Training Algorithms for Robust Face Recognition using a Template-matching Approach

Xiaoyan Mu, Mehmet Artiklar,
Metin Artiklar, and Mohamad H. Hassoun

Department of Electrical and
Computer Engineering
Wayne State University
Detroit, MI 48202

Hassoun@brain.eng.wayne.edu

Paul Watta

Department of Electrical and
Computer Engineering
University of Michigan-Dearborn
Dearborn, MI 48128

watta@umich.edu

Abstract

This paper describes a complete face recognition system. The system uses a template matching approach along with a training algorithm for tuning the performance of the system to solve two types of problems simultaneously: 1. Correct classification experiments: Correctly recognize and identify individuals who are in the database and 2. False positive experiments: Reject individuals who are not part of the database. Experimental results are given which indicate that this training method is capable of consistently producing high correct classification rates and low false positive rates.

1. Introduction

The task here is to design a face recognition system which gives good classification results on 2 different types of recognition problems simultaneously:

1. **Correct classification experiments**. Correctly recognize and identify individuals who are in the database.

2. **False positive experiments**. Reject all images of individuals who are not part of the database.

Many face recognition algorithm designs are trained and tuned to perform well on one of these face recognition problems. The training algorithm described in this paper, though, solves both problems simultaneously.

In this paper, the proposed training algorithm is used with a simple nearest neighbor template matching classifier. Even so, the experimental results we obtained are better than our previous published results, where we used a variety of different classifiers, such as eigenfaces, Fisherfaces, wavelets, etc. (Artiklar, Hassoun, and Watta, 1999).

In the remainder of this paper, we describe how the template matching classifier works and how the output of the classifier is computed. Next, we describe a training method whereby the parameters of the classifier system can be determined from the available training face images. We describe the database of face images that was used in these experiments. Finally, we present and analyze the experimental results.

2. Classification and Decision Algorithms

In this section we describe a decision algorithm that can be used to determine the final output of a template matching-based face recognition (associative memory) system. The decision algorithm involves parameters (thresholds) which can be computed from the given database of face images.

The face recognition problem is stated as follows: Given a database of face images and an input image, the face recognition system must decide on one of the following:

(1) the input image is part of the image database (and identify the best matching individual)

(2) the input is not part of the database and hence the image should be rejected.

Suppose the database of face images of the M known individuals is denoted DB. And suppose that DB

contains K different samples (face images) of each individual (for example, different facial expressions or lighting conditions, etc.):

$$DB = \{\mathbf{I}_{mk} : m = 1, 2, \ldots, M;\ k = 1, 2, \ldots, K\}$$

Some algorithms use the pixel data directly, and some algorithms use other types of representations of the image data (typically to reduce system dimensionality). But once the representation is fixed, each image in the database is mapped to the chosen coordinates:

$$X = \{\mathbf{x}_{mk} = \rho(\mathbf{I}_{mk}) : m = 1, 2, \ldots, M;\ k = 1, 2, \ldots, K\}$$

For example, for a nearest neighbor classifier which operates directly on the pixel data, ρ is simply an identity map. For the eigenfaces method (Turk and Pentland, 1991; Belhumeur, et al., 1997), ρ maps each database image onto the corresponding eigenface coordinates.

To compute the output of the nearest neighbor classifier, we first need a way of measuring the distance between two images (for example, the Euclidean distance). Suppose we denote the distance between images $\mathbf{x}$ and $\mathbf{y}$ by $d(\mathbf{x}, \mathbf{y})$.

The design of the nearest neighbor classifier is simple: we simply store all of the images[1] in X into memory. Then, given an input image $\mathbf{I}$, we first apply ρ to compute its coordinates in the chosen representation: $\mathbf{x} = \rho(\mathbf{I})$. Then we compute the distance between $\mathbf{x}$ and each of the database samples and determine the closest sample $\mathbf{x}_{m^*k^*}$:

$$d(\mathbf{x}_{m^*k^*}, \mathbf{x}) \le d(\mathbf{x}_{mk}, \mathbf{x})$$

for all $m = 1, \ldots, M$ and $k = 1, \ldots, K$.

In a classical nearest neighbor classifier, the output would then be taken as m^*, the individual whose image is closest to the input. In order to address the issue of rejecting individuals which are not in the database (and thus design a useful and practical recognition system), a more useful decision rule might use a threshold to decide whether or not to identify the image or reject it:

$$output = \begin{cases} m^* \text{ if } d(\mathbf{x}_{m^*k^*}, \mathbf{x}) \le T \\ \text{otherwise, reject} \end{cases}$$

Results using this type of decision rule can be found in (Artiklar, Hassoun, and Watta, 1999).

A more sophisticated decision algorithm would involve determining and storing a different threshold for each individual in the database. In this case, the only modification needed in the decision rule above would be to replace T by T_{m^*}, where T_{m^*} is the threshold for the m^*th image (the best matching image).

The decision rule that we propose involves computing and storing two different thresholds for each individual in the database: a *lower threshold* L_m and an *upper threshold* U_m, $m = 1, 2, \ldots, M$:

$$output = \begin{cases} m^* \quad \text{if } d(\mathbf{x}_{m^*k^*}, \mathbf{x}) \le L_{m^*} \\ \text{reject if } d(\mathbf{x}_{m^*k^*}, \mathbf{x}) > U_{m^*} \\ \text{apply heuristic if } L_{m^*} \le d(\mathbf{x}_{m^*k^*}, \mathbf{x}) \le U_{m^*} \end{cases}$$

The lower threshold is used to determine when there is a sufficiently close match between the input $\mathbf{x}$ and the closest sample image $\mathbf{x}_{m^*k^*}$, such that the input can be reliably identified as individual m^*. The upper threshold is used to determine when there is a sufficient mismatch between $\mathbf{x}$ and all of the sample images, even the closest one $\mathbf{x}_{m^*k^*}$, such that the input should be rejected as not known to the system. The only other possibility is when the minimum distance falls between the two thresholds: $L_{m^*} \le d(\mathbf{x}_{m^*k^*}, \mathbf{x}) \le U_{m^*}$. In this case, we propose the following heuristic:

> Identify the input as person m^* if the second best matching image is also a sample of individual m^*;
>
> otherwise, reject the input as not known to the system.

A schematic diagram of the proposed decision rule is shown in Figure 1.

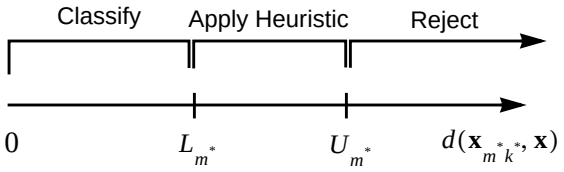

Figure 1. The thresholds used for the decision stage.

1. Technically the elements of X are not images, but are representations of the database images. However, to avoid continually distinguishing between an image vector and its representation, we will refer to both $\mathbf{I}$ and $\mathbf{x}$ as images.

3. Training Phase for Threshold Computation

This section describes one way to adaptively compute the upper and lower thresholds L_m and U_m, $m = 1, 2, \ldots, M$ required in the decision algorithm described in the previous section.

Here we require two data sets: a *classification training set X* and a *false positive training set Y.* The classification training set consists of samples of the people in the database, and is used to tune the identification and classification capability of the system. The false positive training set consists of samples of individuals who are not in the database, and is used to tune the rejection capability of the system.

We first partition the classification database into two disjoint sets: X_1 and X_2. We require that both X_1 and X_2 contain samples of all the individuals in the database. So assuming K samples (K even) of each individual, then

$$X_1 = \{\mathbf{x}_{mk} \in X : m = 1, 2, \ldots, M;\ k = 1, 2, \ldots, K/2\}$$

$$X_2 = \{\mathbf{x}_{mk} \in X : m = 1, 2, \ldots, M;\ k = K/2 + 1, \ldots, K\}$$

is a valid partition.

Next, in-class distances (i.e., distances between X_1 and X_2 images) are computed for each individual in the database. For a fixed individual m, we select an image $\mathbf{x}_{mk}$ from X_1 and then compute the distance between $\mathbf{x}_{mk}$ and each image of person m in X_2. This process is repeated for all such images of person m in X_1. The average of all these distances is computed and this value is set as an approximate lower threshold: $\tilde{L}_m$ for individual m.

Typically, the distance calculations are affected by any post processing (see next Section) computations that are done. Therefore, it is sometimes advantageous to apply the post-processing in the training phase, as well as in the testing phase. For all the training and testing experiments reported here, Euclidean distances are always measured after the shifting algorithm (as described in the next Section) has been applied.

To compute the upper threshold, between-class distances (i.e., distances between X and Y images) are computed. For each image $\mathbf{x}_{mk}$ in X, we compute the distance between $\mathbf{x}_{mk}$ and each of the images in the false positive training set *Y.* The minimum such distance is recorded as an approximate upper threshold: $\tilde{U}_m$ for individual m.

Note that the approximate thresholds $\tilde{L}_m$ and $\tilde{U}_m$ are right on the border-line of the decision regions. For example, for person 1, we know for certain that there was a training image of some other person that had a distance of $\tilde{U}_1$. To allow for a little robustness, we tighten the thresholds a bit by scaling the thresholds using a common scaling factor: $L_m = \alpha\tilde{L}_m$ and $U_m = \alpha\tilde{U}_m$. We found that $\alpha = 0.8$ provides good results.

Note that it is possible to avoid creating a separate false positive training set and just use the classification data set. Here, for an individual m and image $\mathbf{x}_{mk}$, to compute the between class distances, we simply compute the distance between $\mathbf{x}_{mk}$ and each other image in the classification data set, excluding all those images of individual m. So in this case, the false positive set consists of a sequence of 1-deleted training sets; that is, a training set with one individual removed.

Since the computations for $\tilde{L}_m$ and $\tilde{U}_m$ are done independently, it may happen that $U_m < L_m$. In these rare case, another heuristic is used: the upper threshold is simply set to 20% higher than the lower threshold: $U_m = 1.2L_m$.

4. Post-processing

Noise is always present in any practical image capturing system. There are several ways to make the nearest neighbor classifier less prone to errors in small shifts (translations) and rotations in the image. For example, in the elastic band matching, the distance between the input and each of the database images is computed on a rectangular grid of points and neighbors of the grid points (Duc and Fischer, 1999). The collection of grid points which yields the smallest distance is chosen (deformed template).

In previous work, we developed a simple and computationally efficient method which provides invariance to small amounts of image shift (Artiklar, Hassoun, and Watta, 1999). The method involves determining an optimal sequence of shifts to reduce the Euclidean distance between two images. The optimal path is determined using a greedy algorithm of locally moving in the best direction at each step.

A schematic diagram of such a path is shown in Figure 2. In this table, each number represents the Euclidean distance (expressed in percent) between two images if one of the images is shifted by a certain number of pixels.

The center of the table (with value 17.7%) is the Euclidean distance between the two images with no shift. As the input image is shifted 1 pixel down, the distance drops to 13.9%. Shifting the input another pixel in the downward direction results in a distance of 10.4%. At this point, the distance can no longer be decreased by shifting in any of the 8 surrounding directions, and so the algorithm terminates.

Note that it is pointless to try to shift images that are very dissimilar from each other. Hence, in our method, we only allow the top 40 matches to participate in the shifting process.

29.4	28.5	27.5	27.3	27.3	28.5	29.4	30.5	31.5
27.9	27.0	25.8	25.4	25.6	26.4	27.8	29.1	30.6
27.0	25.4	24.2	23.5	23.7	25.0	25.9	27.6	29.9
25.8	23.7	22.0	21.4	21.4	22.8	24.5	26.8	29.2
25.0	22.3	19.6	18.0	17.7	20.1	22.5	25.4	27.6
24.0	21.2	17.2	14.2	13.9	17.0	21.0	24.4	27.1
23.7	19.7	15.7	11.3	10.4	15.8	20.4	23.2	26.2
24.2	20.8	16.9	12.9	12.8	16.6	20.4	23.5	26.9
25.4	22.8	19.8	17.5	17.0	19.2	22.0	25.2	27.9

Figure 2. This diagram shows the Euclidean distance (in percent) between a database image and shifted versions of an input image.

5. The Database of Face Images

A detailed discussion of the construction of the face database can be found in (Watta, Artiklar, Masadeh, and Hassoun, 2000). Briefly, the database consists of 680 different men and women of various ethnic backgrounds and ages (between 15 and 50 years old). The images were collected in a laboratory setting so as to minimize the amount of preprocessing that is necessary in order to eliminate complicating effects, such as tilt, rotation, shifting, scaling, and changes in illumination. For each person, two groups of 4 images were snapped, giving a total of 8 training images per person. The first group of 4 images all show a blank facial expression. The second group of 4 images show different facial expressions: blank, smile, angry, and surprised.

Two test images were also collected for each person: a blank image and an arbitrary image, where the subject gives an unusual expression which might fool the recognition system.

Both the database and test images were snapped at a dimension of 82×115 and stored as 8-bit gray scale (256 levels). The 82×115 images were cropped to a size of 72×72 (using a fixed cropping position) in order to eliminate the hair of the person from the image. Other than cropping, the only other preprocessing performed is intensity normalization.

Figure 3 shows a set of 72×72 sample images for one of the subjects in the database. In (a), the 4 blank images are shown in (b), the 4 facial expression images are shown: blank, smile, angry, and surprised, and in (c), the two test images are shown: blank and arbitrary.

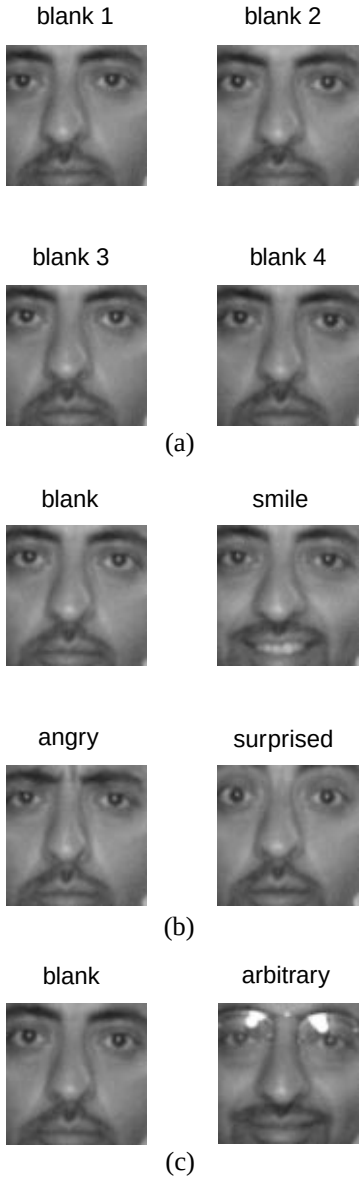

Figure 3. Sample images from the database showing (a) the 4 blank expression images, (b) the four different expressions: blank, smile, angry, surprised, and (c) the two test images.

6. Experimental Results

Here, the training set X consists of all 8 training images for each person in the database. For training, X is partitioned as follows: X_1 consists of the 4 blank expressions for each person, and X_2 consists of the 4

facial expression images. After training, is completed, it is only X_2 that is stored in memory, and so X_1 is used only during training. Note that no separate false positive training set *Y* was used in these experiments. Rather, *Y* consisted of the 1-deleted training sets, as described in Section 3.

After training, the system was tested for both correct classification performance and false positive performance. To see how the performance scales with increasing number of subjects in the database, we looked at the performance of the system as the number of people in the database varied from 100 to 600 individuals.

Table 1 shows the results of the correct classification experiments for both the blank and arbitrary test images. Note that the each number in Table 1 is an average over 3 different trials of the experiment. In each trial, a different database was (randomly) chosen.

Num People	**Blank**		**Arbitrary**		**Train Time** (min)
	Reject	Error	Reject	Error	
100	4.3	0	37.3	0	14
200	3.0	0	39.7	0	30
300	4.0	0.3	37.3	0	46
400	4.8	0.3	43.5	0	63
500	4.3	0.4	43.1	0	83
600	5.7	0.3	45.0	0	106

Table 1. Correct classification results for the blank expression test set and the arbitrary expression test set for various number of people in the database.

The results show that the algorithm scales well and for the blank test images gives an error rate of less than 1% with a reject rate of about 5%. As expected, for the arbitrary test images, the system rejects a much larger number: 40%. However, on this test set, there are no classification errors.

Table 2 shows the results of the false positive experiments. Here, the error rate is (roughly) slightly higher than 1%, and all other images are properly rejected by the system. Of course it is possible to tighten the thresholds so that the false positive rate is always 0%. But the price to be paid is a corresponding increase in the number of rejected images in the correct classification experiments.

Num People	**False Positives** %
100	1.5
200	1.2
300	1.1
400	1.5
500	1.3
600	0.6

Table 2. False positive results for various number of people in the database.

Figure 4 shows how the training time scales as a function of the number of people in the database. Clearly, the training scales linearly with the number of people in the database.

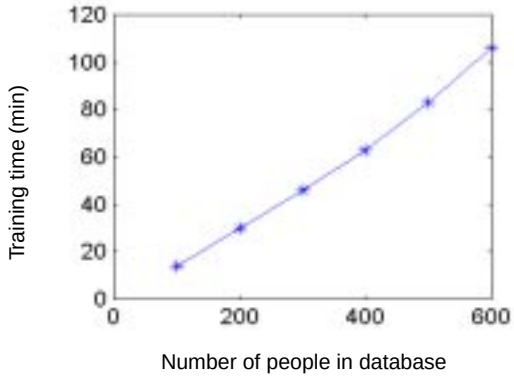

Figure 4. Training time (in minutes) vs. the number of people in the database.

7. Summary

This paper described a simple training algorithm that can be used in a practical face recognition system. Note that this training method is a general purpose algorithm and is not strictly limited for use with face recognition problems. In fact, the training makes no assumptions about the nature of the patterns to be classified.

When using the proposed training algorithm on a large 600-person human face recognition problem, the results indicate that the system can achieve a correct classification rate of over 99%, with a rejection rate of about 5%, and a false positive rate of less than 1.5%.

In future work, we will compare the results of this classifier with other pattern recognition systems (Duda, Hart, and Stork, 2000), as well as other neural net (Hassoun, 1995), and associative memory models (Hassoun, 1993). In particular, we will formulate an RCE-based classifier and a wavelet-based classifier. In addition, we will investigate distance measures other than the Euclidean metric, and other types of post-processing.

Acknowledgments

This work was supported by the National Science Foundation (NSF) under contract ECS-9618597.

References

1. Artiklar, M., Hassoun, M., and Watta, P. (1999). "Application of a Postprocessing Algorithm for Improved Human Face Recognition," *Proceedings of the IEEE International Conference on Neural Networks*, IJCNN-1999, July 10-16, 1999, Washington, DC., Paper #JCNN 2166.

2. Belhumeur, P., Hespanha, J., and Kreigman, D. (1997). "Eigenfaces vs. Fisherfaces: Recognition using Class Specific Linear Projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **19**(7), 711-720.

3. Duc, B. and Fischer, S. (1999). "Face Authentication with Gabor Information on Deformable Graphs," *IEEE Transactions on Image Processing*, **8**(4).

4. Duda, R., Hart, P., and Stork, D. (2000). *Pattern Classification*, Second Ed., John Wiley & Sons, New York.

5. Hassoun, M. H., ed. (1993). *Associative Neural Memories: Theory and Implementation*. Oxford University Press, New York.

6. Hassoun, M. H. (1995). *Fundamentals of Artificial Neural Networks*, MIT Press, Cambridge, Mass.

7. Turk, M., and Pentland, A. (1991). "Eigenfaces for Recognition," *J. Cognitive Neuroscience*, **3**(1), 71-86.

8. Watta, P., Artiklar, M., Masadeh, A., and M. H. Hassoun, (2000). "Construction and Analysis of a Database of Face Images which Requires Minimal Preprocessing," *Proceedings of the IASTED Conference on Modeling and Simulation*, MS-2000, May 15-17, 2000, Pittsburg Pennsylvania, 465-469.