\mathbf{y}$ in any way; even worse, it is possible that information vital to establishing this relation is discarded when projecting the original data onto the PCA-feature space.

There exist, however, several other linear methods that are better suited for regression tasks, for example *Partial Least Squares* (PLS [6]), *Multivariate Linear Regression* (MLR, also referred to as *Reduced Rank Wiener Filtering*, see for example [5]) and *Canonical Correlation Analysis* (CCA [8]). Among these three, only MLR gives a direct solution to the linear regression problem. PLS and CCA will find pairs of directions that yield maximum covariance resp. maximum correlation between the two random variables $\mathbf{x}, \mathbf{y}$; regression can then be performed on these features. CCA, in particular, has some very attractive properties (for example, it is invariant w.r.t. affine transformations - and thus scaling - of the input variables) and can not only be used for regression purposes, but whenever we need to establish a relation between two sets of measurements (e.g., finding corresponding points in stereo images [2]).

As an example for CCA, consider constructing a parametric manifold for pose estimation [13]. Fig. 1(a) shows two extreme views of an object, which was acquired with two varying pose parameters (pan and tilt). Let $\mathbf{X}$ denote the set of training images and $\mathbf{Y}$ the set of corresponding pose parameters.The visualization of the manifold given in Fig. 1(b) is obtained by plotting the projections of the training set onto the first three eigenvectors obtained by standard PCA, whereby neighboring (w.r.t. the pose parameters) projections are connected.

The parametric manifold serves as a starting point for computing pose estimates for new input images. The standard-approach for retrieving these estimates is to resample the manifold using, e.g., bicubic spline interpolation and then to perform a nearest neighbor search for each new image [13].

Fig. 1(c) shows the manifold obtained by projecting the training images onto the first two directions found by computing CCA on $\mathbf{X}$ and $\mathbf{Y}$ (the number of factors obtained by CCA is limited by the dimensionality of the lower-dimensional set). In contrast to the PCA-manifold, the CCA-factors span a perfect grid; one could also say that projections of the training images onto the two linear features found by CCA are topologically ordered w.r.t. their associated pose parameters. It is obvious that pose estimation on the manifold obtained by CCA is much easier than on the PCA-manifold.



(a) (b) (c)

Figure 1: Extreme views of training set (a) and the parametric manifold obtained with PCA (b) and CCA (c).

In this paper we will propose a non-linear extension of CCA by the use of kernel-methods [9]. Kernel-methods have become increasingly popular during the last few years, and have already been applied to PCA [16] and the *Fisher Discriminant* [10]. In our derivation of kernel-CCA we have used the fact that the solutions (principal directions) of CCA can be obtained as the extremum points of an appropriately chosen *Rayleigh Quotient* (this is also true for the other linear techniques discussed thus far, see [2]). We will also demonstrate the benefits of kernel-CCA with an application in the field of appearance-based pose estimation. To this end we will compare the performance of features obtained by PCA, standard CCA and kernel-CCA.

The rest of this paper is organized as follows: While section 2.1 gives a brief introduction to "classical" CCA, section 2.2 discusses kernel-methods. In section 2.3, we will bring the two concepts together and give a formal derivation of kernel-CCA. Several numerical issues pertaining to kernel-CCA will be discussed in section 2.4. Experimental results are given in section 3, followed by conclusions in section 4.

2 Canonical Correlation Analysis (CCA)

2.1 What is CCA?

Given two zero-mean random variables $\mathbf{x} \in \mathbb{R}^p$ and $\mathbf{y} \in \mathbb{R}^q$, CCA finds pairs of directions $\mathbf{w}_x$ and $\mathbf{w}_y$ that maximize the correlation between the projections $x = \mathbf{w}_x^T\mathbf{x}$ and $y = \mathbf{w}_y^T\mathbf{y}$ (in the context of CCA, the projections x and y are also referred to as *canonical variates*). More formally, CCA maximizes the function:

$$\rho = \frac{E[xy]}{\sqrt{E[x^2]E[y^2]}} = \frac{E[\mathbf{w}_x^T\mathbf{x}\mathbf{y}^T\mathbf{w}_y]}{\sqrt{E[\mathbf{w}_x^T\mathbf{x}\mathbf{x}^T\mathbf{w}_x]E[\mathbf{w}_y^T\mathbf{y}\mathbf{y}^T\mathbf{w}_y]}}, \tag{1}$$

$$\rho = \frac{\mathbf{w}_x^T\mathbf{C}_{xy}\mathbf{w}_y}{\sqrt{\mathbf{w}_x^T\mathbf{C}_{xx}\mathbf{w}_x\mathbf{w}_y^T\mathbf{C}_{xy}\mathbf{w}_y}}. \tag{2}$$

Let

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{C}_{xy} \\ \mathbf{C}_{yx} & \mathbf{0} \end{pmatrix}, \ \mathbf{B} = \begin{pmatrix} \mathbf{C}_{xx} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{yy} \end{pmatrix}. \tag{3}$$

It can be shown [2] that the stationary points $\mathbf{w}^* = (\mathbf{w}_x^{*T}, \mathbf{w}_y^{*T})^T$ of ρ (i.e., the points satisfying $\nabla\rho(\mathbf{w}^*) = \mathbf{0}$) coincide with the stationary points of the *Rayleigh Quotient*:

$$r = \frac{\mathbf{w}^T\mathbf{A}\mathbf{w}}{\mathbf{w}^T\mathbf{B}\mathbf{w}}, \tag{4}$$

and thus, by virtue of the *Generalized Spectral Theorem* [5], can be obtained as solutions (i.e., eigenvectors) of the corresponding generalized eigenproblem:

$$\mathbf{A}\mathbf{w} = \mu\mathbf{B}\mathbf{w}. \tag{5}$$

The extremum values $\rho(\mathbf{w}^*)$, which are referred to as *canonical correlations*, are equally obtained as the corresponding extremum values of Eq. 4 or the eigenvalues of Eq. 5, respectively, i.e., $\rho(\mathbf{w}^*) = r(\mathbf{w}^*) = \mu(\mathbf{w}^*)$.

2.2 The Kernel Trick

In the field of pattern recognition, kernel-methods were originally proposed as a non-linear extension of the *Support Vector Machine* (SVM) classifier [3]. Recently, there has been an increased interest in kernel-based methods, leading, for example, to the formulation of kernel-PCA [16] and the kernel-*Fisher Discriminant* [10]. For a comprehensive overview, see [4] or [15] [1].

To illustrate the key idea behind kernel-methods, consider a simple binary classification problem in $\mathbb{R}^p$ with non-overlapping classes. As long as the two classes are linearly separable, linear classifiers like the perceptron or the SVM will find a solution (a $n-1$ dimensional separating hyperplane). If, however, the training data $\mathbf{x}_i \in \mathbb{R}^p$ are not linearly separable, we can still employ a linear classifier by first mapping the input data into some (high dimensional) feature space:

$$\phi : \mathbb{R}^p \rightarrow \mathbb{R}^s, s > p. \tag{6}$$

This approach is sometimes referred to as *Generalized Linear Discriminant* [1].

The kernel-trick can be applied whenever it is possible to formulate the classifier in such a way that it uses only dot products of the transformed input data; the dot products in feature space are than expressed in terms of kernel-functions in input space, i.e., $\phi(\mathbf{x})^T\phi(\mathbf{y}) = k(\mathbf{x}, \mathbf{y})$. A sufficient condition for a kernel-function $k(.,.)$ to correspond to a dot product in some feature space is given by *Mercers Theorem* (see, for example, [4]). Prominent examples of *Mercer*-kernels are RBF-kernels $k(\mathbf{x}, \mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|}{\sigma}}$ and polynomial kernels $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}, \mathbf{y})^v, v \in \mathbf{N}$. In particular, the mapping corresponding to a polynomial kernel of degree v is given by the function $C_v(\mathbf{x})$ which maps its argument to the vector of all vth-order monomial. For example, for $p = v = 2$, we have $C_v(x_1, x_2) = (x_1^2, x_2^2, x_1x_2, x_2x_1)$. It can easily be shown that:

$$C_v(\mathbf{x})^T C_v(\mathbf{y}) = (\mathbf{x}^T\mathbf{y})^v, \forall v \in \mathbf{N}, \mathbf{x}, \mathbf{y} \in \mathbb{R}^p. \tag{7}$$

[1] Interested readers are also referred to the WWW-address: *www.kernel-machines.org*.

Choosing $v = 2$, for example, we obtain a quadratic classifier. Clearly, the above discussion applies not only to classifiers, but to any algorithm that can be expressed solely in terms of dot products.

Kernel-functions allow us to easily construct different nonlinear versions of the original algorithm. Furthermore, the complexity of the transformed problem depends only on the size of the training set, not on the dimensionality of the feature space (this will become more clear in the next section); hence, the kernel-approach enables us to work with feature spaces of arbitrarily high (or even infinite) dimensionality.

2.3 Kernel CCA

Given n pairs of mean-normalized observations $(\mathbf{x}_i^T, \mathbf{y}_i^T)^T \in \mathbb{R}^{p+q}$, and data matrices $\mathbf{X} = (\mathbf{x}_1..\mathbf{x}_n) \in \mathbb{R}^{p\times n}$, $\mathbf{Y} = (\mathbf{y}_1..\mathbf{y}_n) \in \mathbb{R}^{q\times n}$, we obtain the estimates for the covariance matrices $\mathbf{A}, \mathbf{B}$ in Eq. 3 as

$$\hat{\mathbf{A}} = \frac{1}{n}\begin{pmatrix} \mathbf{0} & \mathbf{X}\mathbf{Y}^T \\ \mathbf{Y}\mathbf{X}^T & \mathbf{0} \end{pmatrix}, \; \hat{\mathbf{B}} = \frac{1}{n}\begin{pmatrix} \mathbf{X}\mathbf{X}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}\mathbf{Y}^T \end{pmatrix} \tag{8}$$

If the mean was estimated from the data, we have to replace n by $n-1$ in both equations.

As we know from section 2.1, computing the CCA between the data sets $\mathbf{X}, \mathbf{Y}$ amounts to determining the extremum points of the *Rayleigh Quotient*:

$$r = \frac{\mathbf{w}^T\hat{\mathbf{A}}\mathbf{w}}{\mathbf{w}^T\hat{\mathbf{B}}\mathbf{w}}, \tag{9}$$

which are obtained as the solutions of:

$$\hat{\mathbf{A}}\mathbf{w} = \mu\mathbf{w}\hat{\mathbf{B}}. \tag{10}$$

In order to apply the kernel-trick, we have to show that CCA can completely be expressed in terms of dot products. To this end, we need:

Proposition 1 *For all solutions $\boldsymbol{w}^* = (\boldsymbol{w}_x^{*T}, \boldsymbol{w}_y^{*T})^T$ of Eq. 10, the component vectors $\boldsymbol{w}_x^*, \boldsymbol{w}_y^*$ lie in the span of the training data, i.e., $\boldsymbol{w}_x^* \in span(\boldsymbol{X})$ and $\boldsymbol{w}_y^* \in span(\boldsymbol{Y})$. The proof is given in the appendix.*

According to Proposition 1, for each eigenvector $\mathbf{w}^* = (\mathbf{w}_x^{*T}, \mathbf{w}_y^{*T})^T$ solving Eq. 10, there exist vectors $\mathbf{f}, \mathbf{g} \in \mathbb{R}^n$, so that $\mathbf{w}_x^* = \mathbf{X}f$ and $\mathbf{w}_y^* = \mathbf{Y}g$. Thus, we may reformulate Eq. 9 as:

$$\frac{\begin{pmatrix} \mathbf{f}^T & \mathbf{g}^T \end{pmatrix}\begin{pmatrix} \mathbf{X}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}^T \end{pmatrix}\hat{\mathbf{A}}\begin{pmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y} \end{pmatrix}\begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix}}{\begin{pmatrix} \mathbf{f}^T & \mathbf{g}^T \end{pmatrix}\begin{pmatrix} \mathbf{X}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}^T \end{pmatrix}\hat{\mathbf{B}}\begin{pmatrix} \mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y} \end{pmatrix}\begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix}} =$$

$$\frac{\begin{pmatrix} \mathbf{f}^T & \mathbf{g}^T \end{pmatrix}\begin{pmatrix} \mathbf{0} & (\mathbf{X}^T\mathbf{X})(\mathbf{Y}^T\mathbf{Y}) \\ (\mathbf{Y}^T\mathbf{Y})(\mathbf{X}^T\mathbf{X}) & \mathbf{0} \end{pmatrix}\begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix}}{\begin{pmatrix} \mathbf{f}^T & \mathbf{g}^T \end{pmatrix}\begin{pmatrix} (\mathbf{X}^T\mathbf{X})^2 & \mathbf{0} \\ \mathbf{0} & (\mathbf{Y}^T\mathbf{Y})^2 \end{pmatrix}\begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix}}. \tag{11}$$

Eq. 11 shows that we can indeed reformulate the *Rayleigh Quotient* using only dot products, which is a necessary and sufficient condition to apply the kernel-trick. Furthermore, Proposition 1 shows that we can efficiently compute CCA in high dimensional spaces (since the size of the *Gram* matrices in Eq. 11 depends only on the number of training samples, not on their dimensionality).

Now suppose we want to compute CCA on features that are non-linearly related to the input data rather than on the input data itself, i.e., we are looking for the CCA of the vectors $\phi(\mathbf{X}) = (\phi(\mathbf{x}_1)..\phi(\mathbf{x}_n))$ and $\theta(\mathbf{Y}) = (\theta(\mathbf{y}_1)..\phi(\mathbf{y}_n))$, whereby $\phi : \mathbb{R}^p \rightarrow \mathbb{R}^{m_1}$ and $\theta : \mathbb{R}^q \rightarrow \mathbb{R}^{m_2}$ are non-linear mappings. To simplify the discussion, we shall assume that the mapped data are centered, i.e., have zero-mean; non-centered data can be accounted for in exactly the same fashion as proposed in [16] for kernel-PCA.

The decomposition of the *Rayleigh Quotient* given in Eq. 11 makes it straightforward to apply the kernel-trick, i.e., to compute CCA on the mapped data without actually having to compute the mappings ϕ, θ themselves. To show this, let us define the kernel matrices $\mathbf{K}, \mathbf{L}$ by $\mathbf{K}_{ij} = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ and $\mathbf{L}_{ij} = \theta(\mathbf{y}_i)^T\theta(\mathbf{y}_j)$, $\mathbf{K}, \mathbf{L} \in \mathbb{R}^{n\times n}$. Then, by substituting $\phi(\mathbf{X})$ for $\mathbf{X}$ and $\theta(\mathbf{Y})$ for $\mathbf{Y}$ in Eq. 11, the *Rayleigh Quotient* for the transformed data becomes:

$$\frac{\begin{pmatrix} \mathbf{f}_\phi^T & \mathbf{g}_\theta^T \end{pmatrix}\begin{pmatrix} \mathbf{0} & \mathbf{KL} \\ \mathbf{LK} & \mathbf{0} \end{pmatrix}\begin{pmatrix} \mathbf{f}_\phi \\ \mathbf{g}_\theta \end{pmatrix}}{\begin{pmatrix} \mathbf{f}_\phi^T & \mathbf{g}_\theta^T \end{pmatrix}\begin{pmatrix} \mathbf{K}^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{L}^2 \end{pmatrix}\begin{pmatrix} \mathbf{f}_\phi \\ \mathbf{g}_\theta \end{pmatrix}}, \tag{12}$$

whereby $\mathbf{f}_\phi, \mathbf{g}_\theta$ are the coefficients of the linear expansion of the principal vectors $\mathbf{w}_\phi^*, \mathbf{w}_\theta^*$ in terms of the transformed data, i.e., $\phi(\mathbf{X})\mathbf{f}_\phi = \mathbf{w}_\phi^*$ and $\theta(\mathbf{Y})\mathbf{f}_\theta = \mathbf{w}_\theta^*$.

Now let $k(.,.)$ be the kernel functions corresponding to the mapping θ. The projections onto $\mathbf{w}_\phi^*$ can be computed using only the kernel function, without having to evaluate ϕ itself:

$$\phi(\mathbf{x})^T\mathbf{w}_\phi^* = \sum_{i=1}^{n} f_{\phi i}\phi(\mathbf{x})^T\phi(\mathbf{x}_i) = \sum_{i=1}^{n} f_{\phi i}k(\mathbf{x}, \mathbf{x}_i). \tag{13}$$

The projections onto $\mathbf{w}_\theta^*$ are obtained analogously.

There are several interesting issues pertaining to kernel-CCA:

- Using kernel-CCA, we can compute more than $min(p, q)$ (p, q being the dimensionality of the variable $\mathbf{x}$ and $\mathbf{y}$, respectively) factor pairs, which is the limit imposed by classical CCA.

- By choosing either ϕ or θ to be the identity mapping, we can selectively kernelize either $\mathbf{x}$ or $\mathbf{y}$. In particular, when kernel-CCA is used for regression purposes, it is easier to kernelize only the input-variable $\mathbf{x}$; otherwise we face the problem of computing $\mathbf{y}$ from the estimate obtained for $\theta(\mathbf{y})$, i.e., of inverting θ. Although the problem of inverting a kernel-function has been addressed by Mika et al. [11] in the context of kernel-PCA, their solution is based on the minimization of the reconstruction error and can thus not be applied to CCA. Thus, for kernel-CCA-regression involving a kernelized output-space, a satisfactory solution has still to be found.

- The derivation given above can easily be adapted to other linear feature extractors, provided their principal vectors can be obtained as extremum points of an appropriately chosen *Rayleigh Quotient* (this is the case, for example, for PLS and MLR, see [2]).

2.4 Numerical Issues

Unless the kernel matrices $\mathbf{K}$ and $\mathbf{L}$ introduced in the previous section have full rank n (whereby n is the number of training samples), the block matrix:

$$\mathbf{C} = \begin{pmatrix} \mathbf{K}^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{L}^2 \end{pmatrix} \tag{14}$$

in the denominator of Eq. 12 will become singular. We will briefly discuss two possible approaches to making the problem numerically feasible:

- We can make $\mathbf{C}$ positive definite by adding a multiple of the identity matrix:

$$\mathbf{C}_\kappa = \mathbf{C} + \kappa\mathbf{I} \tag{15}$$

  This approach, which is akin to ridge regression, has also been proposed in [10] for the kernel-*Fisher Discriminant*. It can be regarded as a kind of *regularization*, i.e., of providing *a priori* knowledge on the problem by imposing additional constraints on the solution.

- It is also possible to make $\mathbf{C}$ positive definite by choosing appropriate kernel functions (for example, in the case of a RBF-kernel, we simply have to choose a sufficiently small variance parameter σ; in the limit, we will even obtain a diagonal matrix).

But even if the matrices $\mathbf{K}$ and $\mathbf{L}$ are non-singular, their magnitudes can still differ considerably, in which case the matrix $\mathbf{C}$ will again become (near) singular. For this reason, we divide both $\mathbf{K}$ and $\mathbf{L}$ by their largest eigenvalue; this can be interpreted as rescaling of the transformed data matrices (by the inverse of their largest singular value) and thus will not affect the outcome of the CCA-computation.

3 Experiments

In the following example we apply CCA to a pose estimation problem, where we relate images of objects at varying pose to corresponding pose parameters. Experiments were conducted on three test objects, shown in figure 2. For each object a set of 180 images was obtained by rotating the object through 180 poses in 2 degrees steps. Figure 3 shows some of the views of object 2(b) images.

(a) (b) (c)

Figure 2: The 3 test objects used in the experiments

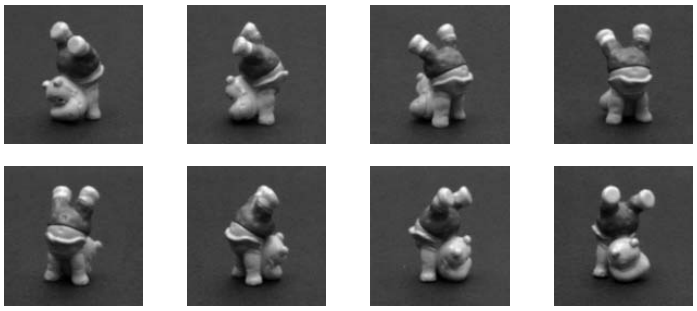
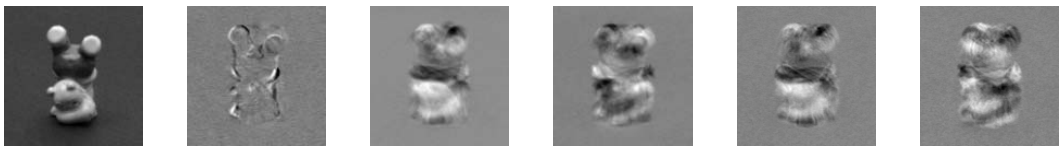


Figure 3: A subset of images of test object 2(b)



(a) (b) (c) (d) (e) (f)

Figure 4: (a) Image $\mathbf{x}_1$ of one of our test objects. (b) Canonical factor $\mathbf{w}_x^*$ computed on $\mathbf{X}_8^t$. (c),(d) Canonical factors obtained on the same set using a non-linear (trigonometric) representation of the output parameter space. (e),(f) 2 factors obtained by kernel-CCA with canonical correlations $\rho = 1.0$.

Let $\mathbf{X} = \langle \mathbf{x}_i | 1 \leq i \leq 180 \rangle$ denote the set of images and $\mathbf{Y} = \langle y_i | y_i \in [0, 358] \rangle$ corresponding pose parameters (horizontal orientation of the object w.r.t. the camera in degrees). The images are represented as 128^2-dimensional vectors that are obtained by sequentially stacking the image columns.

Each image set $\mathbf{X}$ was subsampled to obtain a subset of images that was used as a training set. The remaining images were assigned to the corresponding test set. Let $\mathbf{X}_k$ denote a training set that was generated by subsampling $\mathbf{X}$ at every kth position[2]. Since y_{i_k} is a scalar, standard CCA yields only one feature vector (canonical factor) $\mathbf{w}_x^*$ (see figure 4(b)). Figure 6 gives a quantitative comparison of the pose estimation error ϵ (linear regression model) using the PCA and standard CCA. The pose estimation error ϵ was calculated on the test set as the absolute difference between known and estimated parameter value (orientation in degrees).

Figure 5(a) shows a plot of pose estimates for the complete set of images $\mathbf{X}$ obtained using CCA. The dotted line indicates the true pose parameter values y_i. Pose parameter values of $\mathbf{X}_k$ are marked by filled circles. In figure 5 a linear least-square mapping was used to map the projections of the training set to pose parameters.

Note that the pose estimation error grows rapidly around $i = 180$. This problem is due to the fact that the scalar representation for y_i has a discontinuity at $y_i = 360$. From figure 4(a) it can be seen that the main part of information held in $\mathbf{w}_x^*$ is about the transition of image $\mathbf{x}_{180}$ to image $\mathbf{x}_1$.

[2] $\mathbf{X}_k$ contains all images $\mathbf{x}_{i_k}$ with $i_k \in \{1, k, 2k, \ldots, \lfloor 180/k \rfloor\}$. Since the original set shows the object in 2 degrees steps, $\mathbf{X}_k$ shows the object at $2k$ degrees steps.

For this reason we chose an periodic, trigonometric representation of pose parameters $\mathbf{y}_i = [sin(y_i), cos(y_i)]^T$. The object's pose is now characterized by 2 parameters, and thus CCA yields 2 canonical factors (see figure 4 (c) and (d)) in the image vector space. Estimates obtained for these intermediate output values (again using linear regression) are given in 5(b), while the final pose estimates (obtained by combining these two estimates using atan2) are given in figure 5(d). Thus, by using *a priori* knowledge about the problem domain a significant increase in pose estimation accuracy could be obtained.

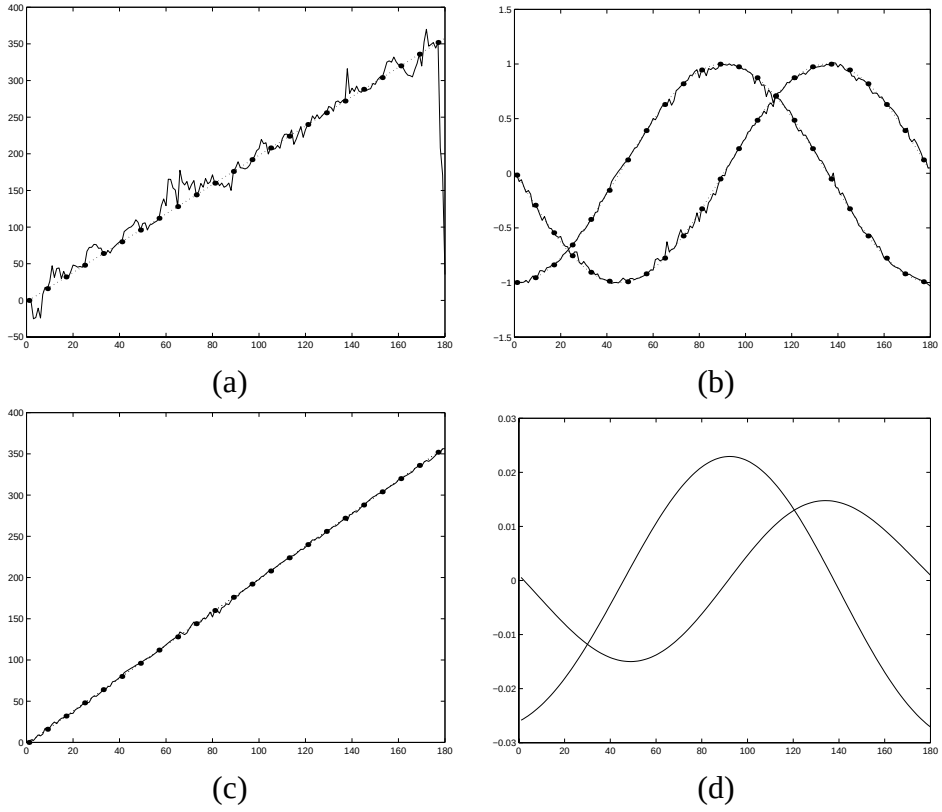


Figure 5: Output parameter estimates obtained from feature projections by the linear regression function of the training set. Horizontal axes correspond to the image indices, vertical axes to estimated output parameter values. The dotted line indicates the true pose parameter values. Parameter values of the training set are marked by filled circles. (a) shows estimates using scalar representation of orientation (b) using trigonometric representation. (c) Estimated pose obtained from trigonometric representation using the four-quadrant arc tangens. Note that the accuracy of estimated pose parameters can be improved considerably. (d) Projections onto factors obtained using kernel-CCA: optimal factors can be obtained automatically.

The last experiment shows the relative performance of CCA, kernel-CCA and PCA when using spline interpolation and resampling [13] for pose estimation. For standard CCA we used the hard-coded trigonometric pose representation (yielding 2 factor pairs), while kernel-CCA obtained similar factors automatically from the original scalar pose representation (see figures 4 (e)-(f) and 5(d)).

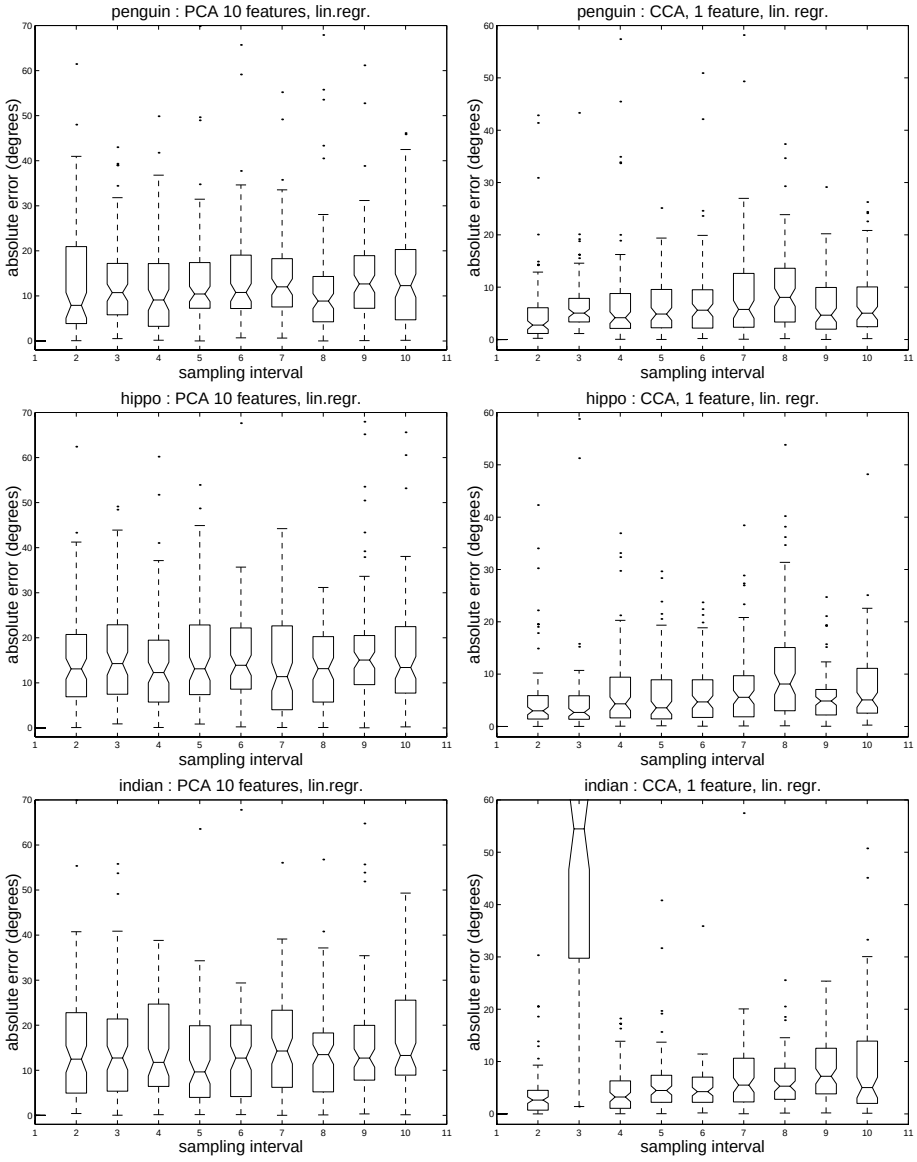


Figure 6: Comparison of pose estimation error when using a linear regression from factor projections to pose parameters. The plots in the left column show the errors for a 10 dimensional PCA-feature space for all three test objects. The plots in the right column show the results when using only one CCA-feature. Image sets have been subsampled at different intervals (horizontal axis) to obtain increasingly smaller training sets. Subsampling was done at at 9 different sampling intervals ($k = \langle 2, ..., 10 \rangle$). Note that the plots have been scaled differently.

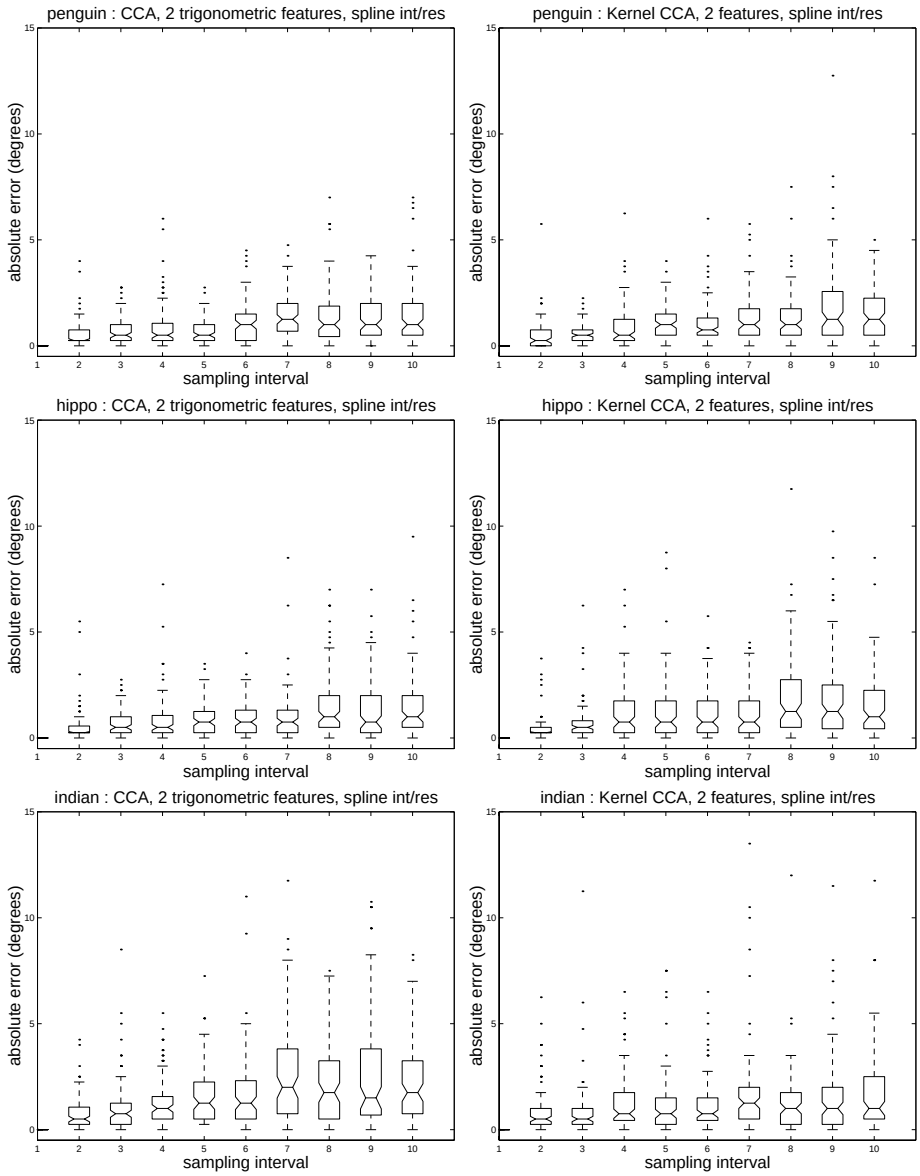


Figure 7: The left column shows results for "classical" CCA when using a 2 dimensional trigonometric representation of output parameters. The right column shows results when using 2 features obtained by kernel-CCA and a scalar pose representation.

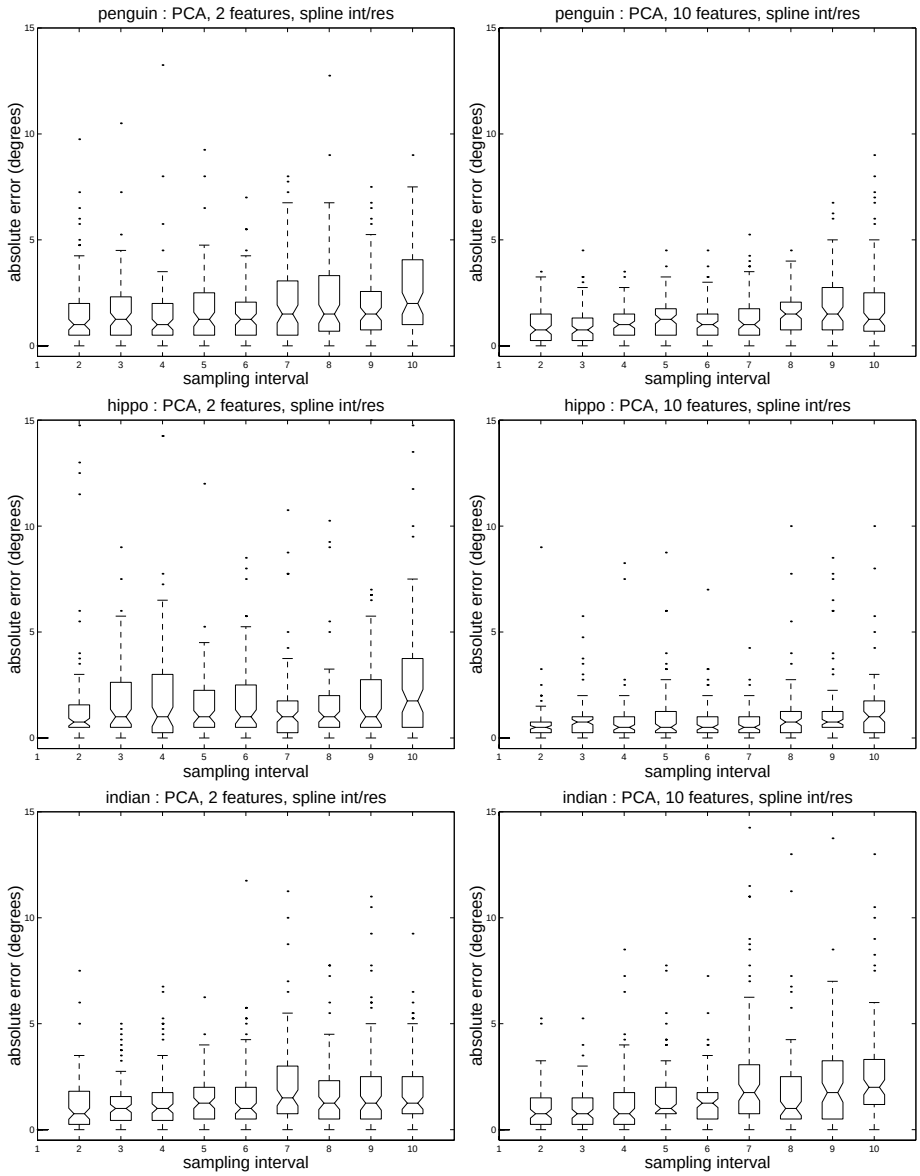


Figure 8: Pose estimation error on 3 image sets using the first 10 eigenvectors (plots in the left column) and only 2 eigenvectors (plots in the right column). Estimates were obtained from feature projections using spline interpolation and resampling.

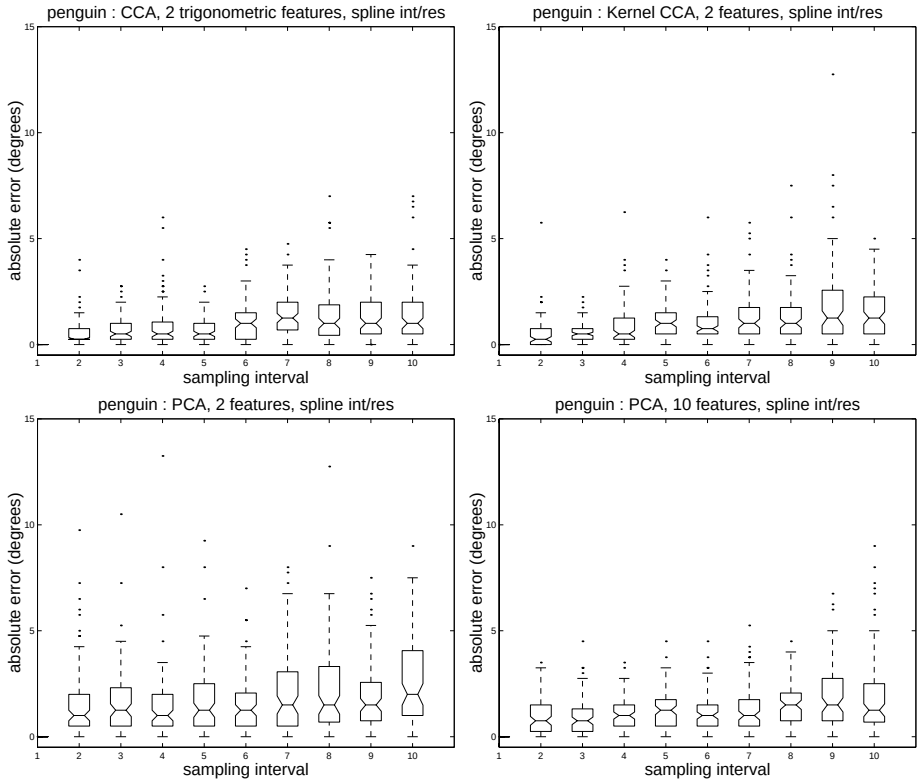


Figure 9: Comparision of pose estimation performance of CCA, kernel-CCA and PCA on the penguin image set. First row: Pose estimation error for CCA-features. The left column shows results for "classical" CCA when using a 2 dimensional trigonometric representation of output parameters. The right column shows results when using 2 features obtained by kernel-CCA and a scalar pose representation. Second row: Pose estimation error using the first 10 eigenvectors (left plot) and only 2 eigenvectors (right plot). Estimates were obtained from feature projections using spline interpolation and resampling.

Results are given in figure 7 and figure 9. Figure 8 shows results for PCA. The pose estimation error is significantly larger compared to CCA when using the same number of features.

4 Conclusion and Outlook

Although little known in the field of pattern recognition and signal processing, CCA is a very powerful and versatile statistical tool that is especially well suited for relating two sets of measurements. CCA, like PCA, can also be regarded as a linear feature extractor. CCA-features are, however, much better suited for regression tasks than features obtained by PCA; this was demonstrated in section 1 for the task of computing a parametric object manifold for pose estimation.

In section 2, we discussed how to non-linearly extend CCA by using kernel-functions. **Kernel-CCA** is a efficient non-linear feature extractor, which also overcomes some of the limitations of classical CCA.

Finally, in section 3, we applied kernel-CCA to an object pose estimation problem. There, it was also shown that kernel-CCA will automatically find an optimal, periodic representation for a training set containing object views ranging from 0 to 360 degrees (i.e., for *periodic* data).

A Proof of Proposition 1

Consider the spectral factorization of $\hat{\mathbf{B}}$ (Eq. 8):

$$\hat{\mathbf{B}} = \mathbf{E}\Lambda E^T = \sum_{i=1}^{p+q} \lambda_i \mathbf{e}_i \mathbf{e}_i^T \tag{16}$$

whereby $\mathbf{E}$ is an orthonormal matrix, whose columns $\mathbf{e}_i$ are the eigenvectors of $\hat{\mathbf{B}}$, and Λ is a diagonal matrix containing the corresponding eigenvalues[3] $\lambda_i > 0$. For each eigenvector $\mathbf{e} = (\mathbf{e}_x^T, \mathbf{e}_y^T)^T$ we have:

$$\hat{\mathbf{B}}\mathbf{e} = \begin{pmatrix} \mathbf{X}\mathbf{X}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}\mathbf{Y}^T \end{pmatrix} \begin{pmatrix} \mathbf{e}_x \\ \mathbf{e}_y \end{pmatrix} =$$

$$\begin{pmatrix} \mathbf{X}(\mathbf{X}^T\mathbf{e}_x) \\ \mathbf{Y}(\mathbf{Y}^T\mathbf{e}_y) \end{pmatrix} = \mu \begin{pmatrix} \mathbf{e}_x \\ \mathbf{e}_y \end{pmatrix} \tag{17}$$

which shows that the component vectors $\mathbf{e}_x, \mathbf{e}_y$ are linear combinations of the training data.

Let us assume for the moment that $\hat{\mathbf{B}}$ is non-singular. We may then reformulate Eq. 10 as:

$$\hat{\mathbf{B}}^{-1}\hat{\mathbf{A}}\mathbf{w} = \mu\mathbf{w}. \tag{18}$$

Combining Eq. 18 and Eq. 16 and observing that $\hat{\mathbf{B}}^{-1} = \mathbf{E}\Lambda^{-1}\mathbf{E}^T$, we obtain:

$$\hat{\mathbf{B}}^{-1}\hat{\mathbf{A}}\mathbf{w}^* = \mathbf{E}(\Lambda^{-1}\mathbf{E}^T\hat{\mathbf{A}}\mathbf{w}^*) = \lambda \begin{pmatrix} \mathbf{w}_x^* \\ \mathbf{w}_y^* \end{pmatrix} = \lambda\mathbf{w}^* \tag{19}$$

for all solutions $\mathbf{w}^*$ of Eq. 18; thus, each $\mathbf{w}^*$ is a linear combinations of the eigenvectors of $\hat{\mathbf{B}}$. Let us further define:

$$\begin{aligned} \mathbf{E}_x &= (\mathbf{e}_{x(1)}..\mathbf{e}_{x(p+q)}) \in \mathbb{R}^{p\times(p+q)}, \\ \mathbf{E}_y &= (\mathbf{e}_{y(1)}..\mathbf{e}_{y(p+q)}) \in \mathbb{R}^{q\times(p+q)}. \end{aligned} \tag{20}$$

From Eq. 17 it is clear that each column of $\mathbf{E}_x$ ($\mathbf{E}_y$) lies in the span of $\mathbf{X}$ ($\mathbf{Y}$), respectively, and this must of course also be true for any linear combination of their columns. Since

$$\mathbf{E} = \begin{pmatrix} \mathbf{E}_x \\ \mathbf{E}_y \end{pmatrix}, \tag{21}$$

we conclude that *Proposition 1* is true, provided that $det(\hat{\mathbf{B}}) > 0$. If $\hat{\mathbf{B}}$, however, is singular, we can perform regularization (cf. Eq. 15). Note that this operation simply shifts the eigenvalues of $\hat{\mathbf{B}}$, but leaves its eigenvectors unchanged, so *Proposition 1* still holds.

[3] We use $\lambda, \mathbf{e}$ to denote the eigenvalue/vector pairs of $\hat{\mathbf{B}}$ to distinguish them from the solutions $\mu, \mathbf{w}$ of the generalized eigenproblem

References

[1] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[2] Magnus Borga. *Learning Multidimensional Signal Processing*. Linköping Studies in Science and Technology, Dissertations, No. 531. Department of Electrical Engineering, Linköping University, Linköping, Sweden, 1998.

[3] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proc. of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

[4] Nello Christianini and John Shawne-Taylor. *Support Vector Machines and Other Kernel-Based Methods*. Cambridge University Press, 2000.

[5] Konstantinos I. Diamantaras and S.Y. Kung. *Principal Component Neural Networks*. John Wiley & Sons, 1996.

[6] A. Höskuldsson. PLS regression methods. *Journal of Chemometrics*, 2:211–228, 1988.

[7] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:498–520, 1933.

[8] H. Hotelling. Relations between two sets of variates. *Biometrika*, 8:321–377, 1936.

[9] Thomas Melzer and Micheal Reiter. Pose estimation using parametric stereo eigenspaces. In Tomas Svoboda, editor, *Proc. of Czech Pattern Recognition Workshop*, pages 77–80. Czech Pattern Recognition Society Praha, 2000.

[10] 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*, volume 9, pages 41–48. IEEE, 1999.

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

[12] Hiroshi Murase and Shree K. Nayar. Illumination planning for object recognition using parametric eigenspaces. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 16(12):1219–1227, December 1994.

[13] Hiroshi Murase and Shree K. Nayar. Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, January 1995.

[14] Shree K. Nayar, Sameer A. Nene, and Hiroshi Murase. Subspace methods for robot vision. *IEEE Trans. Robotics and Automation*, 12(5):750–758, October 1996.

[15] Bernhard Schölkopf. *Support Vector Learning*. PhD thesis, Technische Universität Berlin, 1997.

[16] Bernhard Schölkopf, Alex Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[17] Matthew Turk and Alexander P. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 4(1):71–86, 1991.