Adaptive Face Recognition System Using Fast Incremental Principal Component Analysis

Seiichi Ozawa[1], Shaoning Pang[2], and Nikola Kasabov[2]

[1] Graduate School of Engineering, Kobe University
1-1 Rokko-dai, Nada-ku, Kobe 657-8501, Japan
ozawasei@kobe-u.ac.jp
[2] Knowledge Engineering & Discover Research Institute
Auckland University of Technology, Private Bag 92006,
Auckland 1020, New Zealand
shaoning.pang@aut.ac.nz, nik.kasabov@aut.ac.nz

Abstract. In this paper, a novel face recognition system is presented in which not only a classifier but also a feature space is learned incrementally to adapt to a chunk of incoming training samples. A distinctive feature of the proposed system is that the selection of useful features and the learning of an optimal decision boundary are conducted in an online fashion. In the proposed system, Chunk Incremental Principal Component Analysis (CIPCA) and Resource Allocating Network with Long-Term Memory are effectively combined. In the experiments, the proposed face recognition system is evaluated for a self-compiled face image database. The experimental results demonstrate that the test performance of the proposed system is consistently improved over the learning stages, and that the learning speed of a feature space is greatly enhanced by CIPCA.

1 Introduction

In general, the information processing in face recognition systems is composed of the two parts: feature selection and classifier. This means that when constructing an adaptive recognition system, we should consider two types of incremental learning: one is the incremental learning of a feature space and the other is that of a classifier. As for the feature selection, Hall and Martin [2] have proposed a method to update eigenvectors and eigenvalues in an incremental way called Incremental Principal Component Analysis (IPCA). Recently, to enhance the learning efficiency, we have proposed the two extended algorithms for the original IPCA. One is an extended IPCA in which the eigen-axis augmentation is carried out based on the accumulation ratio instead of the norm of a residue vector [5], and the other is Chunk IPCA in which a chunk of training samples are trained at a time [3]. On the other hand, we also have proposed an incremental learning algorithm for a neural classifier called Resource Allocating Neural Network with Long-Term Memory (RAN-LTM). As we have already presented in [4], IPCA and RAN-LTM are effectively combined to construct a powerful pattern recognition system.

This paper is organized as follows. Section 2 gives a quick review on the two extended IPCA algorithms: IPCA Based on Accumulation Ratio and Chunk IPCA. Then, the face recognition system is briefly explained in Section 3. In Section 4, several experiments are conducted to evaluate the proposed face recognition system. Finally, Section 5 gives a brief summary of this work.

2 Incremental Principal Component Analysis (IPCA)

2.1 IPCA Based on Accumulation Ratio

Assume that N training samples $\boldsymbol{x}_i \in \mathcal{R}^n$ $(i = 1, \cdots, N)$ have been presented so far, and an eigenspace model $\Omega = (\bar{\boldsymbol{x}}, \boldsymbol{U}_k, \boldsymbol{\Lambda}_k, N)$, where $\bar{\boldsymbol{x}} \in \mathcal{R}^n$ is a mean vector, $\boldsymbol{U}_k$ is an $n \times k$ matrix whose column vectors correspond to the eigenvectors, and $\boldsymbol{\Lambda}_k$ is a $k \times k$ matrix whose diagonal elements correspond to the eigenvalues. Here, k is the number of eigen-axes spanning the eigenspace (i.e., eigenspace dimensionality).

Now, assume that the $(N+1)$th training sample $\boldsymbol{y} \in \mathcal{R}^n$ is given. The addition of this new sample will lead to the changes in both mean vector and covariance matrix; therefore, the eigenvectors and eigenvalues should also be updated. The new mean input vector $\bar{\boldsymbol{x}}'$ is easily obtained as follows:

$$\bar{\boldsymbol{x}}' = \frac{1}{N+1}(N\bar{\boldsymbol{x}} + \boldsymbol{y}) \ \in \mathcal{R}^n. \tag{1}$$

To update $\boldsymbol{U}_k$ and $\boldsymbol{\Lambda}_k$, first we need to check if the eigenspace should be enlarged in terms of dimensionality. If almost all energy of the new sample is included in the current eigenspace, there is no need to increase the dimensionality. However, if a certain quantity of energy is included in the complementary eigenspace, the dimensional augmentation is needed, or crucial information on the new sample might be lost. In the original IPCA [2], the determination of the eigenspace augmentation is made based on the norm of the following residue vector $\boldsymbol{h} \in \mathcal{R}^n$:

$$\boldsymbol{h} = (\boldsymbol{y} - \bar{\boldsymbol{x}}) - \boldsymbol{U}_k\boldsymbol{g} \tag{2}$$

where $\boldsymbol{g} = \boldsymbol{U}_k^T(\boldsymbol{y} - \bar{\boldsymbol{x}})$, and T means the transpose of vectors and matrices. However, the threshold for the norm generally depends on datasets; therefore, we have proposed an extended IPCA algorithm [5] in which the accumulation ratio is used instead of the norm as a criterion to determine the dimensional augmentation.

In [5], we have shown that the accumulation ratio can be updated incrementally by

$$A'(\boldsymbol{U}_k) = \frac{N(N+1)\sum_{i=1}^{k}\lambda_i + N\|\boldsymbol{U}_k^T(\boldsymbol{y} - \bar{\boldsymbol{x}})\|^2}{N(N+1)\sum_{i=1}^{n}\lambda_i + N\|\boldsymbol{y} - \bar{\boldsymbol{x}}\|^2} \tag{3}$$

where λ_i is the ith eigenvalue corresponding to the ith diagonal element of $\boldsymbol{\Lambda}_k$.

Note that no past samples are necessary for the incremental update of $A'(\boldsymbol{U}_k)$.

It has been shown that the eigenvectors and eigenvalues are updated by solving the following intermediate eigenproblem [2]:

$$\left(\frac{N}{N+1}\begin{bmatrix}\boldsymbol{\Lambda}_k & \boldsymbol{0}\\ \boldsymbol{0}^T & 0\end{bmatrix} + \frac{N}{(N+1)^2}\begin{bmatrix}\boldsymbol{g}\boldsymbol{g}^T & \gamma\boldsymbol{g}\\ \gamma\boldsymbol{g}^T & \gamma^2\end{bmatrix}\right)\boldsymbol{R} = \boldsymbol{R}\boldsymbol{\Lambda}'_{k+1} \tag{4}$$

where $\gamma = \tilde{\boldsymbol{h}}^T(\boldsymbol{y} - \bar{\boldsymbol{x}})$, $\boldsymbol{R}$ is a $(k+1) \times (k+1)$ matrix whose column vectors correspond to the eigenvectors obtained from the above intermediate eigenproblem, $\boldsymbol{\Lambda}'_{k+1}$ is the new eigenvalue matrix, and $\boldsymbol{0}$ is a k-dimensional zero vector. Using the solution $\boldsymbol{R}$, the new $n \times (k+1)$ eigenvector matrix $\boldsymbol{U}'_{k+1}$ is calculated as follows:

$$\boldsymbol{U}'_{k+1} = [\boldsymbol{U}_k,\ \hat{\boldsymbol{h}}]\boldsymbol{R} \tag{5}$$

where

$$\hat{\boldsymbol{h}} = \begin{cases}\boldsymbol{h}/\|\boldsymbol{h}\| & \text{if } A(\boldsymbol{U}_k) < \theta\\ \boldsymbol{0} & \text{otherwise.}\end{cases} \tag{6}$$

Here, θ is a threshold value. Intuitively, $\boldsymbol{R}$ in Eq. (5) gives a rotation from old eigen-axes to new ones; hence, let us call $\boldsymbol{R}$ *rotation matrix* here.

2.2 Chunk IPCA

The IPCA in 2.1 is applied to one sample at a time, and the intermediate eigenproblem must be solved repeatedly for every training sample. Hence, the learning may get stuck in a deadlock if a large chunk of training samples is given to learn in a short period. To overcome this problem, the above IPCA is modified so that the eigenspace model Ω can be updated with any size of chunk training samples in a single operation. Let us call this extended algorithm 'Chunk IPCA (CIPCA)'.

Let us assume that N training samples $\boldsymbol{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\} \in \mathcal{R}^{n\times N}$ have been given so far and they were already discarded. Instead of keeping actual training samples, we preserve an eigenspace model $\Omega = (\bar{\boldsymbol{x}}, \boldsymbol{U}_k, \boldsymbol{\Lambda}_k, N)$. Now, assume that a chunk of L training samples $\boldsymbol{Y} = \{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_L\} \in \mathcal{R}^{n\times L}$ are presented. Then, the mean vector $\bar{\boldsymbol{x}}'$ is easily updated as follows:

$$\bar{\boldsymbol{x}}' = \frac{1}{N+L}\left(\sum_{i=1}^{N}\boldsymbol{x}_i + \sum_{j=1}^{L}\boldsymbol{y}_j\right) = \frac{1}{N+L}(N\bar{\boldsymbol{x}} + L\bar{\boldsymbol{y}}). \tag{7}$$

To obtain the new eigenspace model, let us further assume that l eigen-axes must be augmented to avoid the serious loss of essential input information; that is, the eigenspace dimensions are increased by l. Let us denote the augmented eigen-axes as follows:

$$\boldsymbol{H} = [\boldsymbol{h}_1, \cdots, \boldsymbol{h}_l] \in \mathcal{R}^{n\times l}. \tag{8}$$

Then, the updated eigenvector matrix $\boldsymbol{U}'_{k+l}$ is represented by using the rotation matrix $\boldsymbol{R}$ and the current eigenvector matrix $\boldsymbol{U}_k$ as follows:

$$\boldsymbol{U}'_{k+l} = [\boldsymbol{U}_k, \boldsymbol{H}]\boldsymbol{R}. \tag{9}$$

A new eigenvalue problem to be solved is given by

$$\Big(\frac{N}{N+L}\begin{bmatrix}\boldsymbol{\Lambda}_k & \boldsymbol{0}\\ \boldsymbol{0}^T & \boldsymbol{0}\end{bmatrix} + \frac{NL^2}{(N+L)^3}\begin{bmatrix}\bar{\boldsymbol{g}}\bar{\boldsymbol{g}}^T & \bar{\boldsymbol{g}}\bar{\boldsymbol{\gamma}}^T\\ \bar{\boldsymbol{\gamma}}\bar{\boldsymbol{g}}^T & \bar{\boldsymbol{\gamma}}\bar{\boldsymbol{\gamma}}^T\end{bmatrix} + \frac{N^2}{(N+L)^3}\sum_{i=1}^{L}\begin{bmatrix}\boldsymbol{g}'_i\boldsymbol{g}'^T_i & \boldsymbol{g}'_i\boldsymbol{\gamma}'^T_i\\ \boldsymbol{\gamma}'_i\boldsymbol{g}'^T_i & \boldsymbol{\gamma}'_i\boldsymbol{\gamma}'^T_i\end{bmatrix}$$
$$+\frac{L(L+2N)}{(N+L)^3}\sum_{i=1}^{L}\begin{bmatrix}\boldsymbol{g}''_i\boldsymbol{g}''^T_i & \boldsymbol{g}''_i\boldsymbol{\gamma}''^T_i\\ \boldsymbol{\gamma}''_i\boldsymbol{g}''^T_i & \boldsymbol{\gamma}''_i\boldsymbol{\gamma}''^T_i\end{bmatrix}\Big)\boldsymbol{R} = \boldsymbol{R}\boldsymbol{\Lambda}'_{k+l} \tag{10}$$

where

$$\bar{\boldsymbol{g}} = \boldsymbol{U}_k^T(\bar{\boldsymbol{y}} - \bar{\boldsymbol{x}}),\ \boldsymbol{g}'_i = \boldsymbol{U}_k^T(\boldsymbol{y}_i - \bar{\boldsymbol{x}}),\ \boldsymbol{g}''_i = \boldsymbol{U}_k^T(\boldsymbol{y}_i - \bar{\boldsymbol{y}}),$$
$$\bar{\boldsymbol{\gamma}} = \boldsymbol{H}^T(\bar{\boldsymbol{y}} - \bar{\boldsymbol{x}}),\ \boldsymbol{\gamma}'_i = \boldsymbol{H}^T(\boldsymbol{y}_i - \bar{\boldsymbol{x}}),\ \boldsymbol{\gamma}''_i = \boldsymbol{H}^T(\boldsymbol{y}_i - \bar{\boldsymbol{y}}).$$

Solving this eigenproblem, a new rotation matrix $\boldsymbol{R}$ and the eigenvalue matrix $\boldsymbol{\Lambda}'_{k+l}$ are obtained. Then, the corresponding new eigenvector matrix $\boldsymbol{U}'_{k+l}$ is obtained by using Eq. (9).

In CIPCA, the number of augmented eigen-axes is also determined by finding a minimum k such that the accumulation ratio $A(\boldsymbol{U}_k)$ satisfies the same condition described in 2.1. However, the update equation in Eq. (3) must be modified such that it can be updated with a chunk of training samples in one-pass [1]. This is given by

$$A'(\boldsymbol{U}_k) = \frac{\sum_{i=1}^{k}\lambda_i + \frac{L}{N+L}\|\bar{\boldsymbol{g}}\|^2 + \frac{1}{N}\sum_{j=1}^{L}\|\boldsymbol{g}''_i\|^2}{\sum_{i=1}^{n}\lambda_i + \frac{L}{N+L}\|\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}}\|^2 + \frac{1}{N}\sum_{j=1}^{L}\|\boldsymbol{y}_j - \bar{\boldsymbol{y}}\|^2}. \tag{11}$$

Finally, let us explain how to determined the augmented eigen-axes $\boldsymbol{H}$ in Eq. (8). In CIPCA, the number of augmented eigen-axes is not restricted to one. If the given L training samples are represented by $\tilde{L}$ linearly independent vectors, the maximum number of augmented eigen-axes is also $\tilde{L}$. However, the feature space spanned by all of the augmented eigen-axes is redundant in general; in addition, if the chunk size is large, the computation costs to solve the intermediate eigenproblem in Eq. (10) would be considerably expensive. Therefore, we should select informative eigen-axes from the $\tilde{L}$ eigen-axes efficiently. Since the number of eigen-axes to be augmented is varied from 0 to $\tilde{L}$, the number of possible combinations of eigen-axes is represented by $\sum_{i=0}^{\tilde{L}} {}_LC_i$. If the chunk size is large, the computation costs for finding an optimal set of augmented eigen-axes would be large. To avoid such an exhaustive search, we introduce a kind of greedy search based on the accumulation ratio.

To construct a compact feature space, we should find a smallest set $\boldsymbol{H}$ of augmented eigen-axes such that the eigenspace includes as much the energy of the given chunk data as possible. A straightforward way to find the set is to select an eigen-axis one by one, each of which gives a maximum accumulation ratio. The algorithm of the eigen-axis selection is summarized below.

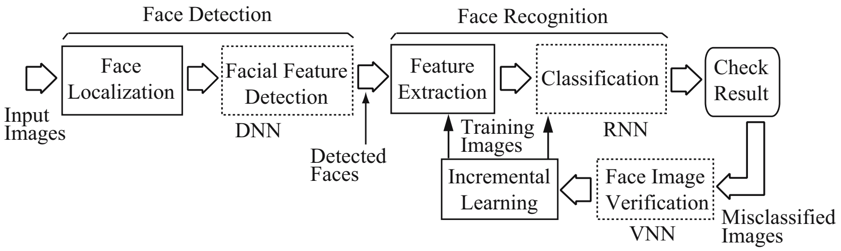

Fig. 1. The block diagram of information processing in the face recognition system. The block with a dotted line is implemented by a neural network.

[Selection of Eigen-axes in CIPCA]

```
Inputs:  – Eigenspace model $\Omega = (\bar{\boldsymbol{x}}, \boldsymbol{U}_k, \boldsymbol{\Lambda}_k, N)$.
         – A chunk of $L$ training samples $\boldsymbol{Y} = \{\boldsymbol{y}^{(1)}, \cdots, \boldsymbol{y}^{(L)}\}$.
         – Threshold $\theta$ of the accumulation ratio.
  Do the following procedure:
    i) Set $\boldsymbol{H} = \{\ \}$ and $l = 0$. Calculate the mean vector $\bar{\boldsymbol{y}}$ of the given training samples $\boldsymbol{Y}$.
    ii) Calculate the accumulation ratio $A'(\boldsymbol{U}'_k)$ based on Eq. (3). If $A'(\boldsymbol{U}'_k) > \theta$, terminate this algorithm.
    iii) Obtain the following residue vectors $\boldsymbol{h}_i$ $(i = 1, \cdots, L)$ for all of the given training samples $\boldsymbol{y}^{(i)}$:
         $$\boldsymbol{h}_i = \frac{\boldsymbol{r}_i}{\|\boldsymbol{r}_i\|} \quad \text{where} \quad \boldsymbol{r}_i = (\boldsymbol{y}^{(i)} - \bar{\boldsymbol{x}}) - [\boldsymbol{U}_k, \boldsymbol{H}][\boldsymbol{U}_k, \boldsymbol{H}]^T (\boldsymbol{y}^{(i)} - \bar{\boldsymbol{x}}).$$
         Define an index set $\mathcal{H}$ of $\boldsymbol{h}_i$.
    iv) Find the following residue vector $\boldsymbol{h}_{i'}$ which gives the maximum increment $\Delta\tilde{A}'_i$: $\boldsymbol{h}_{i'} = \arg\max_{i \in \mathcal{H}} \Delta\tilde{A}'_i$ where
         $$\Delta\tilde{A}'_i = \frac{L}{N+L}\{\boldsymbol{h}_i^T(\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}})\}^2 + \frac{1}{N}\sum_{j=1}^{L}\{\boldsymbol{h}_i^T(\boldsymbol{y}^{(j)} - \bar{\boldsymbol{y}})\}^2.$$
    v) Add $\boldsymbol{h}_{i'}$ to $\boldsymbol{H}$ (i.e., $\boldsymbol{H} \leftarrow [\boldsymbol{H}, \boldsymbol{h}_{i'}]$), $l \leftarrow l + 1$, and remove $i'$ from $\mathcal{H}$. If $\mathcal{H}$ is empty, terminate this algorithm.
    vi) Calculate the updated accumulation ratio $A'(\boldsymbol{U}'_{k+l})$ based on Eq. (11). If $A'(\boldsymbol{U}'_{k+l}) > \theta$, terminate this algorithm. Otherwise, go to Step iv).
Output:  The optimal set of augmented eigen-axes $\boldsymbol{H} = \{\boldsymbol{h}_1, \cdots, \boldsymbol{h}_l\}$.
```

3 Face Recognition System

Figure 1 shows the overall process in our face recognition system. As we can see from Fig. 1, the presented system mainly consists of the four parts: face detection, face recognition, face image verification, and incremental learning.

See [4] for the further details of this system.

4 Performance Evaluation

4.1 Experimental Setup

To simulate real-life consecutive recognition and learning, 224 video clips are collected for 22 persons (19 males and 3 females) during about 11 months such that temporal changes in facial appearances are included. Seven people (5 males and 2 females) are chosen as registrants and the other people (14 males and a female) are non-registrants. The duration of each video clip is 5 - 15 (sec.). A video clip is given to the face detection part, and the detected face images are automatically forwarded to the face recognition part. The numbers of detected face images are summarized in Table 1. The three letters in Table 1 indicate the code of the 22 subjects in which M/F and R/U mean Male/Female and Registered/Unregistered, respectively; for example, the third registered male is coded as MR3.

Table 1. Two face datasets (Set A and Set B) for training and test. The three letters in the upper row mean the registrant code and the values in the second and third rows are the numbers of face images.

Set	MR1	FR1	MR2	MR3	FR2	MR4	MR5	FU1	MU1	MU2	MU3	MU4
A	351	254	364	381	241	400	186	133	181	294	110	103
B	170	220	297	671	297	241	359	126	228	292	80	233

Set	MU5	MU6	MU7	MU8	MU9	MU10	MU11	MU12	MU13	MU14	Total
A	170	186	174	33	79	15	75	17	10	9	3766
B	117	202	182	14	9	14	28	18	9	9	3816

To evaluate the recognition performance based on the two-fold cross-validation, the whole dataset is subdivided into two subsets: Set A and Set B. When Set A is used for learning RNN, Set B is used for testing the generalization performance, and vice versa. Note that since the incremental learning is applied only for misclassified face images, the recognition accuracy before the incremental learning is an important performance measure. Hence, there are at least two performance measures for the training dataset: one is the performance of RNN using a set of training samples given at each learning stage, and the other is the performance using all training datasets given so far after the incremental learning is carried out. In the following, let us call the former and latter datasets as *incremental dataset* and *training dataset*, respectively. Besides, let us call the performances over the incremental dataset and training dataset as *incremental performance* and *training performance*, respectively. We divide the whole dataset into 16 subsets, each of which corresponds to an incremental dataset. Table 2 shows the number of images included in the incremental datasets.

The size of an initial dataset can influence the test performance because different initial eigen-spaces are constructed. However, if the incremental learning is successfully carried out, the final performance should not depend on the size of the initial dataset. Hence, the three different series of incremental datasets

Table 2. Number of images included in the 16 incremental datasets

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Set A	220	232	304	205	228	272	239	258	212	233	290	212	257	188	199	217
Set B	288	204	269	246	273	270	240	281	205	249	194	241	214	226	210	206

Table 3. Three series of incremental datasets. The number in Table 3 corresponds to the tag number of the corresponding incremental dataset.

Stage	Init.	1	2	$\cdots$	12	13	14	15
Case 1	1	2	3	$\cdots$	13	14	15	16
Case 2	1, 2	3	4	$\cdots$	14	15	16	—
Case 3	1, 2, 3	4	5	$\cdots$	15	16	—	—

shown in Table 3 are defined to see the influence. Note that the number in Table 3 corresponds to the tag number (1-16) of the incremental dataset in Table 2. Hence, we can see that Case 1 has 15 learning stages and the number of images in the initial dataset is 220 for Set A and 288 for Set B, which correspond to 6.7% and 7.5% over the whole data. On the other hand, the sizes of the initial datasets in Case 2 and Case 3 are set to a larger value as compared with that in Case 1; while the numbers of learning stages are smaller than that in Case 1. Figure 2 shows the examples of detected face images for three registered persons at several learning stages.

When an initial dataset is trained in RNN, the number of hidden units is fixed with 50 in this experiment. The other parameters are set as follows: $\sigma^2 = 7$, $\varepsilon = 0.01$, and $\delta = 5$. The threshold θ of the accumulation ratio in IPCA is set to 0.9; thus, when the accumulation ratio is below 0.9, a new eigen-axis is augmented.

4.2 Experimental Results

Learning Time. Figure 3 shows the transition of learning time over 15 learning stages when the chunk size L is 10 in CIPCA. The curves of 'CIPCA' and 'IPCA' show the learning time for feature selection, while those of 'CIPCA+RAN-LTM'

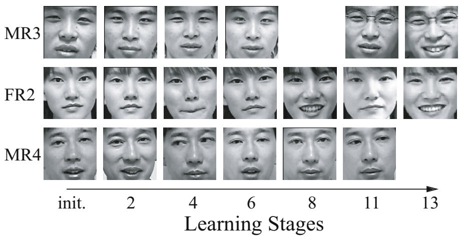

Fig. 2. Examples of face images trained at different learning stages

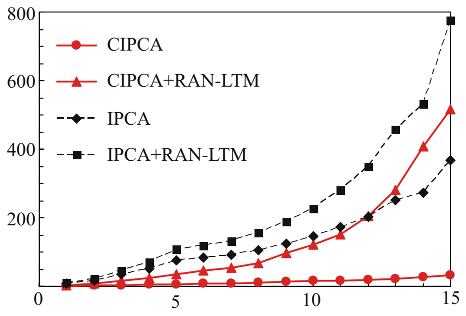

Fig. 3. Transition of learning time (sec.)

Table 4. Comparisons of Learning time and dimensions of feature vectors at the final learning stage. CIPCA(10), CIPCA(50), and CIPCA(100) stand for CIPCA in which the chunk sizes are set to 10, 50, and 100, respectively.

	IPCA	CIPCA(10)	CIPCA(50)	CIPCA(100)
Time (sec.)	376.2	45.6	22.5	18.1
Dimensions	178	167	186	192

and 'IPCA+RAN-LTM' mean the learning time for both feature selection and classifier. As you can see from the results, the learning time of feature selection by CIPCA is greatly reduced as compared with IPCA. This is also confirmed in Table 4.

The learning time of CIPCA decreases as the chunk size increases, and CIPCA is much faster than IPCA even though the feature dimensions at the final stage do not have large differences between IPCA and CIPCA. When the chunk size is 10, CIPCA is about 8 times faster than IPCA. The reason why the decreasing rate of the learning time becomes small for larger chunk size is that the time for finding eigen-axes dominates the total learning time [3].

Classification Accuracy. To evaluate the effectiveness of learning a feature space, the classification accuracy of RAN-LTM is examined when the following three eigen-space models are adopted: (1) static eigenspace model with PCA, (2) adaptive eigenspace model with the extended IPCA, and (3) adaptive eigenspace model with CIPCA. For notational simplicity, these three models are denoted by PCA, IPCA, and CIPCA, respectively.

Figures 4 (a)-(c) show the transition of recognition accuracy over 15 learning stages when the percentage of initial training data is (a) 6.7%, (b) 12.5%, and (c) 20%, respectively. As stated before, the size of an initial dataset can influence the recognition accuracy because different eigenspaces are constructed at the start point. As we can see from Figs. 4(a)-(c), the initial test performance at stage 0 is higher when the number of initial training data is larger; however, the test performance of IPCA and CIPCA is monotonously enhanced over the

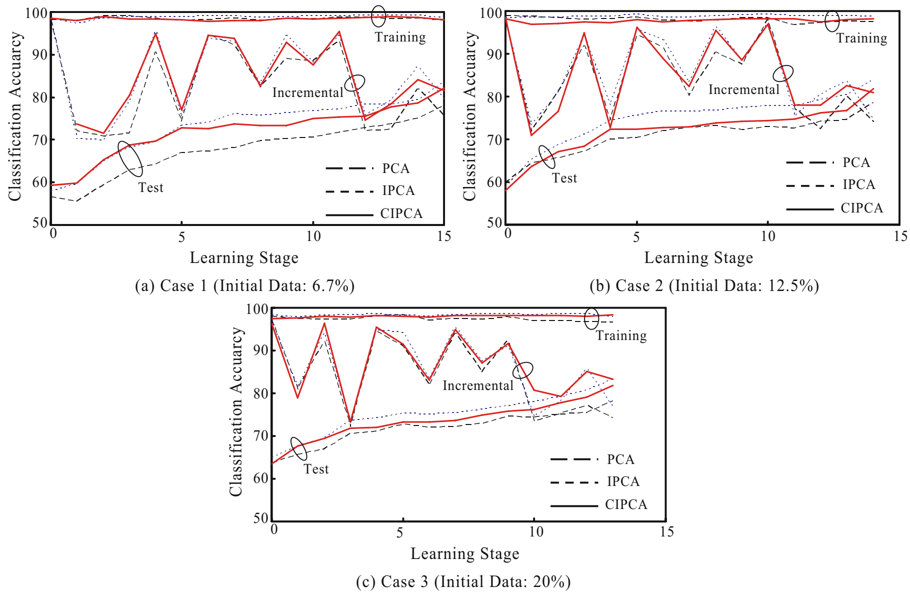

Fig. 4. Time courses of the recognition accuracy rate for three different datasets (incremental, training, test) over the learning stages when the percentages of initial training datasets are set to (a) 6.7%, (b) 12.5%, and (c) 20.0%

learning stages and it reaches almost the same accuracy regardless of the initial datasets. Considering that the total number of training data is the same among the three cases, we can say that all the information included in the training dataset is stably accumulated in RNN without serious forgetting. In addition, the test performance of RNN with IPCA and CIPCA has significant improvement against RNN with PCA although CIPCA has slightly lower performance than IPCA. This degradation originates from the approximation error of the eigenspace model with CIPCA. However, the above results still indicate that the reconstruction of RNN works well in accordance with the evolution of the eigenspace model, and that the incremental learning of a feature space is very effective to enhance the generalization performance of RNN.

Moreover, we can see that although the incremental performance is fluctuated, the training performance of RNN with IPCA and CIPCA changes very stably over the learning stages. On the other hand, the training performance of RNN with PCA rather drops down as the learning stage proceeds. Since the incremental performance is defined as a kind of test performance for the incoming training dataset, it is natural to be fluctuated. The important thing is that the misclassified images in the incremental dataset are trained stably without degrading the classification accuracy for the past training data.

From the above results, we can conclude that the proposed incremental learning scheme, in which the feature space and the classifier are simultaneously learned based on CIPCA and RAN-LTM, works quite well and the learning time is significantly reduced without serious performance degradation.

5 Conclusions

This paper presents a new approach to constructing adaptive face recognition systems in which a low-dimensional feature space and a classifier are incrementally learned in an online fashion. To learn a useful feature space incrementally, we adopt Chunk Incremental Principal Component Analysis (CIPCA) in which a chunk of given training samples are learned at a time to update an eigen-space model.

To evaluate the incremental learning properties, a self-compiled face image database is applied to the proposed model. In the experiments, we verify that the proposed incremental learning works well without serious forgetting and the test performance is improved as the incremental learning stages proceed. Furthermore, we also show that replacing the extended IPCA with CIPCA is very efficient in term of learning time; in fact, the learning speed of CIPCA was at least 8 times faster than IPCA.

Acknowledgment

The authors would like to thank Prof. Shigeo Abe for his useful discussions and comments, and would like to thank Mr. Michiro Hirai for his great devotion to the development of the face recognition system.

References

1. Kasabov, N.: Evolving Connectionist Systems: Methods and Applications in Bioinformatics, Brain Study and Intelligent Machines. Springer, Heidelberg (2002)
2. Hall, P., Martin, R.: Incremental Eigenanalysis for Classification. In: Proc. of British Machine Vision Conference, vol. 1, pp. 286–295 (1998)
3. Ozawa, S., Pang, S., Kasabov, N.: An Incremental Principal Component Analysis for Chunk Data. In: Proc. of FUZZ-IEEE, pp. 10493–10500 (2006)
4. Ozawa, S., Toh, S.L., Abe, S., Pang, S., Kasabov, N.: Incremental Learning of Feature Space and Classifier for Face Recognition. Neural Networks 18(5-6), 575–584 (2005)
5. Ozawa, S., Pang, S., Kasabov, N.: A Modified Incremental Principal Component Analysis for On-line Learning of Feature Space and Classifier. In: Zhang, C., W. Guesgen, H., Yeap, W.-K. (eds.) PRICAI 2004. LNCS (LNAI), vol. 3157, pp. 231–240. Springer, Heidelberg (2004)