A Face Recognition System Using Neural Networks with Incremental Learning Ability

Soon Lee Toh and Seiichi Ozawa
Graduate School of Science and Technology, Kobe University
1-1 Rokko-dai, Nada-ku, Kobe 657-8501, JAPAN
den@frenchblue.scitec.kobe-u.ac.jp
ozawa@eedept.kobe-u.ac.jp

Abstract

This paper presents a fully automated face recognition system with incremental learning ability that has the following two desirable features: one-pass incremental learning and automatic generation of training data. As a classifier of face images, an evolving type of neural network called Resource Allocating Network with Long-Term Memory (RAN-LTM) is adopted here. This model enables us to realize efficient incremental learning without suffering from serious forgetting. In the face detection procedure, face localization is conducted based on the information of skin color and edges at first. Then, facial features are searched for within the localized regions using a Resource Allocation Network, and the selected features are used for in the construction of face candidates. After the face detection, the face candidates are classified using RAN-LTM. The incremental learning routine is applied to only misclassified data that are collected automatically in the recognition phase. Experimental results show that the recognition accuracy improves without increasing the false-positive rate even if the incremental learning proceeds. This fact suggests that incremental learning is a useful approach to face recognition tasks.

1. Introduction

Understanding how people process and recognize faces has been a challenging task in the field of object recognition for a long time. Many approaches have been proposed to simulate the human process, in which various adaptive mechanisms are introduced such as neural networks, genetic algorithms, and support vector machines [1].

Many of the proposed methods can achieve considerably good recognition results for available benchmark datasets. Several commercial products are also based on these methods, but in practice the recognition rates usually drop rather drastically as compared with the rates achieved for benchmark datasets. One primary reason for this failure in the existing commercial face recognition systems is that insufficient or inappropriate data have been used in the training of classifiers. Since human faces are prone to be changed in the course of time, it is quite difficult to collect all of the appropriate training data in advance.

A way to circumvent this problem is to introduce incremental learning [2, 3] into classifiers, much in the same way humans learn to recognize both unknown faces and known but varied faces. Most of the current commercial systems lack a learning ability to improve their performance automatically. Thus we think the concept of incremental learning is the essential factor to build a robust recognition system in real situations.

In this paper, we present a fully automated adaptive face recognition system in which the following two functions are realized for practical purposes by introducing two types of neural networks:

1. one-pass incremental learning,
2. automatic generation of training data.

In the one-pass incremental learning, each of the training data is given only once; that is, the previously given data will never be available in the future [3]. The largest benefit of introducing the one-pass incremental learning is that the system does not need so much large memory to enhance its performance; that is, the retraining of many training data is unnecessary. This property is very useful especially for small-scale systems. On the other hand, the second function is very important in the practical sense, because the incremental learning is carried out only after the actual recognition is done. That is to say, training data should be automatically generated based on the recognition results without any human intervention.

In Section 2, a brief explanation of our proposed incremental learning model is given. Section 3 describes the developed face recognition system in which face detection,

face recognition, and incremental learning are fully automatically carried out online for video streams. In Section 4, some recognition experiments are conducted to evaluate the incremental learning performance, and the concluding remarks are given in Section 5.

2. Incremental Learning Model

Disruption in neural networks, called "forgetting" or "catastrophic interference", often occurs during incremental learning. It is caused by the excessive adaptation of connection weights to new data. One way of overcoming this problem is that only representative training data are kept in memory and some of them are trained with newly given training data. To realize this, we have adopted a "Resource Allocating Network with Long-Term Memory (RAN-LTM)" using a fast learning algorithm based on the linear method [4] as our learning classifier.

2.1. Architecture of RAN-LTM

Figure 1 shows the architecture of RAN-LTM, which consists of two parts: Resource Allocating Network (RAN) [5] and Long-Term Memory (LTM). RAN is an extended model of the Radial Basis Function (RBF) network in which the automated allocation mechanism of hidden units is introduced. Hence, the information processing is almost the same as that in RBF except for the allocation of hidden units.

Let us denote the number of input units, hidden units, and output units as I, J, K, respectively. Moreover, let the inputs be $\boldsymbol{x} = \{x_1, \cdots, x_I\}'$, the outputs of hidden units be $\boldsymbol{y} = \{y_1, \cdots, y_J\}'$, and the outputs be $\boldsymbol{z} = \{z_1, \cdots, z_K\}'$. The calculation in the forward direction is given as follows:

$$y_j = \exp(-\frac{\|\boldsymbol{x} - \boldsymbol{c}_j\|^2}{2\sigma_j^2}) \quad (j = 1, \cdots, J), \tag{1}$$

$$z_k = \sum_{j=1}^{J} w_{kj} y_j + \xi_k \quad (k = 1, \cdots, K) \tag{2}$$

where $\boldsymbol{c}_j = \{c_{j1}, \cdots, c_{jI}\}'$ and σ_j^2 are respectively the center and variance of the jth hidden unit, w_{kj} is the connection weight from the jth hidden unit to the kth output unit, and ξ_k is the bias of the kth output unit.

The items stored in LTM are called "memory items" that correspond to the representative input-output data. These data can be selected from training samples, and they are trained with newly given training data to suppress forgetting. In the learning algorithm based on the linear method, a memory item is created when a hidden unit is allocated: that is, the RBF center and the corresponding output are stored in a memory item. As a result, the number of memory items is equivalent to the number of hidden units in this learning approach.

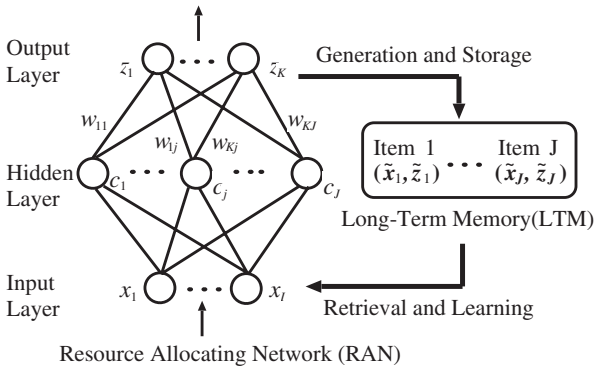

Figure 1. The architecture of RAN-LTM.

2.2. Learning Algorithm of RAN-LTM

The learning algorithm of RAN-LTM based on the linear method is divided into two phases: the dynamic allocation of hidden units (i.e., the selection of RBF centers in an incremental fashion) and the calculation of connection weighs between hidden and output units. The procedure in the former phase is almost the same as that in the original RAN [5], except that hidden units can be added after the update of connection weights (see Step 5 in the learning algorithm) and that memory items are generated at the same time. Once hidden units are allocated, the centers are fixed afterwards. Therefore, the connection weights $\boldsymbol{W} = \{w_{kj}\}$ are only parameters that are updated based on the output errors. To minimize the errors based on the least squares method, it is well known that the following linear equalities should be solved:

$$\boldsymbol{\Phi}\, \boldsymbol{W} = \boldsymbol{T} \tag{3}$$

where $\boldsymbol{T}$ is the matrix whose column vectors correspond to the target outputs. Suppose that a training sample $(\boldsymbol{x}, \boldsymbol{d})$ is given at time t and M memory items $(\tilde{\boldsymbol{x}}_m, \tilde{\boldsymbol{z}}_m)$ $(m = 1, \cdots, M)$ have already been created, then the target matrix $\boldsymbol{T}$ are formed as follows:

$$\boldsymbol{T} = \{\boldsymbol{d}, \tilde{\boldsymbol{z}}_1, \cdots, \tilde{\boldsymbol{z}}_M\}. \tag{4}$$

Furthermore, $\boldsymbol{\Phi}$ in Eq. (3) is calculated from these target vectors as follows:

$$\boldsymbol{\Phi} = \begin{bmatrix} \varphi_{11} & \cdots & \cdots & \cdots & \varphi_{1J} \\ \vdots & \ddots & & & \vdots \\ \vdots & & \varphi_{mj} & & \vdots \\ \vdots & & & \ddots & \vdots \\ \varphi_{M+1,1} & \cdots & \cdots & \cdots & \varphi_{M+1,J} \end{bmatrix}, \tag{5}$$

where

$$\varphi_{1j} = \exp(\frac{\|\boldsymbol{x}-\boldsymbol{c}_j\|}{\sigma_j^2}), \quad \varphi_{mj} = \exp(\frac{\|\tilde{\boldsymbol{x}}_m-\boldsymbol{c}_j\|}{\sigma_j^2}) \\ (j=1,\cdots,J;\ m=1,\cdots,M).$$

To solve $\boldsymbol{W}$ in Eq. (3), Singular Value Decomposition (SVD) can be used. The learning algorithm of RAN-LTM is summarized as follows:

[Learning Algorithm]

1. Find the nearest center $\boldsymbol{c}^*$ to an input $\boldsymbol{x}$ and then calculate the output error E.
2. If $E > \varepsilon$ and $\|\boldsymbol{x}-\boldsymbol{c}^*\| > \delta$, then allocate a hidden unit (i.e., $J \leftarrow J+1$) and create a memory item $(\tilde{\boldsymbol{x}}_M, \tilde{\boldsymbol{z}}_M)$ as follows:

   [Hidden Unit] $\boldsymbol{w}_J = \boldsymbol{d}-\boldsymbol{z}, \quad \boldsymbol{c}_J = \boldsymbol{x},$

   [Memory Item] $\tilde{\boldsymbol{x}}_m = \boldsymbol{x}, \quad \tilde{\boldsymbol{z}}_m = \boldsymbol{d}.$

   Then, go to Step 6. Otherwise, go to Step 3.
3. Calculate hidden outputs for the training sample $(\boldsymbol{x}, \boldsymbol{d})$ and memory items $(\tilde{\boldsymbol{x}}_m, \tilde{\boldsymbol{z}}_m)$ $(l=1,\cdots,M)$, and calculate $\boldsymbol{\Phi}$ in Eq. (3).
4. Using SVD, decompose $\boldsymbol{\Phi}$ as follows: $\boldsymbol{\Phi} = \boldsymbol{U}\boldsymbol{H}\boldsymbol{V}'$ where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices, and $\boldsymbol{H}$ is a diagonal matrix. Then, calculate the weight matrix as follows: $\boldsymbol{W} = \boldsymbol{V}\boldsymbol{H}^{-1}\boldsymbol{U}'\boldsymbol{T}$.
5. Give the input $\boldsymbol{x}$ to RAN-LTM again, and calculate the output error E. If $E > \varepsilon$, add a hidden unit and generate a memory item $(\tilde{\boldsymbol{x}}_M, \tilde{\boldsymbol{z}}_M)$ in the same way of Step 2.
6. If a new training sample is given, go back to Step 1.

3. Face Recognition System

3.1. Outline of Process Flow

The proposed fully automated face recognition system consists of the following four functions: face detection, face recognition, incremental learning, and a self-generation of training data. These functional components must be operated online without any human intervention. Figure 2 shows the overall process flow of the proposed system.

In the face detection part, at each time frame, face regions are localized based on the information of skin color and edges at first. Thereafter three types of facial features (eye, nose, mouth) are searched for within the localized regions through raster operations. In each raster operation, a small sub-image is extracted from a localized region, then the eigen-features of the sub-image are given to a Detection Neural Network (DNN) to verify if it corresponds to any one of the facial features. These eigen-features are obtained

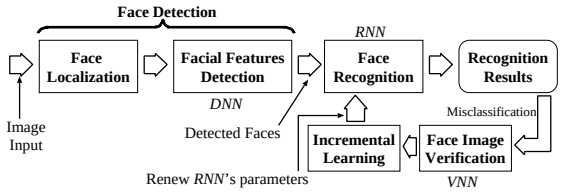

Figure 2. Block diagram of the process flow in the proposed face recognition system. Italic characters show the name of the three neural networks to be adapted.

using Principal Component Analysis (PCA). After all raster operations are done, face candidates are created by combining the identified facial features based on some geometrical constraints. The output of the face detection part is the center positions of the face candidates.

Next, in the face recognition part, the rectangular regions of the face candidates are first extracted from the original image, and then the extracted regions are applied to PCA in order to reduce their dimensions. These PCA features are given to Recognition Neural Network (RNN) as its input for classification. Note that RNN is implemented by RAN-LTM.

In the incremental learning part, misclassified images are collected from video clips as training data. However, there is a possibility that non-face images happen to be mixed into these training data because the perfect face detection is not ensured. Thus, another neural network called Verification Neural Network (VNN) is introduced into this part in order to filter out non-face images. After this verification process, the incremental learning for the misclassified face images is carried out by RNN.

3.2. Face Detection Part

Many approaches to the face detection have been proposed so far [6]. We adopt a common detection process that consists of the following two phases: face localization and face feature identification, and the results from each process are shown in Fig. 3.

3.2.1 Face Localization

To reduce time-consuming computations, facial regions are first localized for input images. This localization is done in the following procedure:

1. Perform the edge extraction by applying the MaxMin filter and Sobel horizontal filter to an input image. (See Fig. 3(a).)

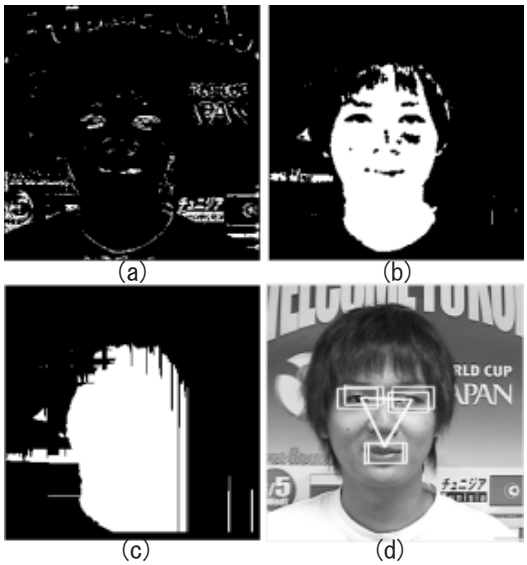

Figure 3. The results of applying (a) edge filter and (b) skin color filter to an input image. (c) A face region (in white) generated by combining the results in (a) and (b). (d) A final result of the face detection part (only one face detected in this case).

2. Calculate the projection (called skin-color features) of the input image to the skin-color axis. (See Fig. 3(b).)

3. Search for facial sub-regions using both the edge information and skin-color features, and extract the sub-regions from the original input image. (See Fig. 3(c).)

The skin-color axis is determined based on the least squares method to approximate the distribution of Japanese skin images in advance. In our preliminary experiment, this facial region extraction succeeded with 99% accuracy for a Japanese face image database.

3.2.2 Facial Feature Identification

Three types of facial features (i.e., eye, nose, mouth) are searched for within the facial regions extracted above at first. For the facial features, all of the combinations of these features are checked if they are satisfied with the defined facial geometric constrains. A minimum of three features must be found on the geometric template to identify as a face candidate. An example can be seen in Fig. 3(d) where the rectangular regions are the identified facial features and the inverted triangle marks the position of a face candidate. The center positions of all face candidates are sent to the next part.

To verify if the searched regions correspond to facial ones, the conventional RAN is adopted here as DNN. Note that there is no need for incremental learning of DNN because a large training data set is used and the features of the human eye, nose, and mouth are considered as invariant. The feature identification procedure is shown as follows:

1. Carry out the following raster operations on the region localized in the previous process:
   (a) Extract a 40×20 pixel sub-image from the localized region.
   (b) Carry out the dimension reduction for the extracted sub-image using PCA.
   (c) Check if the resultant 30-dimensional eigen-feature vector corresponds to one of the facial features using DNN.
2. For all of the identified facial features, construct possible face candidates based on the defined geometrical constraints.
3. Obtain the center positions of the identified face candidates, and send them to the next process.

3.3. Face Recognition Part

In this part, detected face candidates are differentiated between registered faces and non-registered ones. RNN implemented by RAN-LTM is used as the classifier to facilitate the later incremental learning. The eigenface approach [7] is adopted to reduce the dimensions of facial features. The eigen-features are sent to the RNN classifier. The recognition procedure is conducted in the following manner:

1. Extract 90×90 sub-images whose centers correspond to the centers of the face candidates.
2. Transform each of the sub-images into a 70-dimensional eigen-feature vector using PCA.
3. Perform the face recognition using RNN.

3.4. Incremental Learning Part

Suppose that the system is informed whether the classification was right or not. Then, the system can know what facial images should be learned to make correct decisions next time. This adaptation mechanism is realized by the incremental learning ability of RNN. The incremental learning procedure is carried out as follows:

1. If the classification is correct in the previous part, terminate this procedure. Otherwise, go to Step 2.
2. Obtain the misclassified eigen-features and then verify if they are face features using VNN. If they are non-face features, terminate this procedure. Otherwise, go to Step 3.

3. Store a pair of the misclassified eigen-features and the corresponding class label as a training sample into a memory device. If the number of stored training samples exceeds a certain value, go to Step 4. Otherwise, terminate this procedure.
4. Train VNN using all training samples stored in the memory based on the incremental learning algorithm described in Subsection 2.2.
5. Clear all training samples in the memory.

4. Experiments

4.1. Experimental Conditions and Evaluation Method

The largest benefit of introducing incremental learning is that the system does not need so much large memory to enhance its performance. This property is useful for small-scale systems rather than large-scale systems with high-performance servers. In this sense, we have intended to develop our face recognition system aimed for intelligent entrance control systems in residential environments. Hence, we can assume several conditions for the operating environments. The faces to be recognized are presumed to be frontal, allowing for slight rotations both laterally and longitudinally. The lighting conditions are changed to evaluate the robustness in both daylight and indoor conditions. Due to space constraints, we omit the detection results and focus on the recognition performance.

The evaluation dataset is divided into two sets. One is used for assessing the performance of incremental learning in RNN through an online recognition process (called "online dataset"). From the online dataset, images are given to the system one by one, and the recognition and incremental learning are carried out online. The other dataset is used for assessing the generalization performance of RNN (called "test dataset"). Hence, the images in this test dataset are not used for training.

These datasets contain video clips of seven people: four people (2 males and 2 females) are chosen as registered persons and the other three people are non-registered persons. The video clips have durations of $5 \sim 15$ seconds and they are taken over two weeks such that some changes in facial appearances are included. The online dataset consists of 654 images detected from the video clips of the first week; on the other hand, the test dataset consists of 499 images detected from the clips of the second week.

To simplify the evaluation procedure, the online dataset is further divided into six subsets separated by the time when the video is taken to simulate real-life consecutive learning. These subsets are given to the face recognition system in turn and the misclassified data are incrementally trained in RNN. The number of data in each subset is as follows: 139 (initial training set) $\rightarrow$ 107 $\rightarrow$ 75 $\rightarrow$ 98 $\rightarrow$ 106 $\rightarrow$ 129. Hence, five stages of incremental learning are carried out. Then the effectiveness of incremental learning is evaluated in terms of the recognition accuracy for all subsets given so far. Therefore, the number of evaluation data increases as the incremental learning stage proceeds: $139 \rightarrow 246 \rightarrow 321 \rightarrow 419 \rightarrow 525 \rightarrow 654$.

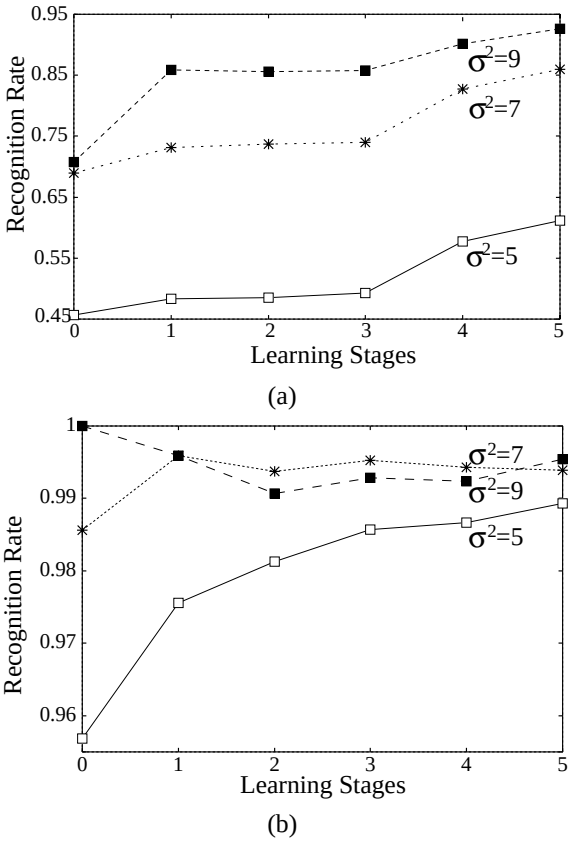

Figure 4. Time courses of recognition results for (a) test dataset and (b) online dataset.

As a security system, it is also important to evaluate the false-positive rate (the rate of the cases where non-registered or non-faces are classified as registered faces). To do this, we evaluate the recognition performance using another set of 3311 images, which consists of 1748 non-face images and 1536 non-registered faces. This dataset is referred to as the FP-dataset.

4.2. Experimental Results

Figure 4 shows the time courses of the recognition performance for two datasets through the five incremental learning stages. We compare the results of three RNNs whose radial bases have different variances ($\sigma^2 = 5, 7, 9$).

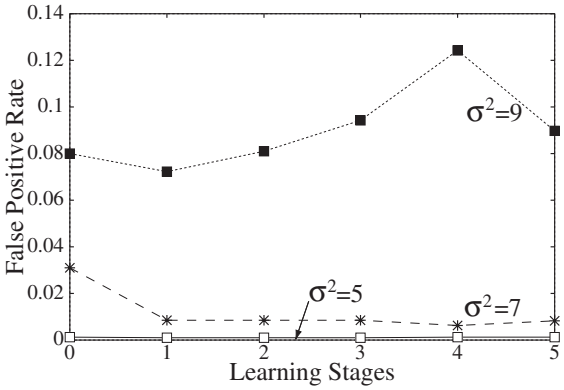

Figure 5. False-positive rate for the FP-dataset.

As seen from Fig. 4(a), the recognition rate for all three variance settings improves as the incremental learning stage proceeds. This result demonstrates that the incremental learning works effectively to enhance the generalization performance for unseen data. However, due to insufficient datasets, we cannot ascertain whether there is a ceiling for the recognition rate (i.e., whether further incremental learning is of use only up to a certain point).

The next important point to evaluate is whether destructive interference has occurred due to the incremental learning. To do this, we evaluate the recognition performance for the online dataset as stated in the previous subsection. The results are shown in Fig. 4(b). Although the recognition rate drops initially in one of the variance settings, overall we can say that the recognition performance stabilizes at an acceptable level. We can thereby conclude that the memory items recalled during incremental learning are effective in suppressing the destructive interference.

Figure 5 shows the false-positive performance for the FP-dataset. In the two variance settings ($\sigma^2 = 5, 7$), the false-positive rate is maintained below 1%. If the variance is set to a large value (e.g., $\sigma^2 = 9$), the base functions have wide response properties but there may be insufficient data initially to determine the connection weights properly. This is one of the reasons why the false-positive rate is rather high in the case of $\sigma^2 = 9$.

5. Conclusions and Further Work

In this paper, we have proposed an intelligent face recognition system with the following two desirable features: one-pass incremental learning and automatic generation of training data. This system can be implemented with limited hardware constraints and is suitable for the use in small devices where hardware capabilities are restricted such as home security systems. A memory-based learning approach (accumulating and retraining all previous data) may possibly obtain better recognition results but it is infeasible for large datasets like image streams. From the online recognition experiments, we can conclude that introducing the incremental learning function into the system is quite effective in the sense that the system can enhance its recognition performance automatically without any administrator intervention.

Several open questions still remain in our face recognition system. First, the face detection method introduced here is still rather immature in terms of the computation costs and accuracy. Moreover, the robustness for image variations in rotations, size, illumination, etc. must be improved. Here, we evaluated the recognition performance only for small datasets of Japanese people. From the aspect of security systems, such simple evaluations are useless. Hence, the evaluation on the robustness for the larger datasets is necessary in practical use.

Acknowledgment

The authors would like to thank Prof. Shigeo Abe, Dr. Motohide Yoshimura, and Shinji Kita for their useful discussions and comments. This research was supported by Matsushita Electric Works, Ltd. and the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Young Scientists (B).

References

[1] L. C. Jain, U. Halici, I. Hayashi, and S. B. Lee. *Intelligent Biometric Techniques in Fingerprint and Face Recognition.* CRC, 1999.

[2] J. L. Elman. Learning and Development in Neural Networks: The Importance of Starting Small. *Cognition*, 48: 71-99, 1993.

[3] N. Kasabov. *Evolving connectionist systems: Methods and applications in bioinformatics, brain study and intelligent machines.* Springer-Verlag, 2002.

[4] K. Okamoto, S. Ozawa, and S. Abe. A Fast Incremental Learning Algorithm of RBF Networks with Long-Term Memory. *Proc. Int. Joint Conf. on Neural Networks*, 102-107, 2003.

[5] J. Platt. A Resource-Allocating Network for Function Interpolation. *Neural Computation*, 3(2): 213-225, 1991.

[6] M. H. Yang, D. J. Kriegman, and N. Ahuja. Detecting Faces in Images: A Survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(1), 2002.

[7] M. Turk and A. Pentland. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3(1): 71-86, 1991