Published in proceedings of ICCV'03, part II, pages 1494-1501, 2003.

Weighted and Robust Incremental Method for Subspace Learning

Danijel Skočaj and Aleš Leonardis
Faculty of Computer and Information Science
University of Ljubljana, Slovenia
e-mail: {danijels,alesl}@fri.uni-lj.si

Abstract

Visual learning is expected to be a continuous and robust process, which treats input images and pixels selectively. In this paper we present a method for subspace learning, which takes these considerations into account. We present an incremental method, which sequentially updates the principal subspace considering weighted influence of individual images as well as individual pixels within an image. This approach is further extended to enable determination of consistencies in the input data and imputation of the values in inconsistent pixels using the previously acquired knowledge, resulting in a novel incremental, weighted and robust method for subspace learning.

1. Introduction

In the real world, visual learning is supposed to be a robust and continuous process. All available visual data is not equally important; in the case of occlusions or other undesirable intrusions in the field of view some visual data can even be misleading. A human visual system treats visual data selectively and builds efficient representations of observed objects and scenes even under non-ideal conditions. Furthermore, these representations can afterwards be updated with newly acquired information, thus adapting to the changing world. In this paper we propose a method, which introduces similar principles in the subspace-based machine visual learning and recognition as well.

Visual learning is often approached by the appearance-based modeling of objects and scenes. Models are usually built using principal component analysis (PCA), which in its original form has, however, several shortcomings with respect to the premises mentioned above. PCA-based learning is traditionally performed in a batch mode, thus requiring all training images to be given in advance, which is not admissible in the framework of continuous learning. In this paper we propose an incremental method, which processes images sequentially one by one and updates the representation at each step accordingly. Each image can be discarded immediately after the model is updated, which makes the method perfectly well suited for real on-line scenarios. In contrast with the traditional batch methods, the incremental approach enables efficient estimation of the principal subspace from a large number of training images as well.

In addition, in the standard PCA approach all pixels of an image receive an equal treatment. Also, all training images have equal influence on the estimation of principal axes. In this paper, we present a generalized PCA approach based on the incremental algorithm, which estimates principal axes and principal components considering weighted pixels and images. We further extend this weighted approach into a method for learning from incomplete data, which builds the model of an object even when a part of input data is missing. Furthermore, we also advance this approach into a method for robust incremental learning, which detects inconsistencies in the training images and builds the representations from consistent data only. As a result, the obtained models are more robust and efficient enabling more reliable visual recognition even when the learning conditions are not ideal.

Several methods for incremental PCA have already been proposed in the computer vision community [12, 4, 7, 8]. These methods are mainly used as an incremental substitute for the standard PCA and are designed neither for weighted nor for robust learning. The incremental methods proposed in [10, 11] are tailored for temporally weighted learning allowing newer images to have a larger influence on the estimation of the current subspace than the older ones. The incremental method presented in this paper is more general and enables assigning arbitrary *temporal* and *spatial* weights. The only previously proposed incremental method that explicitly deals with spatial weights is the method for incremental singular value decomposition of data with missing values introduced by Brand [3]. All other previously reported methods for estimation of principal axes in the presence of data with varying reliability and missing data (e.g., [15, 6, 5, 16]) operate in a batch mode. Also the methods that tackle robust learning of eigenspaces by determining the reliability of individual images [19] and pixels [5, 18, 1] operate in a batch mode, processing all training images simultaneously. Furthermore, they are executed in an iterative manner by repeating time consuming procedures on the entire set of training images. Therefore, the processing time is usually very long, and even becomes

prohibitive when the number of training images is large. In this paper we embed the robust approach into the incremental framework by processing training images individually, which enables efficient visual learning from large training sets as well.

The paper is organized as follows. In section 2, we present the basic algorithm for incremental learning. In section 3 we extend this algorithm into a weighted algorithm, which considers temporal and spatial weights. Next, we present a special case of the algorithm, which can handle missing data. This algorithm is then advanced into a robust incremental algorithm, which can detect and discard inconsistencies in the input images. In section 4 we present the experimental results. Finally, we summarize the paper, expose the contributions, and outline some work in progress.

2. 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 *coefficient vectors* of the first n input images instead of their high-dimensional reconstructions, since coefficient vectors and reconstructed images encompass the same visual variability, i.e., they are just represented in different coordinate frames. Since the dimension of the eigenspace is small, this algorithm is computationally very efficient.

The summarized procedure for one update of the current eigenspace is outlined in Algorithm 1 [1]. This algorithm increases the dimension of the subspace by one. After the update, we can discard the least significant principal vector to preserve the dimension of the subspace [2].

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. Alternatively, one can simply set 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 $\mathbf{A}^{(1)} = 0$. In this way, the algorithm is completely incremental, requiring only one image to be available at each time instant.

[1] $\mathbf{U} \in \mathbb{R}^{M\times k}$ denotes a matrix of k M-dimensional principal axes, $\mathbf{A} \in \mathbb{R}^{k\times n}$ is a matrix of n k-dimensional coefficient vectors, $\boldsymbol{\mu} \in \mathbb{R}^M$ is the mean image. Superscript denotes the step which the data is related to ($\mathbf{U}^{(n)}$ denotes the values of $\mathbf{U}$ at the step n). $\mathbf{1}_{m\times n}$ denotes a $m \times n$ matrix of ones.

Algorithm 1 : Incremental PCA

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

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

1: Project a new image $\mathbf{x}$ into 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 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 $\mathbf{U}''$, and the eigenvalues $\boldsymbol{\lambda}''$.
7: Project the coefficient vectors to the new basis: $\mathbf{A}^{(n+1)} = \mathbf{U}''^{\top}(\mathbf{A}' - \boldsymbol{\mu}''\mathbf{1}_{1\times n+1})$.
8: Rotate the subspace $\mathbf{U}'$ for $\mathbf{U}''$: $\mathbf{U}^{(n+1)} = \mathbf{U}'\mathbf{U}''$.
9: Update the mean: $\boldsymbol{\mu}^{(n+1)} = \boldsymbol{\mu}^{(n)} + \mathbf{U}'\boldsymbol{\mu}''$.
10: New eigenvalues: $\boldsymbol{\lambda}^{(n+1)} = \boldsymbol{\lambda}''$.

It is worth noting that this algorithm estimates the identical principal subspace as the method proposed by Hall et al. [7]. However, the subspace is obtained in a different way. In contrast to our method, which between the learning steps passes coefficient vectors of all training images, the Hall's method passes eigenvalues only. While one may consider this as an advantage, since less data is being passed from step to step and calculation of the covariance matrix is faster, this can also be disadvantageous, because the coefficients are not estimated and maintained during the learning process, thus less information is available. Our algorithm calculates the coefficients at that time instant, when the particular image is added to the model, and then maintains their values throughout the process of incremental learning. This is slightly slower, however it has two advantages. The first advantage is, that each image can be discarded immediately after it has been used for updating the subspace. This is very appropriate (and possibly required) for on-line scenarios (eg. navigation of mobile robots with limited memory resources). And finally, since more information is encompassed in the model, our method can be advanced into a method for weighted learning of eigenspaces, which can consider arbitrary temporal weights.

We will demonstrate the behavior of the proposed algorithm on a simple 2-D example. The 2-D input space contains 41 points shown as black dots in Fig. 1. The goal is to estimate 1-D principal subspace, i.e., the first principal

axis. 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. 1 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. At the end, the estimated eigenspace, which encompasses all training points, is almost identical to the eigenspace obtained using the batch method.

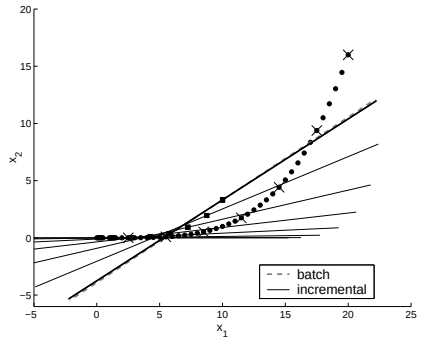

Figure 1: Incremental learning.

3. Weighted and Robust Approach

In order to achieve selective influence of pixels and images, the individual pixels as well as images can be weighted with different weights. In practice, it is useful to deal with two types of weights: *temporal* weights ${}^{\mathbf{t}}\mathbf{w} \in \mathbb{R}^{1\times N}$, which put different weights on individual images and *spatial* weights ${}^{\mathbf{s}}\mathbf{w} \in \mathbb{R}^{M}$, which put different weights on individual pixels within an image[2].

3.1. Temporal Weights

Temporal weights determine how important the individual images are for the estimation of principal subspace. If the temporal weight for one image is higher than the weights for the other images, the reconstruction error of this image should be smaller than the reconstruction errors of the other images. Similarly, the contribution of its principal components to the estimation of the variance should be larger in comparison with that of the other principal components.

From this observation we can derive an algorithm for estimation of the principal subspace considering temporal weights. The principal axes, which maximize the *weighted variance* of the projections of the input images onto the principal axes, can be obtained by eigendecomposition (or, similarly, singular value decomposition) of the *weighted covariance matrix*. If the matrix $\hat{\mathbf{X}} \in \mathbb{R}^{M\times N}$ is composed from N re-scaled input vectors centered around the weighted mean:

$$\hat{\mathbf{x}}_j = \sqrt{{}^{t}w_j}(\mathbf{x}_j - \boldsymbol{\mu}),\ \ j = 1 \dots N\ , \tag{1}$$

the weighted covariance matrix can be calculated as

$$\mathbf{C} = \frac{1}{\sum_{j=1}^{N} {}^{t}w_j}\hat{\mathbf{X}}\hat{\mathbf{X}}^{\top}\ . \tag{2}$$

Using this algorithm, the estimated principal subspace does not depend on all training images equally. For instance, if a training image has the weight 2, while all the other images have the weight 1, the result of this algorithm equals the result of the standard PCA algorithm, which has two copies of the particular image in the training set.

It is quite straightforward to incorporate the temporal weights into the incremental algorithm. The core of this algorithm is still the standard batch PCA on low-dimensional data (step 6 of Algorithm 1). We can replace this standard batch PCA with the weighted algorithm, which considers temporal weights. This is feasible, because our incremental algorithm maintains low-dimensional coefficients of all input images throughout the process of the incremental learning (in contrast with the other incremental approaches). Therefore, the representation of each image can be arbitrarily weighted at each update.

To illustrate the behavior of the proposed algorithm, we put different weights on the training points from our simple 2-D example. We set temporal weights to ${}^{t}w_j = j^2$, which gives a larger influence to the recent points. Fig. 2 depicts the evolution of the eigenspace. By comparing this figure with Fig. 1 it is evident how the weights affect the learning process. At the end of the learning sequence, the weighted mean vector is closer to the points at 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.

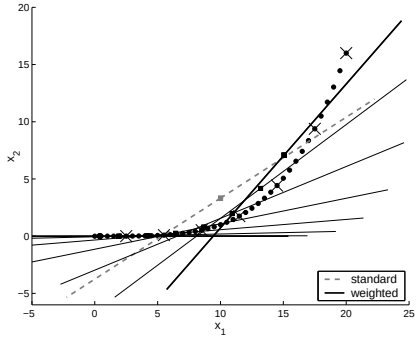

Figure 2: Weighted incremental learning.

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

3.2. Spatial Weights

Spatial weights control the influence of individual pixels within an image. Therefore, if a part of an image is not reliable or important for the estimation of principal components, its influence should be diminished by decreasing the weight of the corresponding pixels.

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 and only its low-dimensional representation is preserved. Therefore, in the later stages we can not associate weights to individual pixels. This can be done only during the update.

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 irrelevant 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. Instead of using the standard projection, the coefficients a_j are obtained by solving an over-determined system of linear equations

$$\sqrt{{}^{s}w_i}x_i = \sqrt{{}^{s}w_i}\sum_{j=1}^{k} a_j u_{ij} \quad , \quad i = 1 \dots M \qquad (3)$$

in the least squares sense. By reconstructing the coefficients we obtain the reconstructed image $\mathbf{y}$ which contains pixel values yielded by the current model. By blending images $\mathbf{x}$ and $\mathbf{y}$, considering spatial weights by using the following equation

$$x_i^{new} = {}^{s}w_i x_i + (1 - {}^{s}w_i) y_i \quad , \quad i = 1 \dots M \quad , \qquad (4)$$

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.

3.3. Missing Pixels

In the real world applications, it is often the case that not all data is available. The values of some pixels are missing or they are totaly non-reliable. Such pixels are referred to as *missing pixels*. Estimation of the principal subspace in the presence of missing pixels can be regarded as a special case of spatially weighted PCA where the weights of missing pixels are set to zero.

The blending step in the algorithm for weighted incremental learning reduces to the imputation of missing pixels. Before the current eigenspace is updated with the new image, the missing pixels have to be optimally filled in. Since not all pixels of an image are known, some coordinates of the corresponding point in the image space are undefined. Thus, the position of the point is constrained to the subspace defined with the values of the known pixels. Given the current principal subspace $\mathbf{U}^{(n)}$, which models the input data seen so far, the optimal location is the point in the missing pixels subspace which is closest to the principal subspace. This point is obtained by filling-in the missing pixels with the reconstructed values, which are calculated from the coefficients estimated from the known pixels only. Since this coefficients reflect the novel information in the new image contained in the known pixels, we may assume that the prediction in the missing pixels will be fine as well. Such an improved image is the best approximation of the correct image that we can obtain from the information contained in the known pixels and in the current eigenspace.

Thus, the new image $\mathbf{x}$ if first projected into the current principal subspace $\mathbf{U}^{(n)}$ by solving a system of linear equations arising from non-missing pixels (3). The obtained coefficient vector $\mathbf{a}$ is then reconstructed and the values in the reconstructed image are used for filling-in the missing pixels. The resulting image is then used for updating the current eigenspace.

A practically equivalent rule for imputation of missing pixels was proposed also by Brand in the context of incremental singular value decomposition [3]. As shown in [3], such a rule for imputation of missing pixels minimizes the distance of the vector representing a new image to the current subspace and maximizes the concentration of the variance in the top singular values. Consequently, such imputation rule minimizes the rank of the updated SVD guaranteeing parsimonious model of the data.

3.4. Robust Approach

The developed method for subspace learning from incomplete data can be further extended in a method for robust learning. In the robust framework the positions of 'bad' pixels are not known in advance, however, we are aware that images may contain outliers. We treat as outliers all pixels, which are not consistent with the information contained in other images. Since at each step we have a current model of the object or scene seen so far, we can detect outliers in the new image and treat them as missing pixels.

This is achieved by projecting the new image into the current eigenspace in a robust manner. Instead of a simple projection, a robust procedure based on subsampling and hypothesize-and-select paradigm is used [9]. Coefficients are obtained mainly from inliers, thus their reconstructions tend to the correct values in outliers as well. Consequently,

the reconstruction error in outliers is large, which makes their detection easier. Therefore, to make the incremental learning robust, we first detect outliers in a new image and replace their values with reconstructed values, which are good approximations of the correct values. Such an improved outlier-free image is then used for updating the eigenspace. Providing that the outliers are detected during the learning process using the robust procedure, the obtained eigenspace is robust as well.

We can refer to this procedure as a 'hard' robust algorithm, since the pixels, which are detected as outliers, are replaced with reconstructed values, while the remaining pixels stay intact. An alternative 'soft' approach is to weight each pixel according to its reliability, which is determined with respect to the reconstruction error. The new image $\mathbf{x}$ is thus projected into the current eigenspace using the simple (and fast) standard projection and the obtained coefficients are used for reconstruction ($\mathbf{y}$). The obtained reconstruction error yields the spatial weights (e.g., ${}^{s}w_i = 1/(|x_i - y_i| + 1)$), which are then used by the weighted algorithm to update the current principal subspace.

To demonstrate the behavior of the robust incremental algorithm, we significantly changed the values of the second coordinate of five points in our 2-D example. Fig. 3 shows that when the non-robust incremental method is used, these outlying points pull the origin in a wrong direction and incorrectly orient the estimated principal axis. On the other hand, the robust method sequentially detects the outlying coordinate values, replaces these values with their reconstructions (shown as circles) and updates the eigenspace accordingly. At the end, the principal axis obtained using this approach is very close to the optimal one.

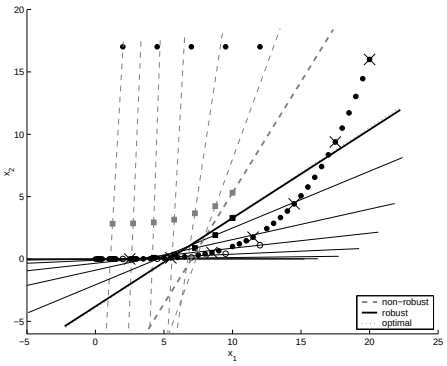

Figure 3: Robust incremental learning.

An important advantage of such *incremental* method is that it processes only one image at each step, while the iterative batch robust methods process all images at each iteration. For that reason, the incremental method is significantly faster and enables robust learning from a large number of training images. Since the model is being incrementally updated with new images, this method is very suitable for on-line applications as well.

On the other hand, it suffers (like all incremental methods) from a potential danger of error propagation. If the initial eigenspaces, built in the early stages of the learning process, encompass only a limited number of appearances of an object or a scene, then all the pixels in the subsequent images, which significantly differ from the appearances of the first images, are considered as outliers and no novel information is added to the model. This particularly holds true for the 'hard' robust version of the updating algorithm. Therefore, the initial eigenspace, which is built in the beginning of the learning process, should be reliable and stable. It should roughly model heterogeneous appearances of an object or a scene and it should be obtained from a set of pixels containing as few outliers as possible. When the model encompasses a sufficient number of appearances it becomes more stable and this is no longer a problem [17].

4. Experimental Results

We performed various experiments to evaluate the proposed methods. Here we first present the results of the experiments where PCA was used for building representations of faces. These experiments nicely demonstrate the efficiency of the proposed methods. Then we present the results of an experiment, where the proposed methods were applied for background modeling.

4.1. Representations of faces

In the first experiment we used for the testbed the ORL face database [14] consisting of 400 images (10 images of each of 40 subjects), rescaled to the size of 32×32 pixels. The images entered into the learning process sequentially one by one (one image of each of 40 subjects first, then the second image of each subject and so on to the tenth image). Six training images are shown in Fig. 5(a). The goal was to represent all 400 images with just 25 images (eigenfaces) using incremental learning.

First, we present the performance of the incremental method in comparison with the standard batch method. Table 1 shows the mean squared reconstruction errors (MSRE) of the images reconstructed from the coefficients obtained by projecting the training images into the eigenspaces, which were built using the batch method (*batch*) and the proposed incremental method (*incX*). The results are very similar; MSRE obtained using the incremental method is only 1% worse. When the coefficients which were obtained and maintained during the learning process are reconstructed, the mean squared reconstruction error (*incA*) is still very similar. In this case, the degradation of the results is 2%. Figs. 5(a-c) show some training images and their reconstructions using the batch and the incremental

(*incA*) method. We can hardly notice any difference between both reconstructed images. The results show that the proposed incremental method is almost as efficient as the batch method (which is optimal in the sense of MSRE).

In the second experiment we put different temporal weights on the training images. Since people's faces are changing through time, we would prefer that the latest images are better represented than the old ones. Such representation would be more appropriate for face recognition, since new images of faces, which will have to be recognized, will be more similar to the last training images than to the first ones. Therefore, we put larger weights on the images at the end of the image sequence. The results are depicted in Fig. 4. For every group of 40 images the mean squared reconstruction error is presented for non-weighted (*incA*, *incX*) and weighted (*WincA*, *WincX*) incremental method. We can observe that the MSRE in the last four groups of images is smaller when the weighted method is used. Thus these images are better represented and their reconstructions are more detailed as can be observed on the last two images in Fig. 4(d). Although the overall mean squared reconstruction error of all training images is bigger, the *weighted* reconstruction error is smaller, as presented in Table 1. This is exactly what we wanted to achieve.

	batch	*incA*	*incX*	*WincA*	*WincX*
MSRE	582	594	587	636	602
WSRE		598	597	570	562

Table 1: Results of batch PCA, IPCA and WIPCA.

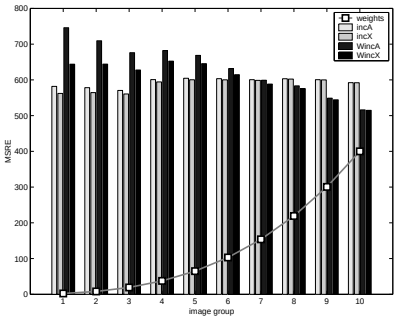

Figure 4: Weighted IPCA.

Then we erased a quarter of each of the last 360 images (Fig. 5(e)) to test the performance of the incremental algorithm which sequentially estimates the values of missing pixels. We left the first 40 images complete in order to make possible to learn at least the initial representations from the complete data. First we calculated the mean image over all non-missing pixels, imputed the missing quarter on each image with the mean values and performed the standard incremental PCA. Then we applied the incremental algorithm for learning from incomplete data, which sequentially imputed missing pixels using the previously acquired knowledge. This method reconstructed the missing quarters significantly better, as can be observed in Figs. 5(f,g). Table 2 shows the overall mean squared reconstruction error between the complete original images and the reconstructed images obtained from the coefficients, which were estimated during the incremental learning. The error obtained using the method which sequentially estimates missing pixels (*robust*) is significantly smaller than when the simple mean-substitution was used (*standard*).

Finally, we occluded each of the last 360 images with a randomly positioned square of a random intensity (Fig. 5(h)). The standard non-robust incremental method included also the squares into the representation, so they appear in the reconstructed images shown in Fig. 5(i) as well. On the other hand, the proposed robust incremental method managed to sequentially detect squares as outliers and reconstruct their values before the update. Therefore, the squares were not included into the representation and the reconstructed images (Fig. 5(j)) look much more similar to the optimal reconstructions shown in Fig. 5(b). Therefore, the robust method significantly outperformed the standard one, as can also be concluded from the mean squared reconstruction errors presented in Table 2.

	standard	*robust*	*optimal*
missing pixels	760	644	594
occlusions	915	710	594

Table 2: Results of standard and robust IPCA.

4.2. Background modeling

The goal of the background modeling is to build a model of the background by detecting and discarding the objects (foreground) in a sequence of images [13, 5]. Due to its incremental nature and simplicity the proposed incremental PCA is very well suited for solving such type of problems.

We performed the experiments on PETS'2001 training sequences[3]. Six images from one sequence are depicted in Fig. 6(a). The goal was to detect pedestrians, cars, and bikers, which are crossing the scene and to adapt the background model accordingly. We built the eigenbackground model consisting of eight eigenvectors. The backgrounds estimated at six time steps of the modelling process (i.e., the reconstructed training images, which were processed in those moments) obtained using three different approaches are presented in Figs. 6(b-d).

First we applied the proposed non-robust incremental method. One could expect that the outliers (pedestrians and cars), which are not consistent with the appearance of the

[3]The images are publicly available on ftp://pets2001.cs.rdg.ac.uk/ .

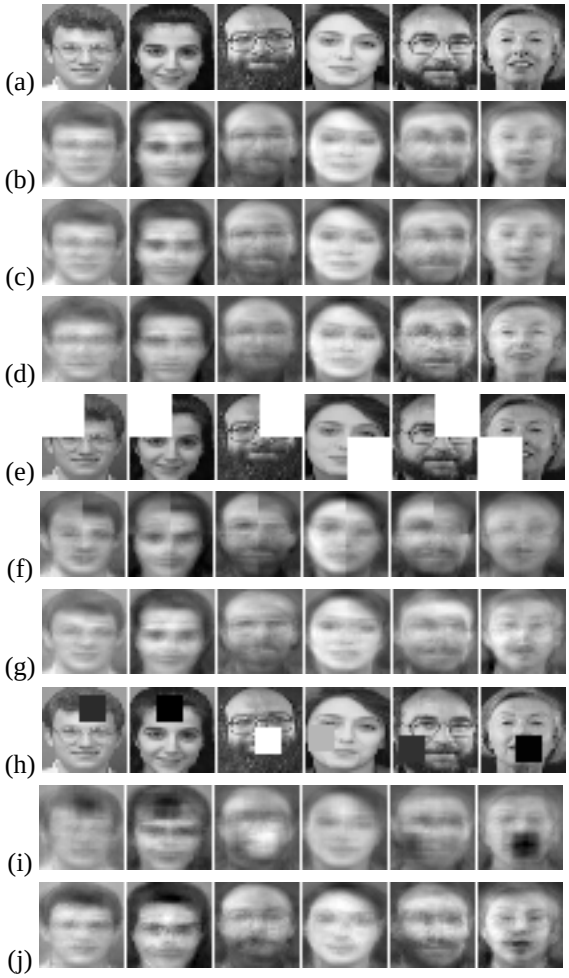

Figure 5: (a) Training images. Reconstructions using (b) batch PCA, (c) IPCA, (d) weighted IPCA. (e) Training images with missing pixels. Reconstructions using (f) mean substitution and standard IPCA, (g) IPCA with reconstruction of missing pixels. (h) Occluded training images. Reconstructions using (i) standard IPCA, (j) robust IPCA.

background, are not modeled within the first principle components and are therefore not included in the subspace representations. However, this is not true in general; one can observe that the cars are still included in the background model in the third and fourth image in Fig. 6(b).

Then we applied the 'hard' robust method. This method successfully detected the pedestrians and cars, reconstructed their values and excluded them from the representation (Fig. 6(c)). In the subsequence of images around the images presented in the third and the fourth columns in Fig. 6 one car leaves the scene and another car parks in the spare lot. This changes are detected as 'foreground' and do not affect the background model. Using the 'hard' robust procedure, the background adapts only to smooth changes, which are not detected as outliers.

If a more flexible model is required, we can use the temporally weighted 'soft' robust method. In this case, the outliers are only down-weighted and are not completely replaced. As a consequence, they are not included in the model, if they appear only for a short period of time, however, if they appear for a longer period, they are gradually incorporated in the model of the background. This is evident from the last two images in Fig. 6(d). The car, which has left the scene is not a part of the background any more, while the new car, which has parked in the spare lot, has been integrated into the current background. In this way, the eigenbackground model can be more adaptive, accommodating to the current appearance of the scene.

5. Conclusion

In this paper we proposed a novel method for weighted and robust incremental learning. The proposed incremental algorithm for PCA has the same general advantages over the batch method as other previously reported incremental approaches: it is significantly faster when the number of training images is high, and it enables updating the current eigenspace and on-line learning. In addition, there are two advantageous features that make our method fundamentally distinct. Firstly, our method maintains the coefficients throughout the process of learning, thus the original images can be discarded immediately after the update. For some applications with a limited amount of memory resources (e.g. mobile platforms, wearable computing) this may be the only option. Using other methods, the images have to be kept in the memory until the end of the learning process, if we want to obtain their representations in the final eigenspace. And secondly, since our method maintains the coefficients of all images, it can be advanced into a weighted method, which considers an arbitrary temporal weight at each image at every step. Furthermore, the proposed weighted method handles also spatial weights, which can be set for each pixel in every image separately. Finally, by adding the robust preprocessing step, the method is suited for visual learning in non-ideal training conditions as well. Due to its incremental nature, this method for robust learning of eigenspaces is significantly faster than previously proposed batch methods.

The method is suitable for continuous on-line learning, where the model adapts to input images as they arrive. 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. These topics are the subject of our ongoing research.

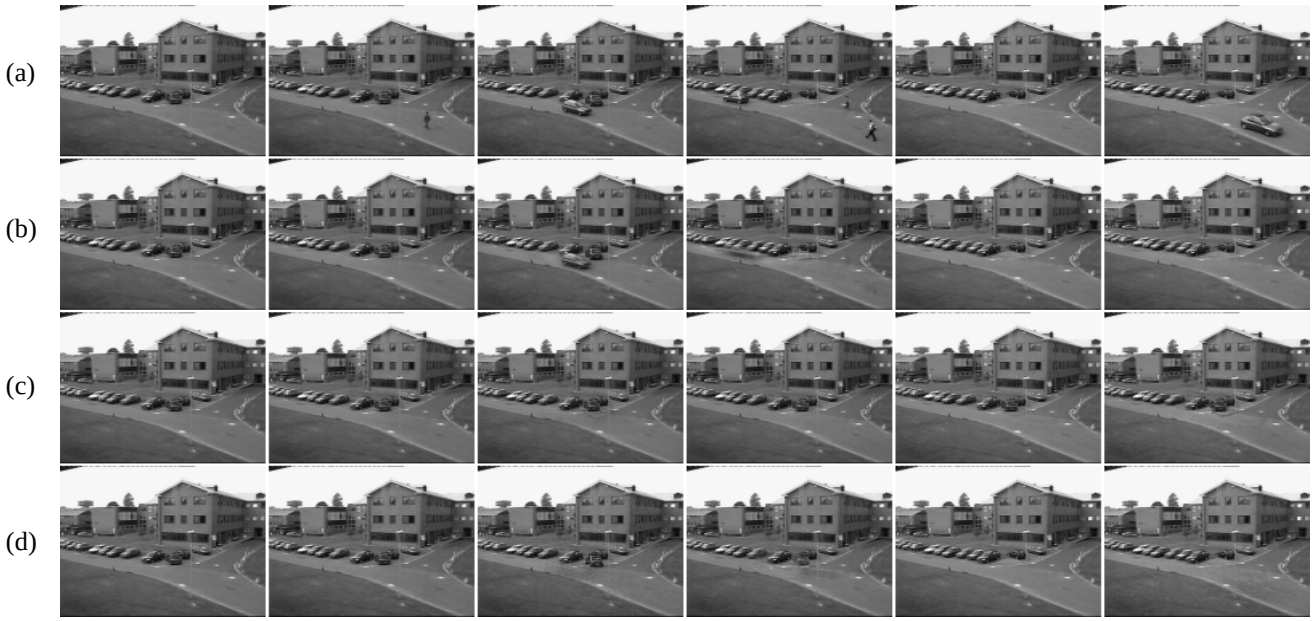

Figure 6: (a) Six input images from PETS'2001 training sequence. Background extracted by (b) non-robust IPCA, (c) 'hard' robust IPCA, and (d) temporally weighted 'soft' robust IPCA.

References

[1] H. Anæs, R. Fisker, K. Åström, and J. M. Carstensen. Robust factorization. *PAMI*, 24(9):1215–1225, September 2002.

[2] M. Artač, M. Jogan, and A. Leonardis. Incremental PCA for on-line visual learning and recognition. In *ICPR 2002*, volume 3, pages 781–784, August 2002.

[3] M. Brand. Incremental singular value decomposition of uncertain data with missing values. In *Computer Vision – ECCV 2002*, volume I, pages 707–720, May 2002.

[4] 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.

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

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

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

[8] P. Hall, D. Marshall, and R. Martin. Merging and spliting eigenspace models. *PAMI*, 22(9):1042–1048, 2000.

[9] A. Leonardis and H. Bischof. Robust recognition using eigenimages. *CVIU*, 78:99–118, 2000.

[10] A. Levy and M. Lindenbaum. Sequential karhunen-loeve basis extraction and its application to images. *IEEE Trans. on Image Processing*, 9:1371–1374, June 2000.

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

[12] H. Murakami and V. Kumar. Efficient calculation of primary images from a set of images. *PAMI*, 4(5):511–515, 1982.

[13] N. M. Oliver, B. Rosario, and A. P. Pentland. A bayesian computer vision system for modeling human interactions. *PAMI*, 22(8):831–843, August 2000.

[14] F. Samaria and A. Harter. Parameterisation of a stochastic model for human face identification. In *2nd IEEE Workshop on Applications of Computer Vision*, December 1994.

[15] H. Shum, K. Ikeuchi, and R. Reddy. Principal component analysis with missing data and its application to polyhedral object modeling. *PAMI*, 17(9):854–867, 1995.

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

[17] D. Skočaj. *Robust subspace approaches to visual learning and recognition*. PhD thesis, University of Ljubljana, 2003.

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

[19] L. Xu and A. Yuille. Robust principal component analysis by self-organizing rules based on statistical physics approach. *IEEE Trans. Neural Networks*, 6(1):131–143, 1995.