

Face Recognition Using Long Haar-like Filters

Y. Higashijima¹, S. Takano¹, and K. Nijima¹

¹Department of Informatics, Kyushu University, Japan.

Email: {y-higasi, takano, nijima}@i.kyushu-u.ac.jp

Abstract

A face recognition method based on long Haar-like filters is proposed. This filter takes a vector form whose components consist of 1 and -1 . Although the learning of the filter is basically by the principal component analysis (PCA), our approach lies in constructing a highpass filter with 1 and -1 as filter coefficients. This approach yields a combinatorial optimization problem. Since the problem is NP-hard, an alternative method for finding approximate solutions is provided. Using the learned long Haar-like filters, the feature vectors of training and query images are constructed. The similarity between the query and the training images is measured by using the cosine distance.

Keywords: Face recognition, long Haar-like filter, learning, feature vector, cosine distance

1 Introduction

There are many approaches to face recognition such as the template matching technique [1], the principal component analysis (PCA) [2], the graph matching method [3], the self-organizing map [4], the support vector machine [5] and so on. Among them, PCA is an excellent technique for sparse representation of given data, which is useful for pattern recognition. Principal components computed from several images by PCA yield lowpass images. This means that PCA approach aims at capturing the features of the images by their lowpass components. On the other hand, there is a thought that the features of images are contained in their highpass components.

In this paper, we propose a face recognition method based on long Haar-like filters. This filter is a highpass filter having a vector form whose components consist of 1 and -1 . We exploit the long Haar-like filter as a tool for extracting the features of facial images. The main purpose of this paper is to propose a learning algorithm of the long Haar-like filters for face recognition. As PCA technique, we maximize the variance of facial images. However, restrictions, under which the variance is maximized, are different from those in PCA. Our approach yields a combinatorial optimization problem. Since the problem is NP-hard, we provide an alternative method for seeking approximate solutions of the problem. This method gives a learning algorithm of long Haar-like filters. Using the learned long Haar-like filters, we construct the feature vectors of training and query images. The similarity of the query

and the training images is measured by using the cosine distance.

This paper is organized as follows: Section 2 describes a face extraction method. In section 3, an algorithm for learning long Haar-like filters is presented. We provide a face recognition method in section 4. Section 5 is a simulation. We close in section 6 with concluding remarks and future work.

2 Face extraction

For face recognition, we need to extract faces from color images in advance. The method extensively used for human face localization is the skin color segmentation algorithm [6]. In this paper, we employ this algorithm for face extraction.

Color images are typically represented in RGB space. We first transform RGB components into YCrCb components. And then, we find a region of the skin in an input image and choose the range of Cr and Cb values included in this area, i.e., $Cr_{\min} < Cr < Cr_{\max}$ and $Cb_{\min} < Cb < Cb_{\max}$. Based on the range of Cr and Cb values, we build a face color model. Skin color segmentation is performed by checking whether or not the pixels of the input image fall within the face color model. We consider the centroid of the skin area as the central point of a face in the input image, and extract a rectangular domain around the central point as a candidate of faces. An example of extracted faces is shown in figure 1.

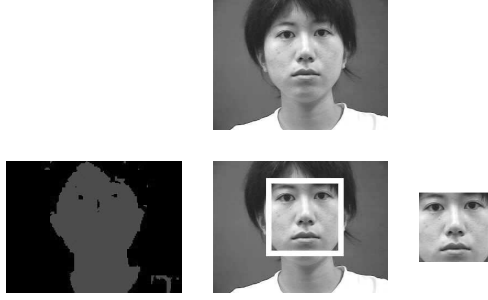


Figure 1: Face extraction

3 Learning algorithm

3.1 Long Haar-like filters

Haar filter takes the form $(1, -1)^T$, which can be considered as a two-dimensional vector, where T denotes transposition. We generalize Haar filter a filter having the form of M -dimensional vector $(g_1, g_2, \dots, g_M)^T$, where $g_i = 1$ or -1 . We call this generalized filter a long Haar-like filter. The original Haar filter is a highpass filter, because $1 + (-1) = 0$. If the long Haar-like filter satisfies the condition

$$\sum_{i=1}^M g_i = 0, \quad (1)$$

then it becomes a highpass filter. However, this condition is too strong for our purpose, so the following alternative condition weaker than (1) is put for later analysis.

$$\sum_{i=1}^M g_i \approx 0. \quad (2)$$

3.2 Learning of long Haar-like filters

By using the face extraction method described in section 2, we construct N squared training facial images for learning long Haar-like filters. At first, we divide each of the images into $M \times M$ squared blocks, each of which has $m \times m$ size, as shown in figure 2.

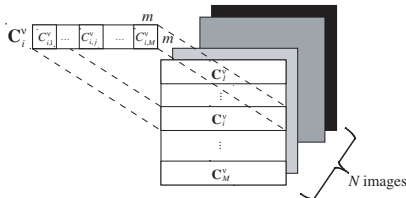


Figure 2: Low frequency image

We compute the sum of elements contained in a block in the ν -th facial image, and construct a low frequency image $C_{i,j}^\nu, i, j = 1, \dots, M$. This image can be considered as a set of M row vectors. We represent the i -th row vector by a column vector

$$\mathbf{C}_i^\nu = (C_{i,1}^\nu, C_{i,2}^\nu, \dots, C_{i,M}^\nu)^T.$$

Let us define a long Haar-like filter \mathbf{g}_i by

$$\mathbf{g}_i = (g_{i,1}, g_{i,2}, \dots, g_{i,M})^T.$$

The inner product of \mathbf{g}_i and \mathbf{C}_i^ν takes the form

$$\mathbf{g}_i^T \mathbf{C}_i^\nu = \sum_{j=1}^M g_{i,j} C_{i,j}^\nu.$$

We introduce the variance of $\mathbf{g}_i^T \mathbf{C}_i^\nu, \nu = 1, \dots, N$:

$$\sigma_i^2 = \frac{1}{N} \sum_{\nu=1}^N (\mathbf{g}_i^T (\mathbf{C}_i^\nu - \bar{\mathbf{C}}_i))^2, \quad (3)$$

where

$$\bar{\mathbf{C}}_i = \frac{1}{N} \sum_{\nu=1}^N \mathbf{C}_i^\nu.$$

The variance σ_i^2 can be rewritten as

$$\sigma_i^2 = \mathbf{g}_i^T \mathbf{V}_i \mathbf{g}_i. \quad (4)$$

Here \mathbf{V}_i denotes the co-variance matrix, and is given by

$$\mathbf{V}_i = \frac{1}{N} \sum_{\nu=1}^N (\mathbf{C}_i^\nu - \bar{\mathbf{C}}_i)(\mathbf{C}_i^\nu - \bar{\mathbf{C}}_i)^T. \quad (5)$$

Our method for learning long Haar-like filters is to maximize the variance (4) in M -dimensional binary vector space $S = \{(g_1, g_2, \dots, g_M) \mid g_i = 1 \text{ or } -1\}$. This learning is accomplished for each of $i = 1, \dots, M$. The proposed method is basically the same as PCA approach. But now, the present problem is a combinatorial optimization problem that finds a solution among 2^M combinations of binary vectors. Since this problem is an NP-hard problem, we try to obtain a real-vector solution which approximates a binary-vector solution under the condition (2). Thus, we arrive at the following minimization problem:

$$J_i = -\mathbf{g}_i^T \mathbf{V}_i \mathbf{g}_i + \frac{K_1}{2} \left(\sum_{j=1}^M g_{i,j} \right)^2 + \frac{K_2}{4} \sum_{j=1}^M (g_{i,j}^2 - 1)^2 \rightarrow \min. \quad (6)$$

Here K_1 and K_2 are penalty constants. The K_1 -term comes from the condition (2), and the K_2 -term from the approximation of binary-vector solutions.

The minimization problem (6) may have many local minima. It is difficult to obtain a global minimum. To seek local minima of (6) fast, we employ Newton's method. To apply Newton's method, we

differentiate the functional J_i in (6) with respect to $g_{i,j}$ and derive a nonlinear system of simultaneous equations as

$$\frac{\partial J_i}{\partial g_{i,j}} = 0, \quad j = 1, \dots, M. \quad (7)$$

We solve (7) by using Newton's method starting from an M -dimensional random vector. The solution is approximated by a binary-valued vector which is employed as a long Haar-like filter. Solving (7) for $i = 1, \dots, M$, we can obtain a set of M long Haar-like filters. For face recognition, we memorize the computed long Haar-like filters and the feature vectors obtained by applying them to the training images $C_{i,j}^\nu, \nu = 1, \dots, N$, in a database.

3.3 Comparison with PCA

PCA approach for learning the filter \mathbf{g}_i is to determine so as to maximize (4) under the constraint $\mathbf{g}_i^T \mathbf{g}_i = 1$. PCA method computes the principal component vectors by solving an associated eigen-value problem. The obtained components are real-valued vectors, each of which involves low frequency elements. This implies that lowpass filters are computed from given data in PCA. In PCA method, a feature vector for input images is obtained by expanding the image with respect to the principal component vectors. However, our idea lies in calculating a highpass filter which is a sequence of 1 and -1 so as to maximize (4). In our approach, a feature vector has components, each of which is an inner product of the long Haar-like filter with the row vector of the image. The computation of the inner product doesn't require the multiplication of the two vectors, because our filter consists of 1 and -1 .

4 Face recognition

We first extract a squared facial image from a query image using the face extraction method described in section 2. Next, an image $C_{i,j}$ of $M \times M$ size is constructed from the facial image based on the method mentioned in section 3.2. From the image $C_{i,j}$, we construct the vectors $\mathbf{C}_i = (C_{i,1}, C_{i,2}, \dots, C_{i,M})^T$. We apply the long Haar-like filters \mathbf{g}_i stored in the database to \mathbf{C}_i to get the inner products $q_i = \mathbf{g}_i^T \mathbf{C}_i$. Arranging them, we construct a feature vector $\mathbf{q} = (q_1, q_2, \dots, q_M)^T$ for the query image. Let $\mathbf{p}^\nu = (p_1^\nu, p_2^\nu, \dots, p_M^\nu)^T$ with $p_i^\nu = \mathbf{g}_i^T \mathbf{C}_i^\nu$ denote the feature vectors for training images, which were memorized in the database. Face recognition is accomplished by measuring the cosine distance between \mathbf{q} and \mathbf{p}^ν :

$$\cos \theta^\nu = \frac{\mathbf{q}^T \mathbf{p}^\nu}{\|\mathbf{q}\| \|\mathbf{p}^\nu\|}. \quad (8)$$

We compute (8) for $\nu = 1, \dots, N$, and choose the index ν such that $1 - \cos \theta^\nu$ is minimum. It is recognized that the query facial image is the most similar to the ν -th training facial image. We show our face recognition system in figure 3.

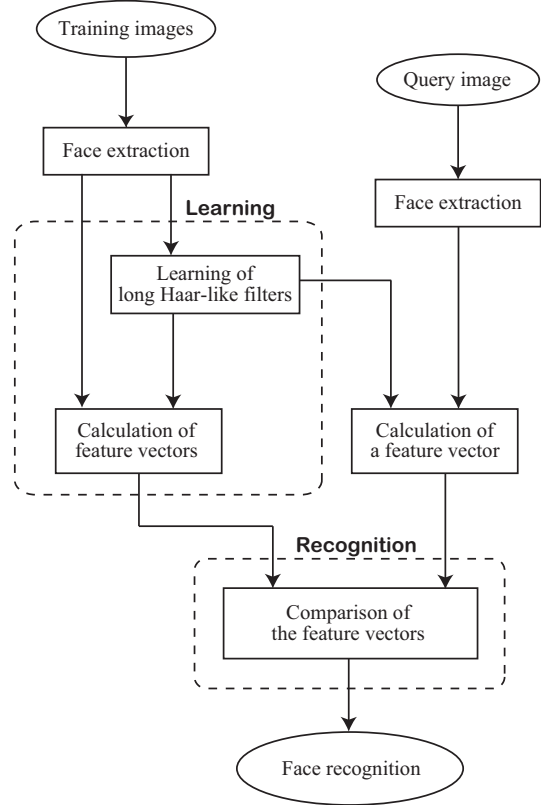


Figure 3: Face recognition system

5 Simulation

For simulation, the images of ten persons are captured by using a network camera. We extract a squared facial image from each of them exploiting the skin color segmentation method described in section 2, and normalize it to the 128×128 size. The extracted facial images are memorized in a database, which are illustrated in figure 4.



Figure 4: Facial images in the database

Training images are constructed by dividing each of the images shown in figure 4 into 16×16 squared

blocks and by computing the average of image in each block. We learn long Haar-like filters using these training images by solving the minimization problem (6). The functional J_i in (6) has many local minima. These minima change depending on the index i , the penalty constants K_1 and K_2 , and starting points of Newton’s iteration. Newton’s method is not always convergent to a local minimum. To get convergence results, we have to choose the penalty constant K_2 depending on the initial values of Newton’s iteration. As a starting point of Newton’s method, we chose a vector whose components are random numbers in the interval $[-10, 10]$. The filters learned for each i were arranged in order of the magnitudes $\mathbf{g}_i^T \mathbf{V}_i \mathbf{g}_i$. This process is performed for all i . We construct a filter pattern (FP) by collecting the same order of the learned filters. The first five learned filter patterns are shown in figure 5, in which a white cell means $g_{i,j} = 1$, and a black cell $g_{i,j} = -1$. In the learning



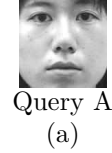
Figure 5: The first five learned filter patterns


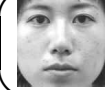




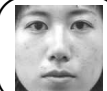













of our filters, we chose the penalty constant K_1 as $K_1 = 500$ and changed the penalty constant K_2 depending on the initial values of Newton’s iteration. Actually, we selected K_2 as in table 1.

Table 1: List of K_2

	FP1	FP2	FP3	FP4	FP5
1	20	30	30	40	40
2	30	30	30	40	40
3	30	40	40	40	50
4	20	30	40	50	50
5	30	30	30	60	30
6	30	60	60	60	60
7	20	20	20	20	50
8	50	40	50	40	40
9	40	30	50	70	50
10	40	20	30	40	30
11	40	40	40	40	40
12	30	30	40	40	40
13	30	30	40	40	50
14	30	40	40	50	50
15	30	40	50	50	50
16	20	30	30	40	40

Applying the learned filter patterns to the images in the database and query A shown in figure 6(a), we compute their feature vectors and perform the experiment of face recognition. Figure 6(b) gives a recognition result.



Filter patterns	First	Second	Third
 FP1	 0.244126	 0.333890	 0.459099
 FP2	 0.283315	 0.297625	 0.649774
 FP3	 0.507954	 0.549734	 0.721386
 FP4	 0.187160	 0.292289	 0.521576
 FP5	 0.427895	 0.536759	 0.673354

(b)

Figure 6: Recognition results for query A

Figure 6(b) shows the images in the database, which are arranged in small-scale order of the similarities $1 - \cos \theta'$.

To evaluate what image in the database is the most similar to the query A, we compute the total of the five similarities for the same person in figure 6(b), which is shown in figure 7.


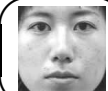


	First	Second	Third
 Query A	 1.773624	 1.887123	 3.11331

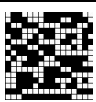


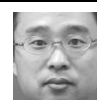



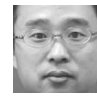
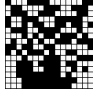


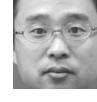








Figure 7: Evaluation of the similarity for query A

Figure 7 shows that query A is recognized as the same person in the database. We notice that query A is a different image from the selected one, although they represent the same person.

To confirm the efficiency of our recognition method, we carry out a few experiments of recognition using the other queries.

The second example is an examination for query B, which is illustrated in figure 8.



Filter patterns	First	Second	Third
 FP1	 0.169191	 0.591721	 0.747832
 FP2	 0.164674	 0.620391	 0.706100
 FP3	 0.131163	 0.622143	 0.652802
 FP4	 0.186084	 0.553077	 0.799091
 FP5	 0.143671	 0.678190	 0.779417

(b)

Figure 8: Recognition results for query B

Although the person of query B wears the glasses, our system can recognize him as the same person who doesn't wear the glasses for all the filter patterns.

Figure 9 is an evaluation of the similarity for query B. It is seen from the similarities that query B is very close to the face of the same person in the database.





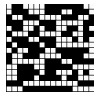



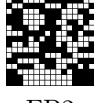



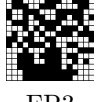











	First	Second	Third
 Query B	 0.794783	 3.338001	 3.449359

Figure 9: Evaluation of the similarity for query B

The third example is an experiment for query C. The recognition results are shown in figure 10. The person of query C turns left a little bit. However, recognition is succeeded for all the filter patterns except for the fourth.



Filter patterns	First	Second	Third
 FP1	 0.363010	 0.428798	 0.685171
 FP2	 0.409966	 0.423281	 0.662733
 FP3	 0.439984	 0.512103	 0.674931
 FP4	 0.368652	 0.632859	 0.664244
 FP5	 0.456235	 0.490182	 0.698575

(b)

Figure 10: Recognition results for query C

Figure 11 evaluates the similarity of query C.

	First	Second	Third
 Query C	 2.223016	 2.302054	 3.870239

Figure 11: Evaluation of the similarity for query C

The total value for the target person is bigger than that of another person. This is due to the large value of the distance from the same person when the fourth filter pattern in figure 10 is employed.

Table 2 is a list of time necessary for extracting one face, constructing five filter patterns, computing ten feature vectors and comparing similarities.

Table 2: Computational time for face recognition

Face extraction		0.052 sec
Construction of five filter patterns		2.06 sec
Computation of ten feature vectors		
	FP1	0.13 sec
	FP2	0.17 sec
	FP3	0.16 sec
	FP4	0.17 sec
	FP5	0.17 sec
Comparison of similarities		
	FP1	0.024 sec
	FP2	0.026 sec
	FP3	0.026 sec
	FP4	0.027 sec
	FP5	0.024 sec

All the experiments were performed on a personal computer with Power PC G4, 1.33 GHz and 1.25 GB DDR SDRAM.

6 Conclusion

We proposed a face recognition method using long Haar-like filters. Our method generates a high-pass filter in the binary-vector form with the help of PCA technique. Since this approach yields a combinatorial optimization problem, which is NP-hard, we approximate the problem by a minimization problem in a real-valued space. To compute approximate solutions of the minimization problem fast, we utilize Newton’s method. For simulation, ten training facial images were constructed from the images captured by a web camera. We generated long Haar-like filters using these images. The experiments of face recognition were performed for three query images. Recognition results are satisfactory from the viewpoints of accuracy and time. However, the number of training images is small in the present case. It remains to examine the recognition ability of long Haar-like filters learned from many training images. Our method is not robust for changing viewing conditions. These are future works.

References

- [1] V. Govindaraju, “Locating human faces in photographs,” *International Journal of Computer Vision*, vol. 19, no. 2, pp. 129–146, 1996.
- [2] W. Zhao, R. Chellappa, and A. Krishnaswamy, “Discriminant analysis of principal components for face recognition,” in *Proceedings of the Third International Conference Automatic Face and Gesture Recognition*, pp. 336–341, 1998.
- [3] L. Wiskott, J.M.Fellous, N.Krueger, and C. von der Malsburg, “Face recognition by elastic bunch graph matching,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 775–779, 1997.
- [4] M.-H. Yang, N. Ahuja, and D. Kriegman, “Mixtures of linear subspaces for face detection,” in *Proceedings of the Fourth International Conference Automatic Face and Gesture Recognition*, pp. 70–76, 2000.
- [5] C. Papageorgiou and T. Poggio, “A trainable system for object detection,” *International Journal of Computer Vision*, vol. 38, no. 1, pp. 15–33, 2000.
- [6] D. Chai and K. N. Ngan, “Locating facial region of a head-and-shoulders color image,” in *Proceedings of the Third International Conference Automatic Face and Gesture Recognition*, pp. 124–129, 1998.