On Incremental and Robust Subspace Learning

Yongmin Li, Li-Qun Xu, Jason Morphett and Richard Jacobs
Content and Coding Lab, BT Exact
pp1 MLB3/7, Orion Building, Adastral Park, Ipswich, IP5 3RE, UK
Email: Yongmin.Li@bt.com

Abstract

Principal Component Analysis (PCA) has been of great interest in computer vision and pattern recognition. In particular, *incrementally* learning a PCA model, which is computationally efficient for large scale problems as well as adaptable to reflect the variable state of a dynamic system, is an attractive research topic with numerous applications such as adaptive background modelling and active object recognition. In addition, the conventional PCA, in the sense of least mean squared error minimisation, is susceptible to outlying measurements. To address these two important issues, we present a novel algorithm of incremental PCA, and then extend it to robust PCA. Compared with the previous studies on robust PCA, our algorithm is computationally more efficient. We demonstrate the performance of these algorithms with experimental results on dynamic background modelling and multi-view face modelling.
Keywords Principal Component Analysis (PCA), incremental PCA, robust PCA, background modelling, multi-view face modelling

1 Introduction

Principal Component Analysis (PCA), or the subspace method, has been extensively investigated in the field of computer vision and pattern recognition (Turk and Pentland, 1991; Murase and Nayar, 1994; Moghaddam and Pentland, 1997). One of the attractive characteristics of PCA is that a high dimensional vector can be represented by a small number of orthogonal basis vectors, i.e. the Principal Components. The conventional methods of PCA,

such as Singular Value Decomposition (SVD) and eigen-decomposition, perform in batch-mode with a computational complexity of $O(m^3)$ where m is the minimum value between the data dimension and the number of training examples. Undoubtedly these methods are computationally expensive when dealing with large scale problems where both the dimension and the number of training examples are large. To address this problem, many researchers have been working on *incremental* algorithms. Early work on this topic includes (Gill et al., 1974; Bunch and Nielsen, 1978). Gu and Eisenstat (Gu and Eisenstat, 1994) developed a stable and fast algorithm for SVD which performs in an incremental way by appending a new row to the previous matrix. Chandrasekaran *et al.* (Chandrasekaran et al., 1997) presented an incremental eigenspace update algorithm using SVD. Hall *et al.* (Hall et al., 1998) derived an eigen-decomposition based incremental algorithm. In their extended work, a method for merging and splitting eigenspace models was developed (Hall et al., 2000). Liu and Chen (Liu and Chen, 2002) also introduced an incremental algorithm for PCA model updating and applied it to video shot boundary detection.

In addition, the traditional PCA, in the sense of least mean squared error minimisation, is susceptible to outlying measurements. To build a PCA model which is robust to "outliers", Xu and Yuille (Xu and Yuille, 1995) treated an entire contaminated vector as an outlier by introducing an additional binary variable. Gabriel and Odoroff (Gabriel and Odoroff, 1983) addressed the general case where each element of a vector is assigned with a different weight. More recently, De la Torre and Black (De la Torre and Black, 2001) presented a method of robust subspace learning based on robust M-estimation. Brand (Brand, 2002) also designed a fast incremental SVD algorithm which can deal with missing/untrusted data, however the missing part must be known beforehand.

One limitation of the previous robust PCA methods is that they are usually computationally intensive because the optimisation problem has to be computed *iteratively*[1], e.g. the self-organising algorithms in (Xu and Yuille, 1995), the criss-cross regressions in (Gabriel and Odoroff, 1983) and the Expectation Maximisation algorithm in (De la Torre and Black, 2001). This computational inefficiency restricts their use in many applications, especially when real-time performance is crucial.

[1]It is important to distinguish an *incremental* algorithm from an *iterative* algorithm. The former performs in the manner of prototype growing from training example 1,2, ...to t, the current training example, while the latter iterates on each learning step with all the training examples 1,2, ... and N until a certain stop condition is satisfied. Therefore, for the PCA problem discussed in this paper, the complexity of algorithms in the order from the lowest to highest is: incremental, batch-mode and iterative algorithm.

To address the issue of incremental and robust PCA learning, we present two novel algorithms in this paper: an incremental algorithm for PCA and an incremental algorithm for robust PCA. In both algorithms, the PCA model updating is performed directly from the previous eigenvectors and a new observation vector. The real-time performance can be significantly improved over the traditional batch-mode algorithm. Moreover, in the second algorithm, by introducing a simplified robust analysis scheme, the PCA model is robust to outlying measurements without adding much extra computation (only filtering each element of a new observation with a weight which can be returned from a look-up-table).

The rest of the paper is organised as follows. The new incremental PCA algorithm is introduced in Section 2. It is then extended to robust PCA in Section 3 as a result of adding a scheme of robust analysis. Applications of using the above algorithms for adaptive background modelling and multi-view face modelling are described in Section 4 and 5 respectively. Conclusions and discussions are presented in Section 6.

2 Incremental PCA

Note that in this context we use $\boldsymbol{x}$ to denote the mean-normalised observation vector, i.e.

$$\boldsymbol{x} = \boldsymbol{x}' - \boldsymbol{\mu} \tag{1}$$

where $\boldsymbol{x}'$ is the original vector and $\boldsymbol{\mu}$ is the current mean vector. For a new $\boldsymbol{x}$, if we assume the updating weights on the previous PCA model and the current observation vector are α and $1-\alpha$ respectively, the mean vector can be updated as

$$\boldsymbol{\mu}^{new} = \alpha\boldsymbol{\mu} + (1-\alpha)\mathbf{x}' = \boldsymbol{\mu} + (1-\alpha)\boldsymbol{x} \tag{2}$$

Construct $p+1$ vectors from the previous eigenvectors and the current observation vector

$$\boldsymbol{y}_i = \sqrt{\alpha\lambda_i}\boldsymbol{u}_i, \qquad i = 1, 2, ..., p \tag{3}$$

$$\boldsymbol{y}_{p+1} = \sqrt{1-\alpha}\boldsymbol{x} \tag{4}$$

where $\{\mathbf{u}_i\}$ and $\{\lambda_i\}$ are the current eigenvectors and eigenvalues. The PCA updating problem can then be approximated as an eigen-decomposition problem on the $p+1$ vectors. An $n \times (p+1)$ matrix $\mathbf{A}$ can then be defined as

$$\mathbf{A} = [\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_{p+1}] \tag{5}$$

Assume the covariance matrix $\mathbf{C}$ can be approximated by the first p significant eigenvectors and their corresponding eigenvalues,

$$\mathbf{C} \approx \mathbf{U}_{np}\mathbf{\Lambda}_{pp}\mathbf{U}_{np}^{\mathsf{T}} \tag{6}$$

where the columns of $\mathbf{U}_{np}$ are eigenvectors of $\mathbf{C}$, and diagonal matrix $\mathbf{\Lambda}_{pp}$ is comprised of eigenvalues of $\mathbf{C}$. With a new observation $\boldsymbol{x}$, the new covariance matrix is expressed by

$$\begin{aligned} \mathbf{C}^{new} &= \alpha\mathbf{C} + (1-\alpha)\boldsymbol{x}\boldsymbol{x}^{\mathsf{T}} \\ &\approx \alpha\mathbf{U}_{np}\mathbf{\Lambda}_{pp}\mathbf{U}_{np}^{\mathsf{T}} + (1-\alpha)\boldsymbol{x}\boldsymbol{x}^{\mathsf{T}} \\ &= \sum_{i=1}^{p} \alpha\lambda_i \boldsymbol{u}\boldsymbol{u}^{\mathsf{T}} + (1-\alpha)\boldsymbol{x}\boldsymbol{x}^{\mathsf{T}} \end{aligned} \tag{7}$$

Substituting (3), (4) and (5) into (7) gives

$$\mathbf{C}^{new} = \mathbf{A}\mathbf{A}^{\mathsf{T}} \tag{8}$$

Instead of the $n \times n$ matrix $\mathbf{C}^{new}$, we eigen-decompose a smaller $(p+1)\times(p+1)$ matrix $\mathbf{B}$,

$$\mathbf{B} = \mathbf{A}^{\mathsf{T}}\mathbf{A} \tag{9}$$

yielding eigenvectors $\{\boldsymbol{v}_i^{new}\}$ and eigenvalues $\{\lambda_i^{new}\}$ which satisfy

$$\mathbf{B}\boldsymbol{v}_i^{new} = \lambda_i^{new}\boldsymbol{v}_i^{new}, \qquad i = 1, 2, ..., p+1 \tag{10}$$

Left multiplying by $\mathbf{A}$ on both sides and using (9), we have

$$\mathbf{A}\mathbf{A}^{\mathsf{T}}\mathbf{A}\boldsymbol{v}_i^{new} = \lambda_i^{new}\mathbf{A}\boldsymbol{v}_i^{new} \tag{11}$$

Defining

$$\boldsymbol{u}_i^{new} = \mathbf{A}\boldsymbol{v}_i^{new} \tag{12}$$

and then using (8) and (12) in (11) leads to

$$\mathbf{C}^{new}\boldsymbol{u}_i^{new} = \lambda_i^{new}\boldsymbol{u}_i^{new} \tag{13}$$

i.e. $\boldsymbol{u}_i^{new}$ is an eigenvector of $\mathbf{C}^{new}$ with eigenvalue λ_i^{new}.

Algorithm 1 The incremental algorithm of PCA

1: Construct the initial PCA from the first $q(q \geq p)$ observations.
2: **for all** new observation $\boldsymbol{x}$ **do**
3: Update the mean vector (2);
4: Compute $\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_p$ from the previous PCA (3);
5: Compute $\boldsymbol{y}_{p+1}$ (4);
6: Construct matrix $\mathbf{A}$ (5);
7: Compute matrix $\mathbf{B}$ (9);
8: Eigen-decompose $\mathbf{B}$ to obtain eigenvectors $\{\boldsymbol{v}_i^{new}\}$ and eigenvalues $\{\lambda_i^{new}\}$;
9: Compute new eigenvectors $\{\boldsymbol{u}_i^{new}\}$ (12).
10: **end for**

The algorithm is formally presented in Algorithm 1. It is important to note:

1. Incrementally learning a PCA model is a well-studied subject (Gill et al., 1974; Bunch and Nielsen, 1978; Chandrasekaran et al., 1997; Hall et al., 1998; Hall et al., 2000; Liu and Chen, 2002; Brand, 2002). The main difference between the algorithms, including this one, is how to express the covariance matrix incrementally (e.g. Equation (7)) and the formulation of the algorithm. The accuracy of these algorithms is similar because updating is based on approximating the covariance with the current p-ranked model. Also, the speed of these algorithms is similar because they perform in a similar way of eigen-decomposition or SVD on the rank of $(p+1)$. Therefore, there is no need to compare the performance of these algorithms. However, we believe the algorithm as presented in Algorithm 1 is concise and easy to be implemented. Also, it is ready to be extended to the robust PCA which will be discussed in the next section.

2. The actual computation for matrix $\mathbf{B}$ only occurs for the elements of the $(p+1)$th row or the $(p+1)$th column since $\{\mathbf{u}_i\}$ are orthogonal unit vectors, i.e. only the elements on the diagonal and the last row/column of $\mathbf{B}$ have non-zero values.

3. The update rate α determines the weights on the previous information and new information. Like most incremental algorithms, it is application-dependent and has to be chosen experimentally. Also, with this updating scheme, the old information stored in the model decays exponentially over time.

3 Robust PCA

Recall that PCA, in the sense of least squared reconstruction error, is susceptible to contaminated outlying measurement. Several algorithms of robust PCA have been reported to solve this problem, e.g. (Xu and Yuille, 1995; Gabriel and Odoroff, 1983; De la Torre and Black, 2001). However, the limitation of these algorithms is that they mostly perform in an *iterative* way which is computationally intensive.

The reason of having to use an iterative algorithm for robust PCA is that one normally does not know which part of a sample are likely to be outliers. However, if a *prototype* model, which does not need to be perfect, is available for a problem to be solved, it would be much easy to detect the outliers from the data. For example, we can easily pick up a "cat" image as an outlier from a set of human face images because we know what the human faces look like, and for the same reason we can also tell the white blocks in Figure 5 (the first column) are outlying measurements.

Now if we assume that the updated PCA model at each step of an incremental algorithm is good enough to function as this *prototype* model, then we can solve the problem of robust PCA *incrementally* rather than *iteratively*. Based on this idea, we develop the following incremental algorithm of robust PCA.

3.1 Robust PCA with M-Estimation

We define the residual error of a new vector $\mathbf{x}_i$ by

$$\mathbf{r}_i = \mathbf{U}_{np}\mathbf{U}_{np}^{\mathsf{T}}\mathbf{x}_i - \mathbf{x}_i \tag{14}$$

Note that the $\mathbf{U}_{np}$ is defined as in (6) and, again, $\mathbf{x}_i$ is mean-normalised. We know that the conventional non-robust PCA is the solution of a least-squares problem[2]

$$min \sum_i \|\mathbf{r}_i\|^2 = \sum_i \sum_j (r_i^j)^2 \tag{15}$$

Instead of sum-of-squares, the robust M-estimation method (Huber, 1981) seeks to solve the following problem via a robust function $\rho(r)$

$$min \sum_i \sum_j \rho(r_i^j) \tag{16}$$

[2]In this context, we use subscript to denote the index of vectors, and superscript the index of their elements.

Differentiating (16) by θ_k, the parameters to be estimated, i.e. the elements of $\mathbf{U}_{np}$, we have

$$\sum_i \sum_j \psi(r_i^j) \frac{\partial r_i^j}{\partial \theta_k} = 0, \qquad k = 1, 2, ..., np \tag{17}$$

where $\psi(t) = d\rho(t)/dt$ is the influence function. By introducing a weight function

$$w(t) = \frac{\psi(t)}{t} \tag{18}$$

Equation (17) can be written as

$$\sum_i \sum_j w(r_i^j) r_i^j \frac{\partial r_i^j}{\partial \theta_k} = 0, \qquad k = 1, 2, ..., np \tag{19}$$

which can be regarded as the solution of a new least-squares problem if w is fixed at each step of incremental updating

$$min \sum_i \sum_j w(r_i^j)(r_i^j)^2 \tag{20}$$

If we define

$$z_i^j = \sqrt{w(r_i^j)} x_i^j \tag{21}$$

then substituting (14) and (21) into (20) leads to a new eigen-decomposition problem

$$min \sum_i \|\mathbf{U}_{np}\mathbf{U}_{np}^{\mathrm{T}}\mathbf{z}_i - \mathbf{z}_i\|^2 \tag{22}$$

It is important to note that w is a function of the residual error r_i^j which needs to be computed for each individual training vector (subscript i) and each of its elements (superscript j). The former maintains the adaptability of the algorithm, while the latter ensures that the algorithm is robust to every element of a vector.

If we choose the robust function as the Cauchy function

$$\rho(t) = \frac{c^2}{2} log(1 + (\frac{t}{c})^2) \tag{23}$$

where c controls the convexity of the function, then we have the weight function

$$w(t) = \frac{1}{1 + (t/c)^2} \tag{24}$$

Now it seems we arrive at a typical iterative solution to the problem of robust PCA: compute the residual error with the current PCA model (14), evaluate the weight function $w(r_i^j)$ (24), compute $\mathbf{z}_i$ (21), and eigen-decompose (22) to update the PCA model. Obviously an iterative algorithm like this would be computationally expensive. In the rest of this section, we propose an *incremental* algorithm to solve the problem.

3.2 Robust Parameter Updating

One important parameter needs to be determined before performing the algorithm: c in (23,24) which controls the sharpness of the robust function and hence determines the likelihood of a measurement being an outlier. In previous studies, the parameters of a robust function are usually computed at each step in an iterative robust algorithm (Huber, 1981; Hampel et al., 1986) or using Median Absolute Deviation method (De la Torre and Black, 2001). Both methods are computationally expensive. Here we present an approximate method to estimate the parameters of a robust function.

The first step is to estimate σ_j, the standard deviation of the jth element of the observation vectors $\{x_i^j\}$. Assuming that the current PCA model (including its eigenvalues and eigenvectors) is already a robust estimation from an adaptive algorithm, we approximate σ_j with

$$\sigma_j = \max_{i=1}^{p} \sqrt{\lambda_i} |u_i^j| \tag{25}$$

i.e. the maximal projection of the current eigenvectors on the jth dimension (weighted by their corresponding eigenvalues). This is a reasonable approximation if we consider that PCA actually presents the distribution of the original training vectors with a hyper-ellipse in a subspace of the original space and thus the variation in the original dimensions can be approximated by the projections of the ellipse onto the original space.

The next step is to express c, the parameter of (23,24), with

$$c_j = \beta \sigma_j \tag{26}$$

where β is a fixed coefficient, for example, $\beta = 2.3849$ is obtained with the 95% asymptotic efficiency on the normal distribution (Zhang, 1997). β can be set at a higher value for fast model updating, but at the risk of accepting outliers into the model. To our knowledge, there are no ready solutions so far as to estimate the optimal value of coefficient β.

We use an example of background modelling to illustrate the performance of parameter estimation described above. A video sequence of 200 frames is used in this experiment. The conventional PCA is applied to the sequence to

obtain 10 eigenvectors of the background images. The variation σ_j computed using the PCA model by Equation (25) is shown in Figure 1(a). We also compute the pixel variation directly over the whole sequence as shown in (b). Since there is no foreground object appeared in this sequence, we do not need to consider the influence of outliers. Therefore (b) can be regarded as the ground-truth pixel variation of the background image. For a quantitative measurement, we compute the ratio of σ_j by Equation (25) to its ground-truth (subject to a fixed scaling factor for all pixels), and plot the histogram in Figure 1(c). It is noted that

1. the variation computed using the low-dimensional PCA model is a good approximation of the ground-truth, with most ratio values close to 1 as shown in Figure 1(c);

2. the pixels around image edges, valleys and corners normally demonstrate large variation, while those in smooth areas have small variation.

3.3 The Incremental Algorithm of Robust PCA

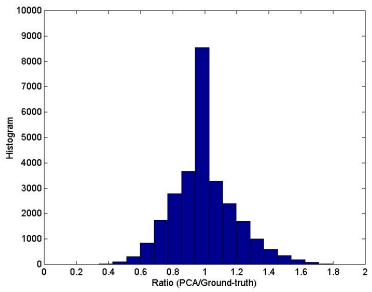
By incorporating the process of robust analysis, we have the incremental algorithm of robust PCA as listed in Algorithm 2. The difference from the non-robust algorithm (Algorithm 1) is that the robust analysis (lines 3-6) has been added and $\boldsymbol{x}$ is replaced by $\boldsymbol{z}$, the weighted vector, in lines 7 and 9. For completeness of description, we include the whole algorithm in Algorithm 2. It is important to note:

1. It is much faster than the conventional batch-mode PCA algorithm for large scale problems, not to mention the iterative robust algorithm;

2. The model can be updated online over time with new observations. This is especially important for modelling dynamic systems where the system state is variable.

3. The extra computation over the non-robust algorithm (Algorithm 1) is only to filter a new observation with a weight function. If the Cauchy function is adopted, this extra computation is reasonably mild. However, even when more intensive computation like exponential and logarithm involved in the weight function w, a look-up-table can be built for the weight item $\sqrt{w(\cdot)}$ in Equation (21) which can remarkably reduce the computation. Note the look-up-table should be indexed by r/c rather than r.

(a) (b)



(c)

Figure 1: Standard deviation of individual pixels σ_j computed from (a) the low-dimensional PCA model (approximated) and (b) the whole image sequence (ground-truth). All values are multiplied by 20 for illustration purpose. Large variation is shown in dark intensity. (c) Histogram of the ratios of approximated σ_j to its ground-truth value.

Algorithm 2 The incremental algorithm of robust PCA

1: Construct the initial PCA from the first $q(q \geq p)$ observations.
2: **for all** new observation $\boldsymbol{x}$ **do**
3: Estimate c_j, the parameter of the robust function, from the current PCA (25,26);
4: Compute the residual error $\mathbf{r}$ (14);
5: Compute the weight $w(r^j)$ for each element of $\boldsymbol{x}$ (24);
6: Compute $\boldsymbol{z}$ (21);
7: Update the mean vector (2), replacing $\boldsymbol{x}$ by $\boldsymbol{z}$;
8: Compute $\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_p$ from the previous PCA (3);
9: Compute $\boldsymbol{y}_{p+1}$ (4), replacing $\boldsymbol{x}$ by $\boldsymbol{z}$;
10: Construct matrix $\mathbf{A}$ (5);
11: Compute matrix $\mathbf{B}$ (9);
12: Eigen-decompose $\mathbf{B}$ to obtain eigenvectors $\{\boldsymbol{v}_i^{new}\}$ and eigenvalues $\{\lambda_i^{new}\}$;
13: Compute new eigenvectors $\{\boldsymbol{u}_i^{new}\}$ (12).
14: **end for**

4 Robust Background Modelling

Modelling background using PCA was firstly proposed by Oliver *et al.* (Oliver et al., 2000). By performing PCA on a sample of N images, the background can be represented by the mean image and the first p significant eigenvectors. Once this model is constructed, one projects an input image into the p dimensional PCA space and reconstruct it from the p dimensional PCA vector. The foreground pixels can then be obtained by computing the difference between the input image and its reconstruction.

Although Oliver *et al.* claimed that this background model can be adapted over time, it is computationally intensive to perform model updating using the conventional PCA. Moreover, without a mechanism of robust analysis, the outliers or foreground objects may be absorbed into the background model. Apparently this is not what we expect.

To address the two problems stated above, we extend PCA background model by introducing (1) the incremental PCA algorithm described in Section 2 and (2) robust analysis of new observations discussed in Section 3.

We applied the algorithms introduced in the previous sections to an image sequence in PET2001 datasets[3]. This sequence was taken from a university site with a length of 3061 frames. There are mainly two kinds of activities happened in the sequence: (1) moving objects, e.g. pedestrians, bicycles and vehicles, and (2) new objects being introduced into or removed from the background. The parameters in the experiments are: image size 192×144 (grey-level), PCA dimension $p = 10$, size of initial training set $q = 20$, update rate $\alpha = 0.95$ and coefficient $\beta = 10$.

4.1 Comparing to the Batch-mode Method

In the first experiment, we compared the performance of our robust algorithm (Algorithm 2) with the conventional batch-mode PCA algorithm. It is infeasible to run the conventional batch-mode PCA algorithm on the same data since they are too big to be fit in the computer memory. We randomly selected 200 frames from the sequence to perform a conventional batch-mode PCA. Then the trained PCA was used as a fixed background model.

Sample results are illustrated in Figure 2 (more results are available in the supplementary video file "pets.mpg"[4]).

It is noted that our algorithm successfully captured the background changes. An interesting example is that, between the 1000th to 1500th frames (the

[3] A benchmark database for video surveillance which can be downloaded at http://www.cvg.cs.rdg.ac.uk/PETS2001/pets2001-dataset.html

[4] Available at http://www.dcs.qmul.ac.uk/∼yongmin/sctv2003/pets.mpg.

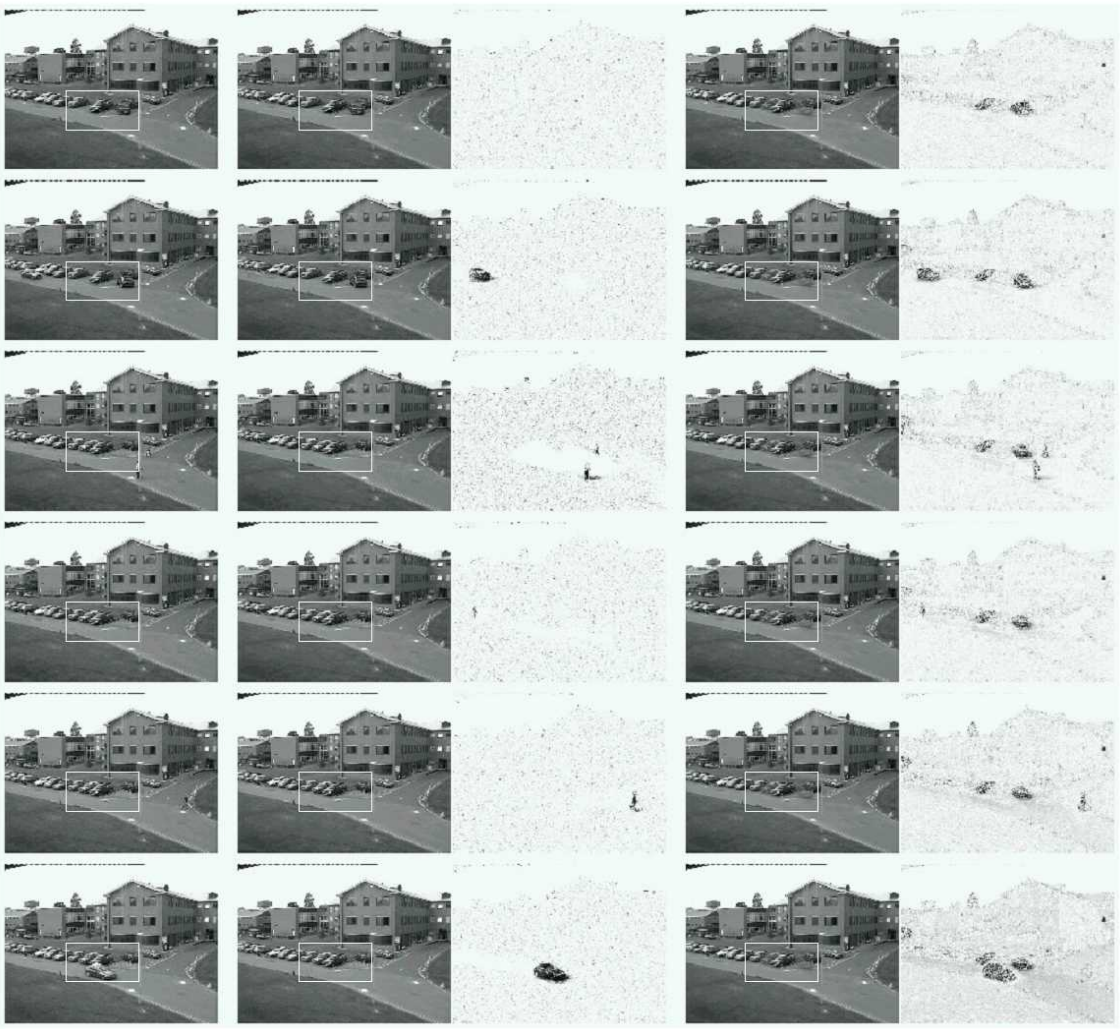


Figure 2: Sample results of background modelling. From left to right are the original input frame, reconstruction and the weights computed by Equation (24) (dark intensity for low weight) of the robust algorithm, and the reconstruction and the absolute difference images (dark intensity for large difference) of the conventional batch-mode algorithm. Results are shown for every 500 frames of the test sequence.

1st and 2nd rows in Figure 2), a car entered into the scene and became part of the background, and another background car left from the scene. The background changes are highlighted by white boxes in the figure. The model was gradually updated to reflect the changes of the background. In

this experiment, the incremental algorithm achieved a frame rate of 5 fps on a 1.5GHz Pentium IV computer (with JPEG image decoding and image displaying). On the other hand, the fixed PCA model failed to capture the dynamic changes of the background. Most noticeably are the ghost effect around the areas of the two cars in the reconstructed images and the false foreground detection.

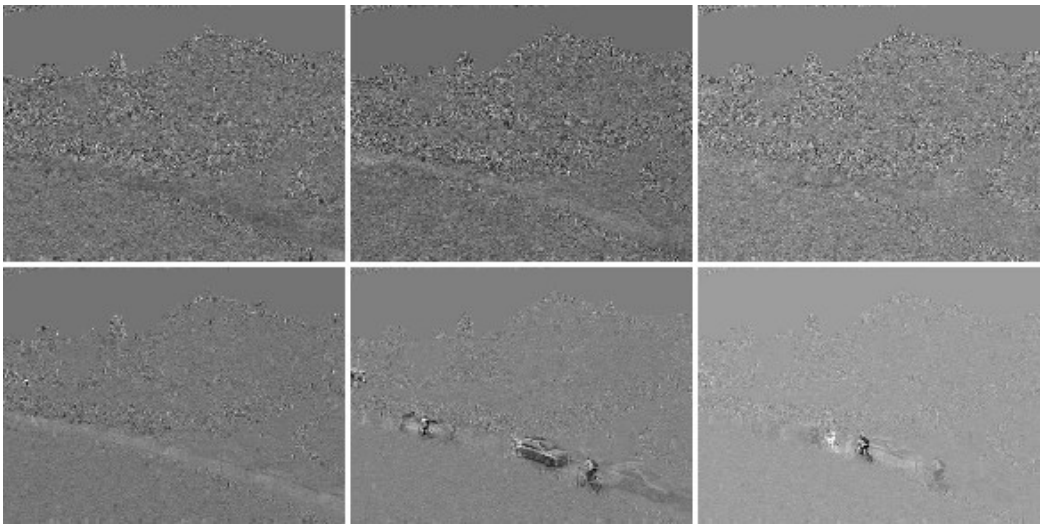
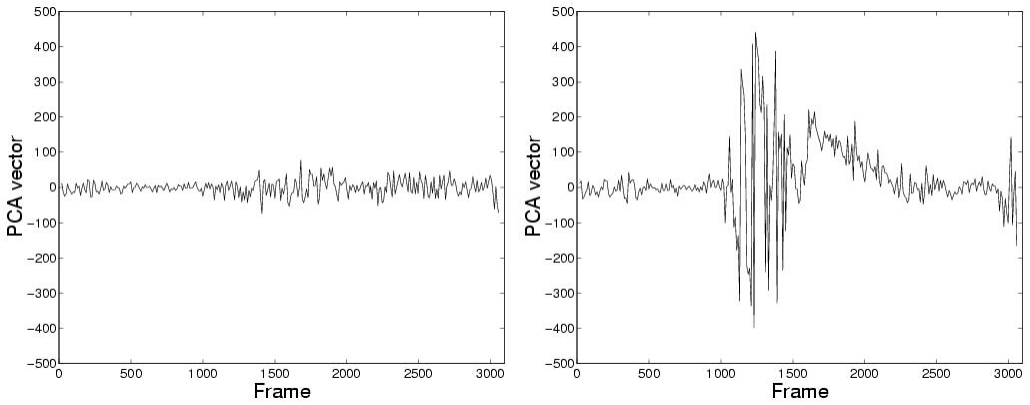


Figure 3: The first three eigenvectors obtained from the robust algorithm (upper row) and non-robust algorithm (lower row). The intensity values have been normalised to $[0, 255]$ for illustration purpose.

4.2 Comparing to the Non-Robust Method

In the second experiment, we compared the performance of the non-robust algorithm (Algorithm 1) and robust algorithm (Algorithm 2). After applying both algorithms to the same sequence used above, we illustrate the first three eigenvectors of each PCA model in Figure 3. It is noted that the non-robust algorithm unfortunately captured the variation of outliers, most noticeably the trace of pedestrians and cars on the walkway appearing in the images of the eigenvectors. This is exactly the limitation of conventional PCA (in the sense of least squared error minimisation) as the outliers usually contribute more to the overall squared error and thus deviate the results from desired. On the other hand, the robust algorithm performed very well: the outliers have been successfully filtered out and the PCA modes generally reflect the

variation of the background only, i.e. greater values for highly textured image positions.



(a) (b)

Figure 4: The first dimension of the PCA vector computed on the same sequence in Figure 2 using the robust algorithm (a) and non-robust algorithm (b).

The importance of applying robust analysis can be further illustrated in Figure 4 which shows the values of the first dimension of the PCA vectors computed with the two algorithms. A PCA vector is a p-vector obtained by projecting a sample vector onto the p eigenvectors of a PCA model. The first dimension of the PCA vector corresponds to the projection to the most significant eigenvector. It is observed that the non-robust algorithm presents a fluctuant result, especially when significant activities happened during frames 1000-1500, while the robust algorithm achieves a steady performance.

Generally, we would expect that a background model (1) should not demonstrate abrupt changes when there are continuous foreground activities involved, and (2) should evolve smoothly when new components being introduced or old components being removed. The results as shown in Figure 4 depict that the robust algorithm performed well in terms of these criteria, while the non-robust algorithm struggled to compensate for the large error from outliers by severely adjusting the values of model parameters.

Figure 5: Sample results of multi-view face modelling. From left to right are: original face image, mean vectors and reconstructions of (1) view-based eigenface method, (2) Algorithm 2, (3) Algorithm 1, and (4) batch-mode PCA, respectively. Results are shown for every 20 frames of the test sequence.

5 Multi-view Face Modelling

Modelling face across multiple views is a challenging problem. One of the difficulties is that the rotation in depth causes the non-linear variation to the 2D image appearance. The well-known eigenface method, which has been successfully applied to frontal face detection and recognition, can hardly provide a satisfactory solution to this problem as the multi-view face images are

largely out of alignment. One possible solution to this problem as presented in (Moghaddam and Pentland, 1997) is to build a set of view-based eigenface models, however, the pose information of the faces need to be known and the division of the view space is often arbitrary and coarse.

In the following experiments we compare the results of four methods: (1) view-based eigenface method (Moghaddam and Pentland, 1997), (2) Algorithm 2 (robust), (3) Algorithm 1 (non-robust), and (4) batch-mode PCA. The image sequences were captured using an electromagnetic tracking system which provides the position of a face in an image and the pose angles of the face. The images are in size of 384×288 pixels and contain faces of about 80×80 pixels. As face detection is beyond the domain of this work, we directly used the cropped face images by the position information provided by the tracking system.

We also added uniformly distributed random noise to the data by generating high-intensity blocks with size of 4-8 pixels at various image positions. Note that the first 20 frames do not contain generated noise in order to obtain a clean initial model for the robust method. We will discuss this issue in the last section.

For method (1), we divide the view space into five segments: left profile, left, frontal, right, and right profile. So the pose information is used additionally for this method. Five view-based PCA models are trained respectively on these segments with the uncontaminated data because we want to use the results of this method as "ground-truth" for comparison. For methods (2) and (3), the algorithms perform incrementally through the sequences. For method (4), the batch-mode PCA is trained from the whole sequence.

The images are scaled to 80×80 pixels. The parameters for the robust method are the same as those in the previous section: $p = 10$, $q = 20$, $\alpha = 0.95$ and $\beta = 10$. Figure 5 shows the results of these methods (more results are available in the supplementary video file "face.mpg"[5]). It is evident that

1. the batch-mode method failed to capture the large variation caused by pose change (most noticeably is the ghost effect of the reconstructions;

2. although the view-based method is trained from clean data and uses extra pose information, the reconstructions are noticeably blurry owing to the coarse segmentation of view space;

3. the non-robust algorithm corrupted quickly owing to the influence of the high-intensity outliers;

[5]Available at http://www.dcs.qmul.ac.uk/∼yongmin/sctv2003/face.mpg.

4. the proposed incremental algorithm of robust PCA performed very well: the outliers have been filtered out and the model has been adapted with respect to the view change.

6 Conclusions

PCA is a widely applied technique in pattern recognition and computer vision. However, the conventional batch-mode PCA suffers from two limitations: computationally intensive and susceptible to outlying measurement. Unfortunately the two issues have only been addressed separately in the previous studies. In this work, we developed a novel incremental PCA algorithm, and extended it to robust PCA.

The main contribution of this paper is the incremental algorithm for robust PCA. In the previous work, the problem of robust PCA is mostly solved by *iterative* algorithms which are computationally expensive. The reason of having to do so is that one does not know what part of a sample are outliers. However, the updated model at each step of an incremental PCA algorithm can be used for outlier detection, i.e. given this "prototype" model, one does not need to go through the expensive iterative process. This is the starting point of our proposed algorithm.

We have provided detailed derivation of the algorithms. Moreover, we have discussed several implementation issues including (1) approximating the standard deviation using the previous eigenvectors and eigenvalues, (2) selection of robust functions, and (3) look-up-table for robust weight computing. These can be helpful to further improve the performance.

Furthermore, we applied the algorithms to the problems of dynamic background modelling and multi-view face modelling. These two applications alone have their own significance: the former extends the static method of PCA background modelling to a dynamic and adaptive method by introducing an incremental and robust model updating scheme, and the latter makes it possible to model faces of large pose variation with a simple, adaptive, model.

Nevertheless, we have experienced problems when the initial PCA model contains significant outliers. Under these circumstances, the assumption (the prototype model is good enough for outlier detection) is broken, and the model would take long time to recover. Although the model can recover more quickly by choosing a smaller update rate α, we argue that the update rate should be determined by applications rather than the robust analysis process. A possible solution to this problem is to learning the initial model using the traditional robust methods. Owing to the small size of initial data,

the performance should not degrade seriously.

References

Brand, M. (2002). Incremental sigular value decomposition of uncertain data with mising values. In *European Conference on Computer Vision*, Copenhagen, Denmark.

Bunch, J. and Nielsen, C. (1978). Updating the singular value decomposition. *Numerische Mathematik*, 31(2):131–152.

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

De la Torre, F. and Black, M. (2001). Robust principal component analysis for computer vision. In *IEEE International Conference on Computer Vision*, volume 1, pages 362–369, Vancouver, Canada.

Gabriel, K. and Odoroff, C. (1983). Resistant lower rank approximation of matrices. In Gentle, J., editor, *Proceedings of the Fifteenth Symposium on the Interface*, pages 304–308, Amsterdam, Netherlands.

Gill, P., Golub, G., Murray, W., and Saunders, M. (1974). Methods for modifying matrix factorizations. *Mathematics of Computation*, 28(26):505–535.

Gu, M. and Eisenstat, S. C. (1994). A fast and stable algorithm for updating the singular value decomposition. Technical report, Department of Computer Science, Yale University. YALEU/DCS/RR-966.

Hall, P. M., Marshall, A. D., and Martin, R. R. (1998). Incremental eigenanalysis for classification. In Lewis, P. H. and Nixon, M. S., editors, *British Machine Vision Conference*, pages 286–295.

Hall, P. M., Marshall, A. D., and Martin, R. R. (2000). Merging and splitting eigenspace models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):1042–1049.

Hampel, F., Ronchetti, E., Rousseeuw, P., and Stahel, W. (1986). *Robust Statistics*. John Wiley & Sons Inc.

Huber, P. J. (1981). *Robust Statistics*. John Wiley & Sons Inc.

Liu, X. and Chen, T. (2002). Shot boundary detection using temporal statistics modeling. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*.

Moghaddam, B. and Pentland, A. (1997). Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):137–143.

Murase, H. and Nayar, S. K. (1994). Illumination planning for object recognition using parametric eigenspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(12):1219–1227.

Oliver, N., Rosario, B., and Pentland, A. (2000). A Bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):831–841.

Turk, M. and Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86.

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

Zhang, Z. (1997). Parameter estimation techniques: A tutorial with application to conic fitting. *Image and Vision Computing*, 15(1):59–76.