Weighted Incremental Subspace Learning *

Danijel Skočaj and Aleš Leonardis
University of Ljubljana
Faculty of Computer and Information Science
Tržaška 25, SI-1001, Slovenia
{danijels,alesl}@fri.uni-lj.si

Abstract

In a cognitive vision system, learning is expected to be a continuous process, which treats input images and pixels selectively. In this paper we present a method for subspace learning, which takes these considerations into account. First, we present a generalized PCA approach, which estimates the principal subspace considering weighted pixels and images. Next, we propose a method for incremental learning, which sequentially updates the principal subspace. Finally, we combine these two techniques into a unified method for weighted incremental subspace learning.

1. Introduction

Learning representations of objects and scenes is an essential part of any cognitive vision system. In the real world, learning is usually a continuous, never-ending process, thus requiring incremental methods for updating previously learnt representations. In addition, one can also expect that previous experience, prior knowledge, and the information obtained by higher-level cognitive processes affect the level to which a newly encountered training images are incorporated in the representation. Therefore, a learning algorithm should enable a selective influence of training images as well as pixels in individual images to the process of learning. In this paper we present a method, which takes these considerations into account and builds representations of objects and scenes using a weighted incremental approach.

Often, learning is approached as appearance-based modelling of objects and scenes, which is commonly realized using Principal component analysis (PCA). However, in the standard PCA approach, all pixels of an image receive equal treatment. Also, all the training images have equal influence on the estimation of the principal axes. In this paper, we present a generalized PCA approach, which estimates principal axes and principal components considering weighted pixels and images.

In addition, the standard PCA approach is usually performed in a batch mode, i.e., all training images are processed simultaneously, which means that all of them have to be given in advance. This is inadmissible in an on-line scenario, where the images to be processed are obtained sequentially. In this paper we propose a method for incremental learning, which processes images sequentially one by one and updates the principal subspace accordingly. Finally, the algorithm for weighted learning is embedded in this incremental framework, resulting in a weighted incremental method.

A number of methods for estimation of principal axes in the presence of data with varying reliability and missing data have already been proposed [4, 16, 3, 12], however, all of them operate in a batch mode. Several algorithms for incremental learning have been proposed as well [5, 6, 2], but they are not suitable for weighted learning. One exception is the method for incremental learning with temporal weighting proposed by Liu and Chen [9]. However, their method considers only a special case of temporal weights. In contrast, we propose a novel method for weighted and incremental learning, which considers arbitrary temporal and spatial weights.

The paper is organized as follows. First, we briefly outline the standard PCA and expose its drawbacks. Then, we present the batch weighted method. In section 4, we propose a method for incremental learning. We combine both techniques in a weighted incremental method in section 5. Then, we present the experimental results. Finally, we discuss some issues related to the proposed method and summarize the paper.

2. Standard PCA

In this section we briefly outline the standard PCA and introduce the notation. Let $\mathbf{x}_i$ =

* This work was supported in part by the EU IST project Cognitive Vision Systems - CogVis (IST-2000-29375), and the following grants funded by the Ministry for Education, Science, and Sport: Research Program *Computer Vision*-1539-506, and SLO-A/07.

$[x_{1i}, \ldots, x_{Mi}]^\top \in \mathbb{R}^M$ be an individual image represented as a vector and $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{M \times N}$ the matrix of all training images. For further processing, we create the mean normalized data matrix $\hat{\mathbf{X}}$ by subtracting the mean image $\boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$ from the input images.

Principal axes are traditionally obtained by maximizing the variance in their directions by performing the eigendecomposition (or, similarly, the singular value decomposition) of the covariance matrix (or, alternatively, the inner product matrix) of the input data. We will denote the eigenvectors (principal axes) by $\mathbf{u}_i = [u_{1i}, \ldots, u_{Mi}]^\top \in \mathbb{R}^M$; $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_N] \in \mathbb{R}^{M \times N}$. The columns of $\mathbf{U}$, i.e., the eigenvectors, are arranged in a decreasing order with respect to the corresponding eigenvalues. Usually, only k, $k < N$, eigenvectors with the largest eigenvalues are needed to represent $\hat{\mathbf{x}}$ to a sufficient degree of accuracy as a linear combination of eigenvectors:

$$\tilde{\mathbf{x}} = \sum_{j=1}^{k} a_j \mathbf{u}_j = \mathbf{U}\mathbf{a} \ , \tag{1}$$

where $\tilde{\mathbf{x}}$ denotes the approximation of $\hat{\mathbf{x}}$. The entire set of images $\hat{\mathbf{X}}$ can thus be represented as $\tilde{\mathbf{X}} = \mathbf{U}\mathbf{A}$ where $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_N] \in \mathbb{R}^{k \times N}$ consists of coefficient vectors $\mathbf{a}_i = [a_{1i}, \ldots, a_{ki}]^\top \in \mathbb{R}^k$. A coefficient vector characterizing an image is traditionally calculated by a standard projection of the input image into the eigenspace $\mathbf{a} = \mathbf{U}^\top \hat{\mathbf{x}}$, that is by projecting the image to each principal vector:

$$a_i = < \hat{\mathbf{x}}, \mathbf{u}_i > = \sum_{j=1}^{M} u_{ji} \hat{x}_j \ \ , \ \ i = 1 \ldots k \ . \tag{2}$$

The standard formulation of PCA allows neither to treat images or pixels sequentially nor selectively. The calculation of the covariance matrix and the standard projection (2) require that all training images are processed simultaneously and that all pixels are treated equally. In order to achieve a selective influence and enable incremental learning, we will take a different approach to calculating PCA.

3. Weighted PCA

It is well known that PCA minimizes the squared reconstruction error between the input images and the reconstructed images, thus minimizing the following function:

$$\mathcal{E} = \sum_{i=1}^{M} \sum_{j=1}^{N} \left(\hat{x}_{ij} - \sum_{l=1}^{k} u_{il} a_{lj} \right)^2 \ . \tag{3}$$

Therefore, the principal axes and principal components can be obtained as a solution of this minimization problem. The minimization can be performed by iterating the two step procedure where first the coefficients are estimated and then the principal vectors are computed.

Roweis has derived such an algorithm from the probabilistic point of view [11]. He has considered PCA as a limiting case of a linear Gaussian model, when the noise is infinitesimally small and equal in all directions. From this observation he has derived the algorithm for calculating principal axes, which is based on the EM (expectation-maximization) algorithm. This algorithm consists of two steps, E and M, which are sequentially and iteratively executed:

- **E-step**: $\mathbf{A} = (\mathbf{U}^\top \mathbf{U})^{-1} \mathbf{U}^\top \hat{\mathbf{X}}$
- **M-step**: $\mathbf{U} = \hat{\mathbf{X}} \mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1}$.

The initial solution for principal axes can be determined using some other method or simply by setting the elements of $\mathbf{U}$ to random values. At convergence, the columns of $\mathbf{U}$ span the space of the first k principal axes, minimizing Eq. (3). These vectors are, in general, not orthogonal, but, when desired, they can be orthogonalized later.

Now, we will adapt this algorithm to the weighted minimization by introducing weights into Eq. (3). The weights are composed in the matrix $\mathbf{W} \in \mathbb{R}^{M \times N}$, where w_{ij} is the weight of the i-th pixel in the j-th image. The goal is to minimize the *weighted* squared reconstruction error

$$\mathcal{E} = \sum_{i=1}^{M} \sum_{j=1}^{N} w_{ij} \left(\hat{x}_{ij} - \sum_{l=1}^{k} u_{il} a_{lj} \right)^2 \ . \tag{4}$$

Here, the values of the matrix $\hat{\mathbf{X}}$ are obtained by subtracting the *weighted* mean vector $\boldsymbol{\mu}$ from the training images $\mathbf{x}_i$. The elements of the mean vector are calculated as

$$\mu_i = \frac{\sum_{j=1}^{N} w_{ij} x_{ij}}{\sum_{j=1}^{N} w_{ij}} \ \ , \ \ i = 1 \ldots M \ . \tag{5}$$

Since Eq. (1) holds for each image pixel $\hat{x}_i$, the coefficients a_j can also be computed by solving an overconstrained system of linear equations

$$\hat{x}_i = \sum_{j=1}^{k} a_j u_{ij} \ \ , \ \ i = 1 \ldots M \ , \tag{6}$$

in a least-squares sense, where $\hat{x}_i$ and u_{ij} are known and a_j unknown variables [8, 14]. The E-step of the described EM algorithm can also be replaced by solving the system of linear equations (6). This can be observed by noting that $\mathbf{a}_i = (\mathbf{U}^\top \mathbf{U})^{-1} \mathbf{U}^\top \hat{\mathbf{x}}_i = \mathbf{U}^\dagger \hat{\mathbf{x}}_i$. The least squares solution of (6) is equivalent to the pseudo-inverse. A very similar observation holds also for the M-step. Therefore, by considering weights, we can perform the EM-algorithm by iteratively solving the following systems of linear equations:

- **E-step**: Estimate coefficients in $\mathbf{A}$ in the following way: For each image j, $j = 1 \dots N$, solve the following system of linear equations in the least squares sense:

$$\sqrt{w_{ij}}\hat{x}_{ij} = \sqrt{w_{ij}} \sum_{p=1}^{k} u_{ip} a_{pj} \quad , \quad i = 1 \dots M \quad . \tag{7}$$

- **M-step**: Estimate principal axes in $\mathbf{U}$ in the following way: For each pixel i, $i = 1 \dots M$, solve the following system of linear equations in the least squares sense:

$$\sqrt{w_{ij}}\hat{x}_{ij} = \sqrt{w_{ij}} \sum_{p=1}^{k} u_{ip} a_{pj} \quad , \quad j = 1 \dots N \quad . \tag{8}$$

Using this approach, every pixel can have a different influence on the estimation of the principal subspace.

In practice, it is useful to deal with two types of weights: *temporal* weights ${}^{t}\mathbf{w} \in \mathbb{R}^N$, which put different weights on individual images and *spatial* weights ${}^{s}\mathbf{w} \in \mathbb{R}^M$, which put different weights on individual pixels within an image[1]. In this case, the weight matrix $\mathbf{W}$ is formed in the following way: $\mathbf{W} = [{}^{t}w_1 {}^{s}\mathbf{w}_1, \dots, {}^{t}w_N {}^{s}\mathbf{w}_N]$.

Temporal weights determine how important for estimation of principal subspace the individual images are. For instance, if a training image has weight 2, while all other images have weight 1, the result of the proposed algorithm is the same as the result of the standard algorithm, which has two copies of the particular image in the training set.

Spatial weights control the influence of individual pixels within an image. Therefore, if a part of an image is non-reliable or not important for estimation of the principal subspace, its influence can be diminished by decreasing its weight. In this case, we would also like to have a weighted method for estimating the coefficients. By introducing weights in (6), we can calculate coefficients by solving the following system of linear equations:

$$\sqrt{{}^{s}w_i}x_i = \sqrt{{}^{s}w_i} \sum_{j=1}^{k} a_j u_{ij} \quad , \quad i = 1 \dots M \quad . \tag{9}$$

By putting different weights on different pixels, we can achieve selective influence of pixel values to the estimation of coefficients.

We will demonstrate the behavior of the proposed algorithm on a simple 2-D example. We generated 41 2-D points shown as black dots in Fig. 1. The goal is to estimate 1-D principal subspace: one principal axis and one coefficient for each input point.

We put different weights on the training points. We set temporal weights to ${}^{t}w_j = j^2$ meaning that the points at the end of the training sequence should have a stronger influence on the estimation of the principal subspace. Consequently, the reconstructions of these points should be better. Fig. 1 depicts the principal axis estimated using the standard and the weighted PCA. The weighted mean vector is closer to the end of the point sequence, since the weights of these points have higher values. The principal axis is oriented in such a direction, that enables superior reconstruction of these points. Consequently, the mean squared reconstruction error in the points at the end of the point sequence is much lower when the weighted PCA is used, which results in a lower weighted reconstruction error.

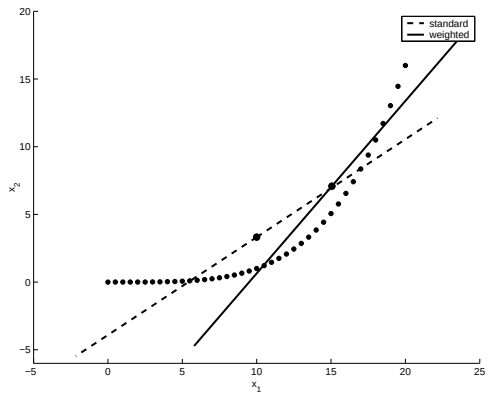


Figure 1. Principal axis obtained using temporal weights.

4. Incremental PCA

In this section we propose a method for incremental learning. It takes the training images sequentially and computes the new eigenspace from the subspace obtained in the previous step and the current input image.

Let us suppose that we have already built an eigenspace from the first n images. In the step $n+1$ we could calculate a new eigenspace from the *reconstructions* of the first n input images and a new image using the standard batch method. The computational complexity of such an algorithm would be prohibitive, since at each step we would have to perform the batch PCA on a set of high-dimensional data. However, identical results can be obtained by using low-dimensional *coefficients* of the first n input images instead of their high-dimensional reconstructions, since coefficients and reconstructed images encompass the

[1]The left superscript is used to distinguish between temporal (${}^{t}\mathbf{w}$) and spatial (${}^{s}\mathbf{w}$) weights.

same visual variability, i.e., they are the same points represented in different coordinate frames. Since the dimension of the eigenspace is small, this algorithm is computationally very efficient.

In practice, this algorithm is realized in the following way. First, a novel image is projected to the current eigenspace and reconstructed. The difference between the input image and its reconstruction is orthogonal to the current eigenspace, therefore if the current eigenspace is enlarged with the residual vector, a new basis is obtained. In this basis all the points from the current eigenspace and the new image can be represented without any loss. Now, the batch PCA is performed on these points in the low-dimensional space and the principal axes are obtained. In order to retain the dimension of the eigenspace, the least significant basis vector can be discarded [1, 6]. All the low dimensional points are then re-projected to the new eigenspace. Finally, the current basis vectors, expressed in the image space coordinates, are rotated to match the new basis vectors expressed in the subspace coordinates, and the mean vector is updated.

The summarized algorithm for updating an eigenspace looks as follows:

Input:
current mean value $\boldsymbol{\mu}^{(n)}$, current eigenvectors $\mathbf{U}^{(n)}$, current coefficients $\mathbf{A}^{(n)}$, new input image $\mathbf{x}$. [2]

Output:
new mean value $\boldsymbol{\mu}^{(n+1)}$, new eigenvectors $\mathbf{U}^{(n+1)}$, new coefficients $\mathbf{A}^{(n+1)}$, new eigenvalues $\mathbf{l}^{(n+1)}$.

Algorithm:

1: Project a new image $\mathbf{x}$ in the current eigenspace: $\mathbf{a} = \mathbf{U}^{(n)\top}(\mathbf{x} - \boldsymbol{\mu}^{(n)})$.
2: Reconstruct the new image: $\mathbf{y} = \mathbf{U}^{(n)}\mathbf{a} + \boldsymbol{\mu}^{(n)}$.
3: Compute the residual vector: $\mathbf{r} = \mathbf{x} - \mathbf{y}$. $\mathbf{r}$ is orthogonal to $\mathbf{U}^{(n)}$.
4: Append $\mathbf{r}$ as a new basis vector: $\mathbf{U}' = \begin{bmatrix} \mathbf{U}^{(n)} & \frac{\mathbf{r}}{||\mathbf{r}||} \end{bmatrix}$.
5: Determine the coordinates of the coefficients in the new basis: $\mathbf{A}' = \begin{bmatrix} \mathbf{A}^{(n)} & \mathbf{a} \\ \mathbf{0} & ||\mathbf{r}|| \end{bmatrix}$.
6: Perform PCA on $\mathbf{A}'$. Obtain the mean value $\boldsymbol{\mu}''$, the eigenvectors $\mathbf{E}''$, and eigenvalues $\mathbf{l}''$.
7: Optionally drop the least significant vector of the new basis: $k^{(n+1)} = k^{(n)}$, $\mathbf{U}'' = [\mathbf{e}''_1, \ldots, \mathbf{e}''_{k^{(n+1)}}]$.[3]
8: Project the coefficients to the new basis: $\mathbf{A}^{(n+1)} = \mathbf{U}''^{\top}(\mathbf{A}' - \boldsymbol{\mu}''\mathbf{1}_{1\times n+1})$.[4]
9: Rotate the subspace $\mathbf{U}'$ for $\mathbf{U}''$: $\mathbf{U}^{(n+1)} = \mathbf{U}'\mathbf{U}''$.
10: Update the mean: $\boldsymbol{\mu}^{(n+1)} = \boldsymbol{\mu}^{(n)} + \mathbf{U}'\boldsymbol{\mu}''$.
11: New eigenvalues: $\mathbf{l}^{(n+1)} = [l''_1, \ldots, l''_{k^{(n+1)}}]$.

[2] Superscript denotes the step which the data is related to. $\mathbf{U}^{(n)}$ denotes the values of $\mathbf{U}$ at the step n.

[3] By discarding a basis vector we preserve the dimension of the eigenspace.

[4] $\mathbf{1}_{m\times n}$ denotes a matrix of dimension $m \times n$, where every element equals to 1.

The initial values of the mean image, the eigenvectors, and the coefficients can be obtained by applying the batch PCA on a small set of images or simply by setting the first training image as the initial eigenspace by assigning $\boldsymbol{\mu}^{(1)} = \mathbf{x}_1$, $\mathbf{U}^{(1)} = \mathbf{0}_{M\times 1}$, and $\mathbf{A}^{(1)} = 0$.

It is worth noting that this algorithm estimates the identical principal subspace as the method proposed by Hall et al. [5]. However, the subspace is obtained in a different way. A significant advantage of our method is that it is able to treat different images differently, which enables to advance it into a weighted incremental method. Furthermore, our method maintains the low dimensional representations of the previously learnt images throughout the entire learning stage, meaning that each training image can be discarded immediately after the update.

To demonstrate the behavior of the incremental method on our simple 2-D example, the eigenspace is being built incrementally. At each step one point (from the left to the right) is added to the representation and the eigenspace is updated accordingly. Fig. 2 illustrates how the eigenspace evolves during this process. The principal axis, obtained at every sixth step, is depicted. The points, which were appended to the model at these steps, are marked with crosses. One can observe, how the origin of the eigenspace (depicted as a square) and the orientation of the principal axis change through time, adapting to the new points, which come into the process. In the end, the estimated eigenspace, which encompasses all training points, is almost equal to the eigenspace obtained using the batch method.

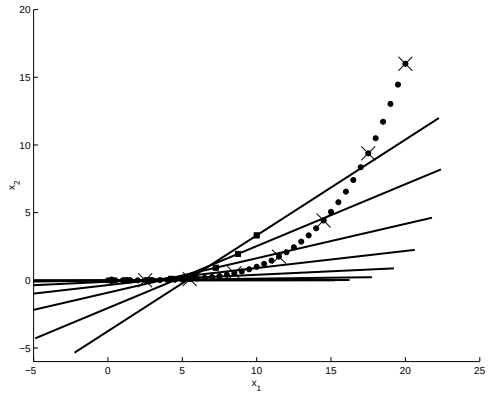


Figure 2. Incremental learning.

5. Weighted incremental PCA

After we have introduced the weighted PCA and the incremental PCA, we combine both techniques into a unified, weighted and incremental approach.

It is quite straightforward to incorporate temporal weights into the incremental algorithm. The core of this algorithm is still the standard batch PCA on low

dimensional data (step 6 of the algorithm). We can replace this standard PCA with the weighted algorithm, which considers temporal weights. This is feasible, because our incremental algorithm maintains the low dimensional coefficients of all input images throughout the proces of the incremental learning (in contrast to some other incremental approaches [5, 9]). Therefore, the representation of each image can be arbitrarily weighted at each update.

Incorporating spatial weights into the process of incremental learning is more complex. After the current eigenspace is updated with a new input image, this image is discarded. Therefore, in the later stages we can not associate weights to individual pixels. This can be done only during the update. For that reason we define spatial weights in a different manner.

Let us assume, that the weights range from 0 to 1. If a weight is set to 1, it means that the corresponding pixel is fully reliable and should be used as is. If a weight is set to 0, it means that the value of the corresponding pixel is completely useless and it is not related to the correct value. We can recover an approximate value of this pixel by considering the knowledge acquired from the previous images. By setting the weight between 0 and 1, we can balance between the influence of the value yielded by the current model and the influence of the pixel value of the input image.

We can achieve this by adding a preprocessing step to the update algorithm. First we calculate the coefficients of the new image $\mathbf{x}$ by using the weighted method (9) considering larger influence of the pixels with higher weights. By reconstructing the coefficients we obtain the reconstructed image $\tilde{\mathbf{x}}$ which contains pixel values yielded by the current model. By blending images $\mathbf{x}$ and $\tilde{\mathbf{x}}$ considering spatial weights by using the following equation

$$x_i^{new} = {}^s w_i x_i + (1 - {}^s w_i)\tilde{x}_i \ , \ i = 1 \dots M \ , \quad (10)$$

we obtain the image which is then used for updating the current eigenspace. In this way, a selective influence of pixels is enabled also in the incremental framework.

A potential drawback of incremental methods is a propagation of errors, since images are added sequentially. In the case of the weighted incremental algorithm this could be problematic if in the early stages of the learning process many images contain significant number of pixels with small weights. In this case the algorithm fails to correctly recover the values of these pixels because the model is too flexible and it can not predict the correct values. When the model encompasses a sufficient number of views it becomes more stable and this is no longer a problem [15].

Fig. 3 depicts the evolution of the eigenspace when the temporal weights are introduced into our simple 2-D example. As in one of the previous examples, we set the weights to ${}^t w_j = j^2$. By comparing Fig. 3 with Fig. 2 it is evident how the points at the end of the point sequence have larger influence to the estimation of the principal axis. At the end of the sequence, the estimated eigenspace is again very similar to the eigenspace obtained using the batch weighted method.

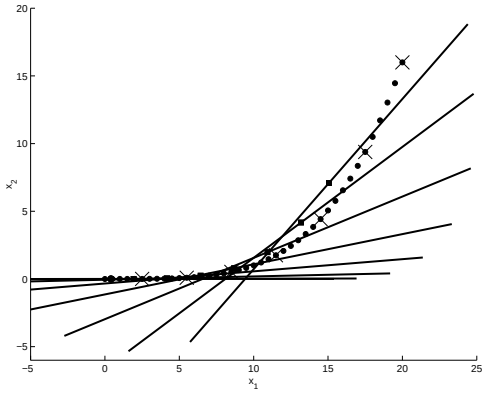


Figure 3. Weighted incremental learning.

6. Experimental results

We performed several experiments to evaluate the performance of the proposed methods on the images from the COIL20 database and on the panoramic images taken by the omnidirectional sensor mounted on the mobile robot.

First we will demonstrate the performance of the method for *weighted learning* using *temporal weights*. Imagine that we want to build the subspace representation of the 'duck' object from the COIL20 database. If we assume that it is more likely to see the duck from the front than from the back it is reasonable to model the front views more accurately [10]. Therefore, we put higher weights on the images of the front (and side) views of the object and built the eigenspace using the batch weighted method. Fig. 4 shows the images reconstructed by the standard method and by the weighted method. One can observe that the first three images are reconstructed worse with the weighted method, since they represent the rear views of the duck where the weights were low (Fig. 4(c)). In contrast, the front (and side) views of the duck are reconstructed much better than with the standard approach. Fig. 5 confirms this finding. It shows the reconstruction errors in the images for both methods. To increase clarity, the reconstruction error was smoothed over the consecutive images reducing local deviations. One can observe that in the second half of the image sequence the reconstruction error produced by the proposed weighted method is lower. Consequently, the weighted reconstruction error is smaller when the

weighted method is used (660 for standard vs. 605 for weighted method). This is exactly what we wanted to achieve.

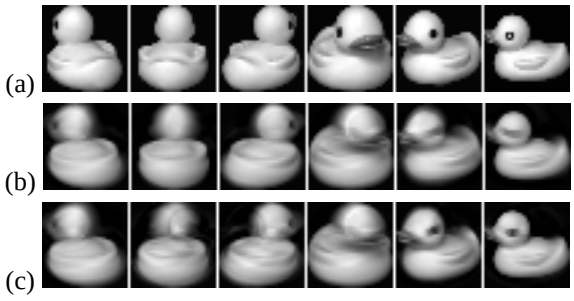


Figure 4. (a) Six images from training set. (b) Reconstruction using standard PCA. (c) Reconstruction using weighted PCA.

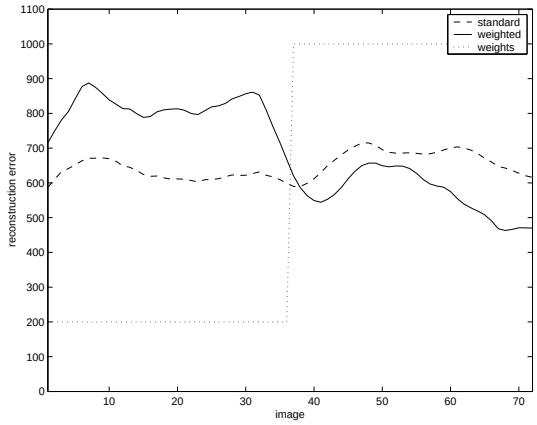


Figure 5. Reconstruction errors of standard and weighted methods.

The performance of the weighted method for estimating coefficients using *spatial weights* is presented in Fig. 6. In this experiment we modelled three objects from the COIL20 database. The training set consisted of 24 images of each of the three objects — altogether 72 images of size 32×32 pixels. Some of the images are shown in Fig. 6(a). Later, we wanted to recognize the object depicted in Fig. 6(b). The left half of this test image represents the 'duck' object, while the right half represents the 'cat' object. If the standard recognition is used, the method can not chose between the 'duck' and the 'cat' and the reconstruction is indistinguishable (Fig. 6(c)). When the weighted method is used and one half of the image is considered more important (left part in Fig. 6(d) and right part in Fig. 6(e)), the method reconstructs very well the 'duck' and the 'cat', respectively. Here, we assume that the information about the weights was signalled by some higher-level cognitive process or by checking the consistency with the other input images.

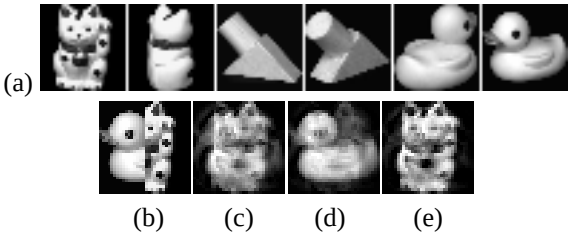


Figure 6. (a) Six images from training set, (b) test image, (c) standard, and (d,e) weighted reconstructions.

Now, we will briefly demonstrate the performance of the *incremental* method. We built eigenspaces of various dimensions from 720 images of all twenty objects from the COIL20 database. Fig. 7 depicts the mean squared reconstruction errors of the images reconstructed from the coefficients obtained by projecting the training images into the eigenspaces, which were built using the batch method (in plots indicated as *batch*) and the proposed incremental method (*incX*). The curve *incA* represents reconstruction errors of images obtained from the coefficients, which were calculated at that time instant, when the particular image was added to the model and then maintained throughout the process of incremental learning. Using this approach, an image can be discarded immediately after the model is updated. As one can observe, the squared reconstruction errors are rather similar (the difference is smaller than 10%), which means that the incremental method is almost as efficient as the batch method.

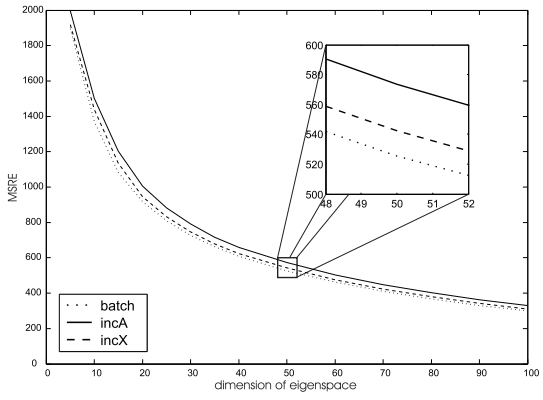


Figure 7. Comparison of standard batch and incremental method.

Then, we built the eigenspace from the same set of images, but with different temporal weights, and used the *incremental weighted* method. We put a weight on each image, which was proportional to the second

power of the image index, giving more influence to the objects and images at the end of the image sequence. The results are depicted in Fig. 8. The reconstruction errors of the incremental weighted method (*WincA*, *WincX*) do not differ significantly from the results of the batch weighted method (*Wbatch*). And certainly, the results of the batch and incremental weighted methods are better than the results of standard methods for images with larger weights. This is also reflected in better weighted squared reconstruction errors (see Table 1).

Table 1. Weighted reconstruction errors of batch and incremental methods.

batch	*incA*	*incX*	*Wbatch*	*WincA*	*WincX*
617	658	648	554	583	565

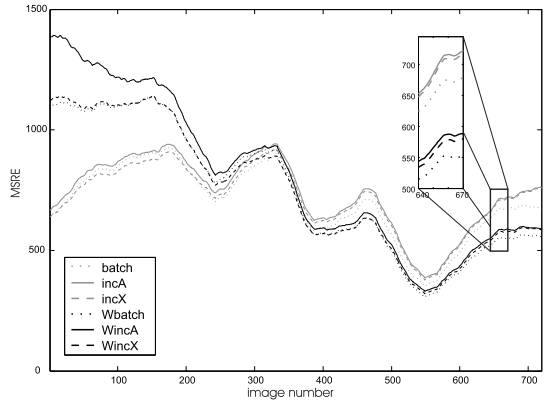


Figure 8. Reconstruction errors of batch and incremental standard and weighted methods.

In both tests of the incremental algorithm, the curves *incX* and *incA* are quite close together, as well as *WincX* and *WincA* (with the exception for the images with very small weights). From this we can conclude that by discarding input images after the eigenspace is updated, we do not degrade results significantly.

Finally, we present the results of the proposed incremental and weighted method applied to improve the results of the *visually-based localization* of a mobile robot [1]. In the training stage, the representation of the environment (our lab in this case) is built from panoramic images taken from several locations. We can simulate the in-plane robot rotation by shifting cylindrical panoramic images and generating spinning images [7]. Three such views of two locations are depicted in Figs. 9(a,b). We thus obtain all necessary views of the environment, which are used for building the representation using PCA. Later, in the localisation stage, a novel image is taken and projected to the eigenspace. The location of the robot is determined by searching for the closest projected training image.

However, due to the construction of the panoramic sensor, not only the environment is captured in the image, but also the holder of the panoramic mirror (the dark vertical bar in Figs. 9(a,b)) and the surface of the robot (lower part of the images). If the robot is oriented differently in the localisation stage, the holder appears in a different position in the image, which makes the test image less similar to the correct training image and the localisation can fail.

The method proposed in this paper offers a solution to this problem. Since we know, that the holder is not a part of the environment, we mask it out during the learning, and learn only the parts which belong to the environment. We can achieve this by using the weighted incremental method and setting the weights of the environment to 1 and weights of the undesirable parts to 0 (see Figs. 9(c)).

In the training stage, the robot was moving from one part of the lab to the other. In the localization stage, the robot returned to the starting position following approximately the same path in the opposite direction. The results are presented in Fig. 10. The gray levels represent coefficient errors; i.e., the distances between the projections of the test images (given in the x axis) and the projections of the training image (y axis). Since the path of the robot was approximately the same as in the training stage, we expect that the coefficient error would be minimal on the diagonal of the error matrix. Since the standard approach incorporated in the representation also the vertical holder, which was in a different image position in the localisation stage, the results of the standard method are not very good (the diagonal of the error matrix in Fig. 10(a) is very non-distinctive). In contrast, the proposed method did not incorporated the holder into the representation of the environment. Consequently, the values around the diagonal in Fig. 10(b) are significantly smaller, which makes the localisation much more accurate and reliable.

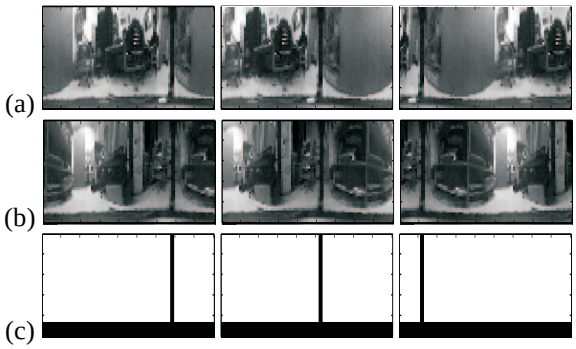
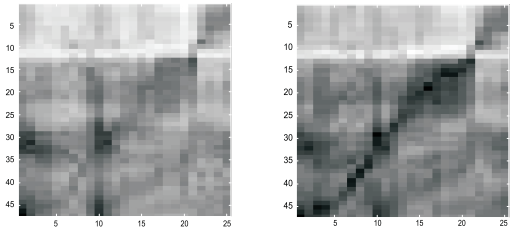


Figure 9. (a,b) Spinning images from two locations. (c) Weights.



(a) (b)

Figure 10. Coefficient errors using (a) standard and (b) weighted learning.

7. Discussion and Conclusions

Before we conclude we will give some additional remarks regarding the computational issues. It is evident, that the proposed weighted method is computationally more demanding than the standard PCA. The time complexity of the standard projection is $\mathcal{O}(mk)$, while the time complexity of the weighted method is $\mathcal{O}(mk^2)$, since it involves solving a sistem of m linear equations with k unknowns. The time complexity of the training weighted method is even higher, since each iteration of the EM algorithm requires $\mathcal{O}(nmk^2)$ for E-step and, similarly, $\mathcal{O}(mnk^2)$ for M-step in general. However, the improvements of the results in the scenarios where the selective treatment of individual images and pixels is required, justifies this extra computational power. In addition, the algorithm could be sped up by using alternative techniques for weighted least squares minimization [3] or by performing SVD on a weighted covariance matrix [13]. Furthermore, the algorithm could also be completely parallelized, since each pixel/image is processed independently.

Weighted PCA is suitable to use in situations when an additional knowledge about significance of individual images as well as separate parts of images is available. If the central part of an image is more informative, then bigger weights should be assigned to these pixels. If the values of some pixels are not defined (e.g. in range images) or are known to be unreliable, their weights should be low. Similarly, selective influence of individual images can be achieved by setting different temporal weights.

In this paper we also proposed the incremental method for weighted subspace learning. The method is suitable for continuous on-line learning, where the model adapts to input images as they arrive (SLAM principle). The algorithm is flexible, since it is able to treat each pixel and each image differently. Therefore, more recent (or more reliable, or more informative, or more noticeable) images can have a stronger influence on the model then others. The principles of short-term and long-term memory, forgetting, and re-learning can be implemented and investigated. This principles are the subject of the ongoing research.

References

[1] M. Artač, M. Jogan, and A. Leonardis. Mobile robot localization using an incremental eigenspace model. In *Proceedings 2002 IEEE International Conference on Robotics and Automation*, pages 1025–1030, Washington DC, USA, May 2002.

[2] S. Chandrasekaran, B. S. Manjunath, Y. F. Wang, J. Winkeler, and H. Zhang. An eigenspace update algorithm for image analysis. *Graphical Models and Image Processing*, 59(5):321–332, September 1997.

[3] F. De la Torre and M. Black. Robust principal component analysis for computer vision. In *ICCV'01*, pages I: 362–369, 2001.

[4] K. Gabriel and S. Zamir. Lower rank approximation of matrices by least squares with any choice of weights. *Technometrics*, 21(21):489–498, 1979.

[5] P. Hall, D. Marshall, and R. Martin. Incremental eigenanalysis for classification. In *British Machine Vision Conference*, volume 1, pages 286–295, September 1998.

[6] P. Hall, D. Marshall, and R. Martin. Merging and spliting eigenspace models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):1042–1048, 2000.

[7] M. Jogan and A. Leonardis. Robust localization using the eigenspace of spinning-images. In *IEEE Workshop on Omnidirectional Vision*, pages 37–44, Hilton Head Island, SC, 2000.

[8] A. Leonardis and H. Bischof. Robust recognition using eigenimages. *Computer Vision and Image Understanding*, 78:99–118, 2000.

[9] X. Liu and T. Chen. Shot boundary detection using temporal statistics modeling. In *ICASSP 2002*, Orlando, FL, USA, May 2002.

[10] D. Perret, M. W. Oram, and E. Ashbridge. Evidence accumulation in cell popoulations responsive to faces: an account of generalisation of recognition withouth mental transformations. In *Object recognition in man, monkey, and machine*, pages 111–145. MIT/Elsevier, 1998.

[11] S. Roweis. EM algorithms for PCA and SPCA. In *Neural Information Processing Systems 10 (NIPS'97)*, pages 626–632, 1997.

[12] H. Sidenbladh, F. de la Torre, and M. Black. A framework for modeling the appearance of 3D articulated figures. In *AFGR00*, pages 368–375, 2000.

[13] D. Skočaj. Incremental learning considering temporal weights. Technical report LRV-04-2002, University of Ljubljana, Faculty for computer and information science, August 2002.

[14] D. Skočaj, H. Bischof, and A. Leonardis. A robust PCA algorithm for building representations from panoramic images. In *Computer Vision – ECCV 2002*, volume IV, pages 761–775, May 2002.

[15] D. Skočaj and A. Leonardis. Incremental approach to robust learning of eigenspaces. In *Proceedings of ÖAGM'02*, page in press, September 2002.

[16] T. Wiberg. Computation of principal components when data are missing. In *Proc. Second Symp. Computational Statistics*, pages 229–236, 1976.