Combining Gabor Features: Summing vs.Voting in Human Face Recognition*

Xiaoyan Mu and Mohamad H. Hassoun
Department of Electrical and Computer Engineering
Wayne State University
Detroit, MI 48202

muxiaoyan@wayne.edu hassoun@eng.wayne.edu

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

watta@umich.edu

Abstract - *Gabor wavelet-based feature extraction has been emerging as one of the most promising ways to represent human face image data. In this paper, we examine the performance of two types of classifiers that can be used with Gabor features. In the first classifier, the distance between two images is computed by summing the local distances among all the nodes. In the second classifier, a voting strategy is used. In addition, we examine two types of shift optimization procedures. The first is the standard elastic graph matching algorithm, and the second is a constrained version of the algorithm. Experimental results indicate that the voting-based classifier with constrained elastic graph matching gives improved results.*

Keywords: Face recognition, Gabor transform, voting, classification, local features.

1. Introduction

In the automated face recognition problem, we are given a database of image samples of known individuals. Suppose the database is denoted **DB** and contains M people and K samples per person:

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

Here, $\mathbf{I}_{mk}$ is the kth image sample of individual m, and ID_m is the name of the mth person.

The task is to design a system such that for any input image, the system either identifies the input with one of the known individuals, or else rejects the input as not known to the system.

There are two main parts to the face recognition problem: feature extraction and classification. In the *feature extraction* problem, the task is to find an efficient way to represent the pixel data. This involves designing a function to map from the original space of images to another (typically lower dimensional) space of feature vectors. The feature extraction is applied to all database images, yielding a set of feature vectors:

$$\mathbf{DB\text{-}M} = \{(\mathbf{x}_{mk}, ID_m): m = 1, 2, \ldots, M;\ k = 1, 2, \ldots, K\}$$

Here, $\mathbf{x}_{mk}$ is the feature vector derived from image $\mathbf{I}_{mk}$. Since **DB-M** is typically much smaller than **DB**, it requires less storage and processing time, especially for template matching-based classification.

The *classification* problem involves designing a function to map feature vectors to the appropriate class label. As mentioned above, in addition to the class labels of the M known people: $ID_1, ID_2, \ldots, ID_M$, the system must be capable of rejecting the input. It is important to remember that although the database may contain hundreds, thousands, or maybe even millions of people, the set of people who are not in the database (and have to be rejected) is always much larger; presently, that's about 6 billion people—all the rest of the people on the planet! The crux of the face recognition problem, as with all pattern recognition problems, lies in balancing the ability of the system to recognize known people with the ability to reject strangers.

Recent research results suggest that the Gabor transform can be used to efficiently extract features from face images [2, 3, 10]. In this paper, we will use Gabor-based features and examine the performance of two different template matching-based classifiers. The first classifier is the one typically used in the literature [11] and involves simply summing the local distance measures provided by the Gabor features. The second classifier uses a voting strategy and a higher level decision network to determine the output of the system [6]. We will show that the voting-based classifier provides improved performance over the summing-based classifier.

2 Gabor Features

Gabor filters are widely used in pattern recognition because they are robust to changes in size and orientation [1]. A 2-D Gabor function with scale parameters a and b (in the x and y directions, respectively), frequency ω and orientation φ is defined as:

$$G(\omega, \varphi) = e^{-\pi\left(\frac{x^2}{a^2}+\frac{y^2}{b^2}\right)} e^{-2\pi i \omega r \cos(\theta - \varphi)} \qquad (1)$$

Actually, G is really a function of four variables: $G(a, b, \omega, \varphi)$. However, we will consistently set the scale parameters a and b as:

$$a = b = \frac{1}{\omega}$$

The Gabor function is the multiplication of two parts: a Gaussian function and a Fourier-like function. The Fourier part gives information about the frequency and orientation of an image. This Fourier-type information, though, is scaled by the Gaussian weighting function. So pixels near the center $x = y = 0$ contribute more than pixels far from the center.

The Gabor transform is typically applied locally rather than globally over the entire face image. In this case, we create a grid of nodes, as shown in Figure 1(a). We compute a local Gabor transform centered at each node. To do this, we first select values for the parameters ω, and φ. A filter matrix **G** is created by sampling $G(\omega, \varphi)$ over the x and y variables. In our implementation, we created a filter matrix of size 7×7, and both x and y assume values in the range: $-3, -2, -1, 0, 1, 2, 3$ (the filter is centered on each node, so the node point corresponds to $x = y = 0$).

Next, the filter matrix **G** is convolved with the pixel data in the neighborhood of the node. The resulting value gives one component of the feature vector associated with the node. The process is repeated for several different values of the frequency ω and orientation φ. In our case, we used 5 different spatial frequencies: $\omega = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$ and 2 different orientations: $\varphi = 0, \frac{\pi}{2}$. Hence at each node, we compute 10 Gabor features.

Finally, the total feature vector **x** consists of all the Gabor features computed over all the nodes in the image. In our implementation, we used a grid of 13×13 nodes, hence each feature vector is 1690-dimensional ($13 \times 13 \times 10$).

3 Classification

3.1 Basic Template Matching

The basic template matching classifier operates as follows. First, given the database of images **DB**, we compute the set of corresponding Gabor feature vectors **DB-M**. This can be done in advance and hence is an off-line computation.

To classify a new input image **I**, we compute the corresponding feature vector **x**. The input feature vector **x** is compared to each of the feature vectors in **DB-M** using a suitable distance measure:

$$d_{mk} = d(\mathbf{x}, \mathbf{x}_{mk}) \qquad (2)$$

There are many distance functions that can be used. One of the most common is the class of Minokowski metrics, which for two N-dimensional vectors **s** and **t** is given by

$$d(\mathbf{s}, \mathbf{t}) = \left(\sum_{i=1}^{N} |s_i - t_i|^p \right)^{\frac{1}{p}}$$

Setting $p = 1$ results in the city-block distance, while $p = 2$ gives the Euclidean distance.

Finally, the smallest distance is found: $d_{m^*k^*} \le d_{mk}$. Typically, the rejection mechanism is implemented by applying a threshold T to the best distance. For example, if $d_{m^*k^*} \le T$, the input is sufficiently close to the stored samples to be identified as known person m^*. Otherwise, if $d_{m^*k^*} > T$, the input is rejected. If sufficient training data is available, a separate threshold can be stored for each known person: T_m.

3.2 Local Template Matching

We get more flexibility if we view the distance computation as arising from local calculations at the node level, rather than a global computation at the feature vector level. For example, one obvious benefit of using a local template matching approach is that the computation can be implemented in parallel using an array of processors (one processor allocated for each node).

Suppose we have W nodes in the grid. For the input feature vector $\mathbf{x}$, denote the local features as $\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(W)$; that is, $\mathbf{x}(i)$ is the set of features associated with node i. The database patterns are partitioned in the same way:

$$\mathbf{x}_{mk}(1), \mathbf{x}_{mk}(2), \ldots, \mathbf{x}_{mk}(W)$$

At each window i, we compute a *local distance* $d_{11}(i), \ldots, d_{MK}(i)$ between the input image and all the prototype patterns. The question is: how does one combine the local distance measures $d_{mk}(1), d_{mk}(2), \ldots, d_{mk}(W)$ in order to construct a global distance d_{mk} between the input $\mathbf{x}$ and prototype $\mathbf{x}_{mk}$? One obvious way is simply to add up all of the local distances:

$$d_{mk} = d_{mk}(1) + d_{mk}(2) + \ldots + d_{mk}(W) \qquad (2)$$

This metric, which we will call the *additive metric*, has been used frequently in the literature [11].

Besides the additive metric, another method that has started to receive increased attention is to use a voting strategy [5, 6], whereby each window i acts like a local nearest neighbor classifier and determines the database pattern $m^*k^*(i)$ which is closest and casts a vote for that pattern. Denote the number of votes received by each prototype pattern by v_{mk}. Then the total number of votes received by class m is given by:

$$v_m = \sum_{k=1}^{K} v_{mk}$$

This number of votes v_m gives a measure of the similarity (inversely proportional to the distance) between the input and class m. A max-select can be used to determine the person with the largest number of votes.

In order to provide a rejection capability, both the additive and voting metrics require a threshold. In the additive metric, the threshold T_a indicates whether the total distance (summed over the local windows) is too large for the input to be considered a known person. Inputs with total distance larger than T_a are rejected. In the case of voting, the threshold T_v is used to indicate whether the number of votes received by the best matching person is sufficiently large. In the case that the number of votes received is less than T_v, then the input is rejected.

4 Invariance to Image Noise

Image noise due to shift, rotation, and scaling is always present in any practical image capturing system [4]. There are several ways to make template matching-based classifiers less prone to these distortions.

4.1 Elastic Graph Matching

Elastic graph matching has been used extensively with Gabor features to minimize the effect of shift and rotation. In this method, we fix the position of the nodes in the database images. Each node of the input image, though, is allowed to move around within a fixed neighborhood during the distance computation.

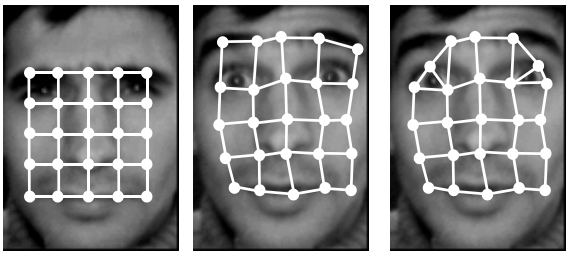

Figure 1. (a) The database (reference) image with rectangular grid shown. (b) The input image and the resulting deformed grid after elastic graph matching. (c) A common type of distortion in the deformation process. Here, areas in the eyebrow are mismatched with areas near the eyes.

Suppose the default position of node i in the input image is denoted (x_i, y_i) and suppose the neighborhood of node i is denoted π_i. To compute the local distance at node i between the input and the database pattern $\mathbf{x}_{mk}(i)$, we extract Gabor features centered at each point in π_i. In each case, we compute the distance between the extracted features $\mathbf{x}(i)$ and $\mathbf{x}_{mk}(i)$. The position in π_i which gives the smallest distance is taken as the final position of node i in the input, and the resulting distance is taken as $d_{mk}(i)$. This process is repeated for each

node in the grid, resulting in a deformed grid over the input image. Figure 1(a) shows the default position of the nodes, and 1(b) shows the final node positions after the elastic graph matching. Notice how the elastic grid is able to compensate for the raised eyebrows in the input.

4.2 Topographical Constraints

Although the deformation of the grid provides flexibility for the matching process, it also introduces the risk of mismatch. For example, one part of the face may be matched to another part. This is shown in Figure 1(c). Here, node points that should have matched with points on the eyebrows were erroneously deformed to positions near the eye.

Wislott [12] systematically investigated how topographical constraints contribute to face recognition accuracy. There are two common types of topographical constraints that can be introduced to reduce the number of mismatches. One way is to allow a maximum number of shifts for each node [1, 12]. The obvious advantage of this method is calculation efficiency, but the drawback is that the topographical relationship among neighboring nodes is not preserved.

The other method, which is commonly used in the face recognition research literature, is to introduce a topography term in the distance calculation [10]:

$$d_{mk} = \sum_{i=1}^{W} d_{mk}(i) + \lambda P(\Delta \mathbf{x}, \Delta \mathbf{y})$$

Here P is a penalty term measuring the total amount of grid distortion as a function of the total distortion in the x-direction: $\Delta \mathbf{x}$ and the y-direction: $\Delta \mathbf{y}$. $P = 0$ if the final node positions coincide with the default positions, and P increases as the final node positions move away from the default positions. Here, the distance between two images depends on the distance of the local features vectors, as well as the value of the penalty topography term, which reflects the distortion of the input image grid relative to the prototype grid. The positive parameter λ can be used to weight the topography term so that a proper balance between the two factors is maintained.

4.3 Group Shift/Local Deformation

We propose a variation of elastic graph matching, called the *group shift*/*local deformation algorithm*. This algorithm reaps the benefits of the above two methods. Here, nodes are clustered into groups, as shown in Figure 2(a). In the first stage of processing, we optimize the position of each group of nodes. In this case, all of the nodes in each group move rigidly, as shown in Figure 2(b). In the second stage of processing, the individual nodes within each group are deformed to optimize their position. The fine-tuning deformation stage is shown in Figure 2(c). In this method, the shifting of the nodes is constrained by other nodes to reduce the opportunity for mismatches. The fine-tuning stage allows relative shift among the nodes in a group and hence gives additional flexibility to the deformation.

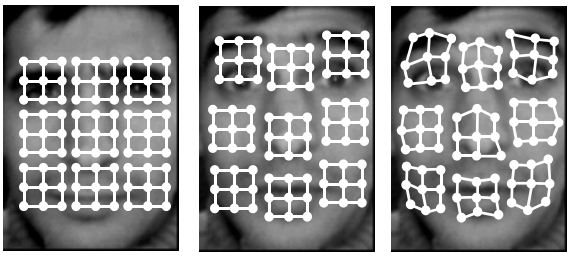

Figure 2. (a) The database (reference) image with rectangular grid partitioned into groups. (b) In the first stage of processing, the groups move rigidly to optimize their position. (c) In the second stage, the nodes within each group deform to fine-tune their position.

5 Methodology

In our experimental tests, we will use two databases. The first database is called the CNNL face database and was collected at Wayne State University. This database contains 1200 different people and 10 samples of each person: 6 showing a blank facial expression, 1 with a smile, 1 with an angry expression, 1 with a look of surprise, and one with an arbitrary facial expression, where the subject tries to fool the system. All images are grayscale and of size 82×115.

A few sample images of the CNNL database are shown in Figure 3. A detailed discussion of the construction of the face database can be found in [9].

We will partition the CNNL database in the following way. The database of stored prototypes **DB-M** will contain 1000 people and 4 image samples for each person: a blank, smile, angry, and surprised image. Three test sets will be used. **TS-b** will contain an additional blank sample and **TS-a** will contain the arbitrary sample for each person in the database. These test sets will be used to measure the rejection rate (RR)

of the system. The test set **TS-FAR** will be used to measure the false acceptance rate (FAR) and contains 200 new people (not in **DB-M**) and all 10 samples per person (hence 2000 images).

The second database is called the FERET database [7, 8] and was collected at NIST. This database contains 1196 different people and 2 samples of each person (**fa** and **fb**). In our experiments, **DB-M** will contain 1000 **fa** images. The test set **TS** will contain the corresponding 1000 images of **fb**. The test set **TS-FAR** will be used to measure the false acceptance rate (FAR) and contains the **fa** and **fb** images of the remaining 196 new people (not in **DB-M**). Note that all the images in the FERET database were preprocessed so that the face was centered in an image of size 82×115.

For all experiments, we will use Gabor features and the city-block distance function when computing local distances.

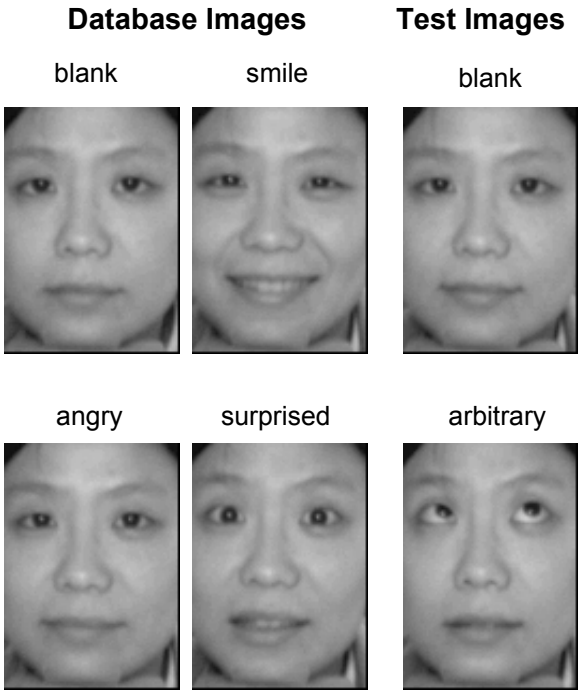

Figure 3. Samples images in the CNNL database.

6 Results

As mentioned at the outset, there is no one optimal operating point for a practical human face recognition system. Rather, systems must be assessed by how well they are able to balance their ability to correctly identify known people with their ability to reject strangers. Typically, both of these values are reported as error measures. The rejection rate (RR) measures the percentage of inputs (in the test set **TS**) that are rejected. The face acceptance rate (FAR) measures the percentage of strangers (in the test set **TS-FAR**) that are erroneously identified with one of the known individuals. In the experiments given below, we will examine how the RR and FAR error rates vary as a function of the threshold used to make the decision.

Figure 4 shows results using the CNNL database with (a) test set **TS-b** and (b) test set **TS-a**. In both cases, the voting-based template matching gives better performance (smaller values of FAR and RR) than the usual local template matching approach resulting from the additive metric.

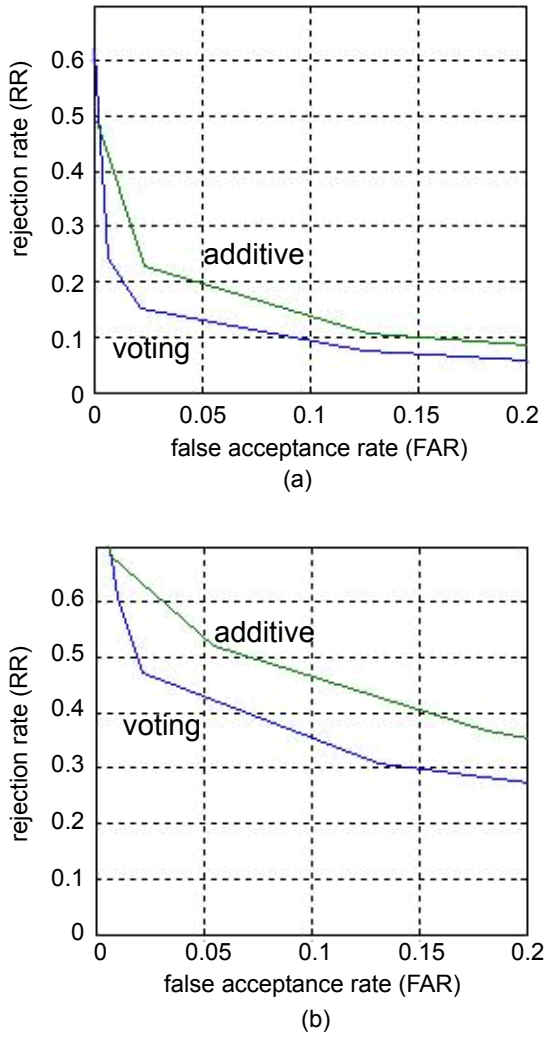

Figure 4. FAR vs. RR for the CNNL database using (a) the test set **TS-b** containing blank test images and (b) the test set **TS-a** containing arbitrary facial expressions. In both cases, standard elastic graph matching is used.

For example, if we insist on a false acceptance rate of no more than 5%, then the additive metric will reject about 20% of the blank facial expression images and 52% of the arbitrary facial expression images. The voting-based metric, though, rejects only 12% of the blank test images and 42% arbitrary images.

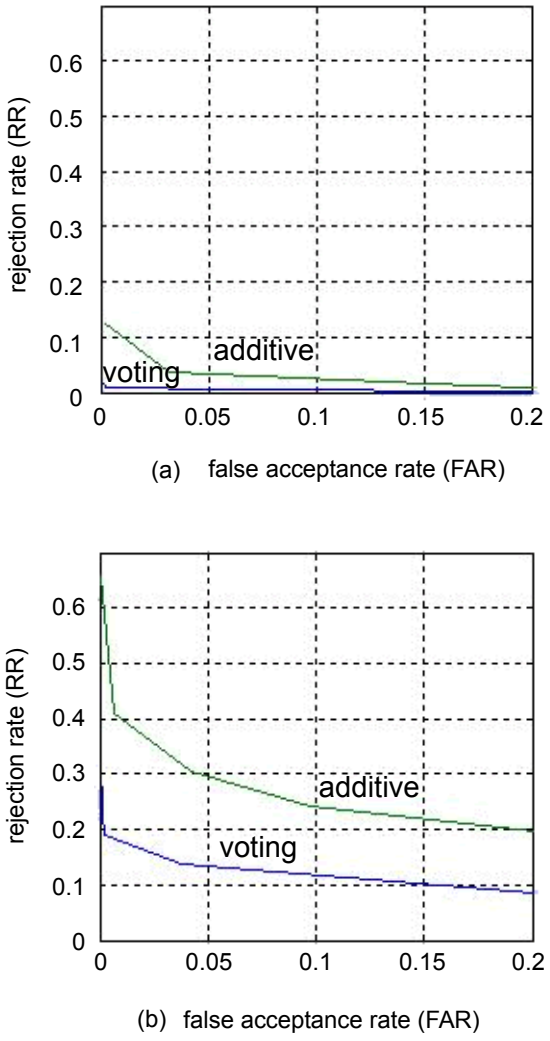

Figure 5. False acceptance rate (FAR) vs. rejection rate (RR) for the CNNL database using (a) the test set **TS-b** containing blank test images and (b) the test set **TS-a** containing arbitrary facial expressions. In both cases, the proposed group shift/local deformation algorithm is used.

Clearly, the test set containing arbitrary facial expression **TS-a** is much more difficult than the test set containing blank facial expressions **TS-b**.

Note that the results in Figure 4 are based on using standard elastic graph matching. Figure 5 shows the results when the proposed group shift/local deformation algorithm is used. Here, as before (a) shows the results on the test set **TS-b** and (b) shows the results on **TS-a**. Comparing Figure 4 to Figure 5, it is clear that the proposed shifting algorithm offers a significant improvement in performance.

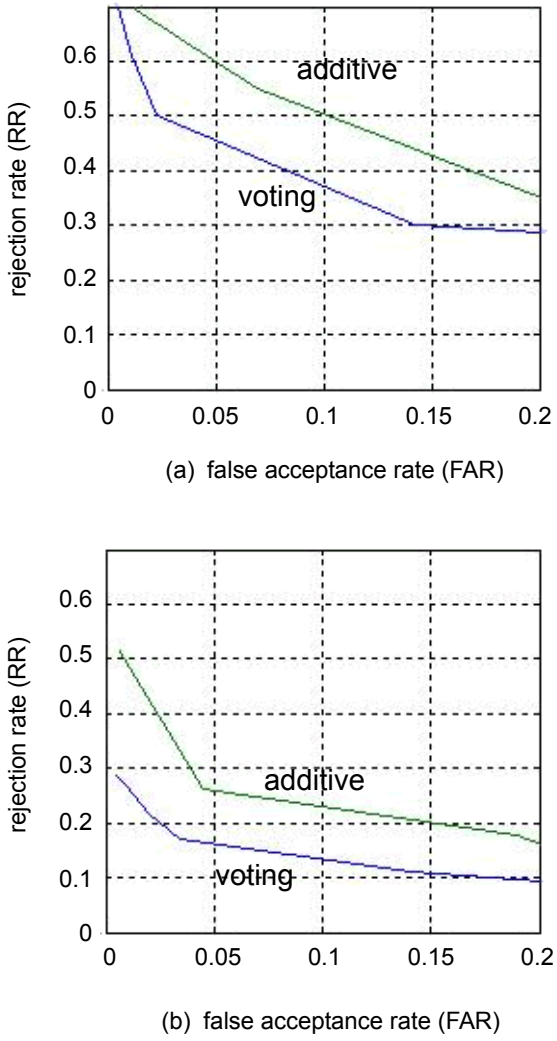

Figure 6. (a) FAR vs. RR on the FERET database using (a) standard elastic graph matching and (b) the proposed group shift/local deformation algorithm.

For example, if we again insist on a false acceptance rate of no more than 5%, the voting-based classifier gives a rejection rate of 12% on the test set **TS-b** using the standard elastic graph matching, but less than 1% rejection using the group shift/local deformation algorithm. The results are even more dramatic on the test set **TS-a**. Here, the rejection rate reduces from 42% to about 13%.

The results on the FERET database are shown in Figure 6. Here, (a) shows the results using standard elastic graph matching and (b) shows the results using the group shift/local deformation algorithm. The same results apply: voting-based matching gives better performance than the additive metric-based matching. The results can be improved even more by using the group shift/local deformation algorithm.

It is interesting to note the similarity of performance seen in the test set **TS-a** of the CNNL database with the FERET database. An open problem in face recognition research is: given a face database, determine the difficulty of the resulting recognition problem. For example, a database containing a large number of identical twins should be more difficult than a database containing people who have vastly different appearance.

7 Summary

In this paper, we examined two different local template matching classification methods. The first is based on summing local distances and the second is based on using a voting strategy. In addition, we examined two different strategies for optimizing the position of the local nodes to compensate for image shift, scale, and rotation. The first is standard elastic graph matching and the second is a variation called the group shift/local deformation algorithm.

Experimental results on two large and independent face databases show that the voting-based classifier outperforms the standard classifier based on summing local distances. In addition, the group shift/local deformation algorithm outperforms standard elastic graph matching. The best results were obtained when using both the voting-based classifier and the group shift/local deformation algorithm.

Acknowledgements

Portions of the research in this paper use the FERET database of facial images collected under the FERET program (Phillips, Wechsler, Huang, and Rauss, 1998).

The authors would like to thank Mehmet Artiklar for his assistance with preprocessing the FERET database images to a standard size and scale.

References

[1] M. Artiklar, M. H. Hassoun, P. Watta, "Application of a post-processing algorithm for improved human face recognition," *Proc. International Joint Conference on Neural Networks*, Vol. 5, pp.3280 -3283, 1999.

[2] D. P. Casasent, "Neural Net Design of Macro Gabor Wavelet Filters for Distortion-Invariant Object Detection in Clutter," *Optical Engineering* 33(7), 2264-2271, 1994.

[3] D. Casaent and J. Smokelin, "Real, Imaginary, and Clutter Gabor Filter Fusion Detection with Reduced False Alarms," *Optical Engineeing* 33(7), 2255-2263, 1994.

[4] C. H. Chen, L. F. Pau, and P. S. P. Wang, *Handbook of Pattern Recognition and Computer Vision*, World Scientific, 1993.

[5] L. Chen and N. Tokuda, "Robustness of regional Matching Scheme Over Global Matching Scheme," *Artificial Intelligence*, In press, 2003.

[6] N. Ikeda, P. Watta, M. Artiklar, and M. Hassoun, "Generalizations of the Hamming Net for High Performance Associate Memory," *Neural Networks*, 14(9), pp.1189-1200, 2001.

[7] P. J. Phillips, H. Moon, S. A. Rizvi and P. J. Rauss, "The FERET Evaluation Methodology for Face-Recognition Algorithms," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 22, No.10, 2000.

[8] P. J. Phillips, H. Wechsler, J. Huang, and P. Rauss, "The FERET database and evaluation procedure for face recognition algorithms," *Image and Vision Computing J*, Vol. 16, No.5, pp 295-306, 1998.

[9] P. Watta, M. Artiklar, A. Masadeh, 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, Pittsburg Pennsylvania, 465-469, May 15-17, 2000.

[10] L. Wiskott, J. M. Fellous, N. Kruger, and C. von der Malsburg, "Face Recognition by Elastic Bunch Graph Matching," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Volume 19, pp. 775-779, 1997.

[11] L. Wiskott, J. M. Fellous, N. Kruger, and C. von der Malsburg, "Face Recognition by Elastic Bunch Graph Matching," *Intelligent Biometric Techniques in Fingerprint and Face Recognition*, Springer-Verlag, 1999.

[12] L. Wiskott, "The Role of Topographical Constraints in the Face Recognition," *Pattern Recognition Letters* 20, 89-96, 1999.