A Weighted Voting and Sequential Combination of Classifiers Scheme for Human Face Recognition

Author 1
Anonymous Address
Anonymous Address
Anonymous Address
anonymous@email

Author 2
Anonymous Address
Anonymous Address
Anonymous Address
anonymous@email

Author 3
Anonymous Address
Anonymous Address
Anonymous Address
anonymous@email

Abstract - *In this paper, we examine the performance of a weighted voting classification strategy for human face recognition. Here, local template matching is used, but instead of summing the local distance measures, a weighted voting scheme based on rank information is used to combine the results of the local classifiers. This strategy can be used with any suitable features; for example, simple pixel features, or Gabor features, etc. If multiple features are available, we show how a sequential combination strategy can be devised to efficiently and reliably compute the final classifier output. Test results are presented for the problem of human face recognition on a large database of faces.*

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

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:

$$\mathbf{DB} = \{(\mathbf{I}_k, ID_k) : k = 1, 2, \ldots, M\}$$

Here, $\mathbf{I}_k$ is the image sample of individual k, and ID_k is the name of the kth 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:

$$\mathbf{DB\text{-}M} = \{(\mathbf{x}_k, ID_k) : k = 1, 2, \ldots, M\}$$

Here, $\mathbf{x}_k$ is the feature vector derived from image $\mathbf{I}_k$. 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.

In this paper, we examine the performance of a weighted voting classification strategy for human face recognition. Here, local template matching is used, but instead of summing the local distance measures, a weighted voting scheme based on rank information is used to combine the results of the local classifiers. This strategy can be used with any suitable features; for

example, simple pixel features, or Gabor features, etc. If multiple features are available, we show how a sequential combination strategy can be devised to efficiently and reliably compute the final classifier output. Test results are presented for the problem of human face recognition on a large database of faces.

2. Voting-based Classification

Many face recognition systems use a local feature extraction approach. In this case, the input image and each database image are partitioned into a collection of windows, and local features, say of dimension *n,* are extracted at each window. Let $\mathbf{x}[i]$ denote the local features extracted from window i of the input image. The N-dimensional feature vector $\mathbf{x}$ consists of a concatenation of all the local features extracted from the N/n windows: $\mathbf{x}[1], \mathbf{x}[2], \ldots, \mathbf{x}[N/n]$. Similarly, for the kth database image, the feature vector $\mathbf{x}_k$ consists of the locally extracted features:

$$\mathbf{x}_k[1], \mathbf{x}_k[2], \ldots, \mathbf{x}_k[N/n], \; k = 1, 2, \ldots, m$$

A common classification strategy involves local template matching. In this case, we require a distance measure to be computed locally at each window. Let d be a distance measure between two n-dimensional vectors. Any suitable distance function can be used. In this paper, we use the city-block distance. Hence, the distance between two local features $\mathbf{x}_k[i]$ and $\mathbf{x}[i]$ is given by:

$$d_{k[i]} = d(\mathbf{x}_k[i], \mathbf{x}[i]) = \sum_{j=1}^{n} |\mathbf{x}_k[i]_j - \mathbf{x}[i]_j|$$

where $\mathbf{x}_k[i]_j$ and $\mathbf{x}[i]_j$ denote the jth component of $\mathbf{x}_k[i]$ and $\mathbf{x}[i]$, respectively.

A simple way to obtain a global distance measure between the N-dimensional feature vector $\mathbf{x}$ and database pattern $\mathbf{x}_k$ is to simply sum up the local distances between $\mathbf{x}$ and $\mathbf{x}_k$:

$$d_k = d(\mathbf{x}, \mathbf{x}_k) = d_{k[1]} + d_{k[2]} + \ldots + d_{k[N/n]}$$

These distances can then be used in a simple nearest neighbor classification strategy. In this case, the smallest distance is found, say d_{k^*}, and the candidate output of the classifier is taken to be person k^*. If the distance d_{k^*} is suitably small, then the candidate is accepted as the system output; otherwise, the input is rejected.

The popular techniques reported in Wiskott et al., (1997 and 1999) use the Gabor transform for local feature extraction, and then use the above summing strategy for combining local distance measures.

An attractive alternative to summing local distances is to use a voting strategy. In this case, at each window i, local distances are computed: $d_{1[i]}, \ldots, d_{m[i]}$ between the input and each of the M database patterns. The smallest distance is found, say $d_{k^*[i]}$, and then the local window i casts a vote for database pattern k^*. Each window votes for the database pattern to which it is locally closest. A decision network then examines the votes cast from all the windows and choose the database pattern with the most votes (a plurality decision rule).

It is easy to introduce a rejection mechanism in the voting classifier. We simply use a threshold T to indicate whether the number of votes received by the best matching pattern is sufficiently large. In the case that the number of votes received is less than T, the input is rejected.

In Anonymous (2003), it was shown that simple pixel-based features with a voting-based classification yields slightly better recognition results (and requires considerably less computation time) than using Gabor features with the summing strategy. In the next section, we generalize the voting-based classifier to allow for each window to cast a set of weighted votes.

3. Weighted Voting

At the local level, the voting classifier works in an all-or-nothing fashion. That is, the database pattern that has the smallest (local) distance gets a vote and all the other database patterns get nothing. It possible, though, that for a noisy input, the correct pattern may not appear first on the list of best matching patterns. In this case, the worst happens: the window casts a vote for a non-correct pattern and the correct pattern gets nothing.

The weighted voting model operates as follows. As before, we compute local distance measures at each window. But instead of just choosing the smallest distance and assigning a vote to the corresponding database pattern, we sort all the distances and assign a rank to each. Let the database pattern that has the smallest (local) distance be assigned $rank = 1$. The pattern with the next smallest distance will have $rank = 2$, etc. The distance computations and ranking of database patterns are done independently by each local window, hence this type of classifier is amenable to implementation on parallel processing systems.

A simple example will help clarify the concepts. Suppose we have a database consisting of $m = 3$ patterns and suppose the patterns are partitioned into $N/n = 9$ windows (in a 3×3 arrangement). Suppose that for a given input, the local distances are computed and sorted, and the resulting rankings are as shown in

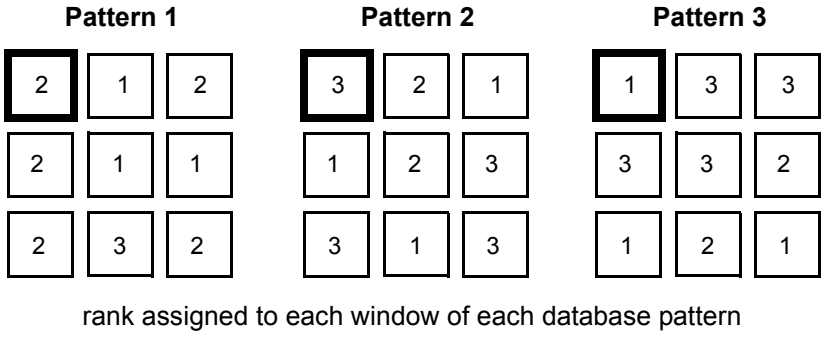


Figure 1. An example of a database with 3 patterns and 3×3 local classifiers. The rankings assigned to each window of each database pattern are shown.

Figure 1. For example, for the first window (highlighted in the Figure), the local distances d_1, d_2, and d_3 were found to satisfy $d_3 < d_1 < d_2$. Hence, for this window, database pattern #1 is assigned $rank = 2$, database pattern #2 is assigned $rank = 3$, and database pattern #3 is assigned $rank = 1$. The ranking from all the other windows are shown, as well.

Simple voting can be seen as a special case of weighted voting, where we only consider the $rank = 1$ information. In the case of Figure 1, though, each database pattern would receive 3 votes, and hence the simple voting strategy cannot adequately discriminate among the database patterns. Clearly in this example, the second and third place rankings give additional information that would lead us to prefer database pattern #1 over the other two.

We propose that the total number of votes $N_{\mathbf{x}_k}$ assigned to pattern $\mathbf{x}_k$ be a weighted sum of the number of windows at each ranking. For database pattern $\mathbf{x}_k$, let $N_{\mathbf{x}_k}^{(1)}$ be the number of windows that were found to have rank 1, $N_{\mathbf{x}_k}^{(2)}$ be the number of windows that have rank 2, etc. In general, let $N_{\mathbf{x}_k}^{(r)}$ denote the number of windows that have rank r, $r = 1, 2, \ldots, m$. For example, for the first database pattern $\mathbf{x}_1$ in Figure 1, $N_{\mathbf{x}_1}^{(1)} = 3$, $N_{\mathbf{x}_1}^{(2)} = 5$, and $N_{\mathbf{x}_1}^{(3)} = 1$.

The total vote received by database pattern $\mathbf{x}_k$ is given by:

$$N_{\mathbf{x}_k} = \alpha_1 N_{\mathbf{x}_k}^{(1)} + \alpha_2 N_{\mathbf{x}_k}^{(2)} + \ldots + \alpha_m N_{\mathbf{x}_k}^{(m)} \qquad (1)$$

where the weights $\alpha_1, \alpha_2, \ldots, \alpha_m$ are used to adjust the relative importance of each ranking. Note that the simple voting classifier described in the previous section has: $\alpha_2 = \alpha_3 = \ldots = \alpha_m = 0$.

Now how should the weights be chosen? First, let's define a quantity that locally relates rank r to the probability of choosing the correct database pattern (denoted ω). Let $P^{(r)}$ be

$$P^{(r)} = \text{Prob}(\omega | rank = r) \qquad (2)$$

That is, given the fact that a database pattern is locally ranked r, $P^{(r)}$ gives the probability that it is, in fact, the correct database pattern. Similarly, let $E^{(r)}$ be the probability that, given the rank is r, an incorrect database pattern is chosen. We will assume that, in the case of incorrect classification, each of the $M-1$ non-correct database patterns are equally likely to be chosen; hence

$$E^{(r)} = \frac{1 - P^{(r)}}{M - 1} \qquad (3)$$

Assuming that the local classifiers make decisions independently, Lin et al. (2003) showed that the optimal set of weights is given by

$$\alpha_r = \log\left(\frac{P^{(r)}}{E^{(r)}}\right), \; r = 1, 2, \ldots, m \quad (4)$$

To summarize, the weighted voting classifier works as follows. Given an input pattern $\mathbf{x}$, local distance computations and ranking are made between the input and each database pattern $\mathbf{x}_k$ (see Figure 1). For each database pattern, we determine how many of the local windows were ranked first: $N_{\mathbf{x}_k}^{(1)}$, ranked second: $N_{\mathbf{x}_k}^{(2)}$, etc. Finally, for each database pattern, Equation 1 (with weights given by Equation 4) is used to compute the final weighted vote. The pattern that receives the largest weighted vote is the candidate output. If the weighted vote of the candidate is greater than the system threshold T, then the system outputs the candidate pattern; otherwise the input is rejected.

If multiple images for each person in the database are available, it is possible to compute $P^{(r)}$ and $E^{(r)}$ by using a delete-one approach. When only 1 image per person is available, such a training phase is not possible. In this case, we use the following heuristic to set the weights:

$$\alpha_r \equiv \frac{1}{r} \qquad (5)$$

We have found empirically that this choice of weights works fairly well (at least for the face recognition problem).

4. Combination of Classifiers

The above weighted voting classification strategy can be used with any type of features. If multiple features are available, then separate weighted voting classifiers can be formulated, and a combination of classifiers strategy can be used to combine the various outputs to get a single decision. To simplify the discussion, we consider the case of combining the outputs of two voting-based classifiers: A and B, as shown in Figure 2.

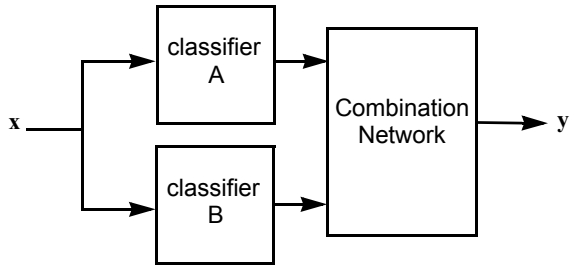


Figure 2. Combination of classifiers.

Various combination strategies have been studied, such as averaging and max/min. In the averaging combination strategy, a system threshold T is chosen. The input is classified by both classifiers A and B. If A and B output different classes, then the input is rejected. However, if A and B output the same class, then the weighted votes received by both classifiers are averaged and compared to T. If the average number of votes exceeds T, the common output is accepted as the system output; otherwise, the input is rejected.

In the max/min combination scheme, the system output is determined by whichever classifier receives the most votes. In both the average and min/max schemes, the outputs of both classifiers must be computed, and hence the computation time is the sum of the times required by A and B (assuming that the system is run on an ordinary single processor machine).

An appealing feature of voting-based classification is that we can use the total number of votes as a confidence measure for the system output. The more votes the output receives, the more likely the output is the correct match.

Consider the sequential strategy shown in Figure 3. Here, the input is always first classified by A. If the number of votes received by the candidate output is sufficiently large, we accept the decision of A. Otherwise, additional processing is required, and the input is sent to classifier B. When both classifiers are need, a traditional combination scheme can be used.

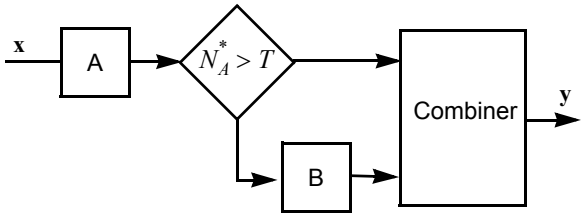


Figure 3. Sequential combination of classifiers.

5. Testing Methodology

The face recognition research community has developed protocols for assessing system performance (Phillips et al., 1998, 2000). There are 2 main tests commonly used: *identification test* and *watch list test*. In both cases, we are given a database of face images. In the identification test, the system is tested with images of known people (of course, the test images are different from the database images). For a given input image, the task is to identify which database person it is. In this case, no rejection state is needed. The measure of merit here is the *identification rate* (IR), which is the probability that a given input image will be matched with the correct database person.

In the watch list test, the system must have a rejection mechanism. Here, two test sets are required: TS_G, which contains images of the known people, and TS_N, which contains images of strangers (people not in the database). For the TS_G test set, there are two measures of merit: the *detection and identification rate* (DIR), which is the percentage of images that are correctly matched with the known individuals, and the *false rejection rate* (FRR), which is the percentage of images that are rejected by the system (Phillips, 1998). For the TS_N test set, there is only one measure of interest: the *false acceptance rate* (FAR), which gives the percentage of imposter images that are incorrectly matched with someone in the database.

Of course there is a trade-off between the detection

and identification rate and the false acceptance rate. Typically, face recognition systems are designed with a tunable parameter or threshold T which allows one to control the trade-off between DIR and FAR. A *receiver operating characteristic* (ROC) curve can be constructed which shows how DIR and FAR vary as a function of T. Note that the identification test can be seen as a special case of the watchlist test on TS_G, where the threshold is set to 0 (so the system does not reject any images).

The face database used for the experiments reported here contains 1400 different people and 4 samples of each person. The 4 samples show different facial expressions: blank expression, smile, angry expression, and surprise. There are two test images per person. The first test image shows a blank facial expression (different from the blank database image). The second test image shows an arbitrary facial expression, where the subject tries to fool the system. A few sample images are shown in Figure 4. A detailed discussion of the construction of the face database can be found in (Anonymous, 2000). All images in the database are of size 82×115, and show the subject centered in the image.

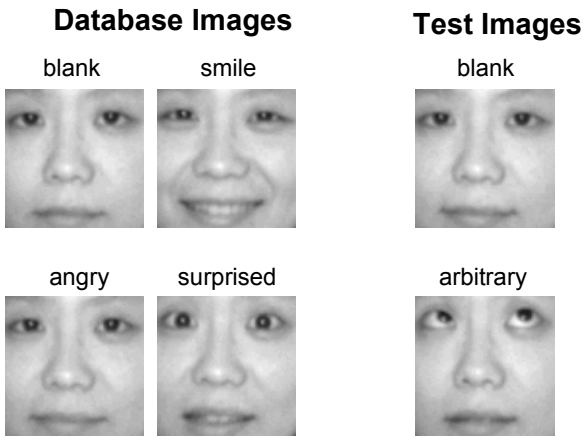


Figure 4. Samples of database and test images.

In the experiments presented in the next section, we will use a database of 1300 people. The remaining 100 people will be used as a false positive test set (needed to compute the false acceptance rate).

6. Experimental Results

6.1 Weights

Since the given database contains multiple images per person, it is possible to compute the probabilities $P^{(r)}$ and $E^{(r)}$ using a delete-one training method. Figure 5 shows the results of such an experiment. Here, we computed separate values of $P^{(r)}$ for each of the different facial expressions. When the rank is less than 5, there can be a significant difference in the values of $P^{(r)}$ among the expressions. As the rank increases, though, all 4 sets of weights converge to similar values. Depending on the application, one might choose to use just one of the sets of probabilities, or possibly compute the average of all 4 values. As mentioned above, a useful heuristic measure for the weights is $\alpha^{(r)} = 1/r$. These values are shown as circles on the graph.

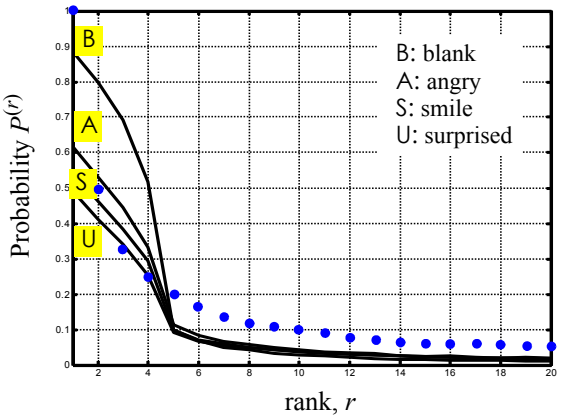


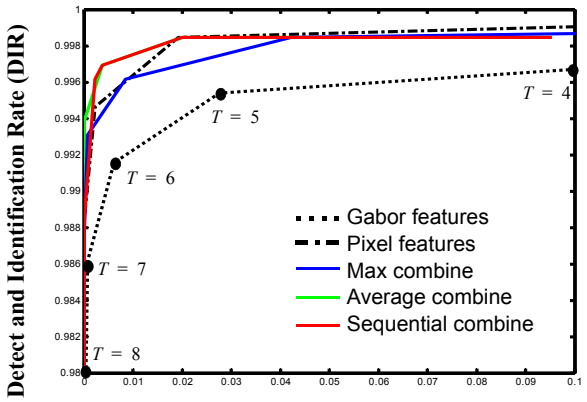
Figure 5. Probability $P^{(r)}$ as a function of rank r.

6.2 Sequential Combination of Classifiers

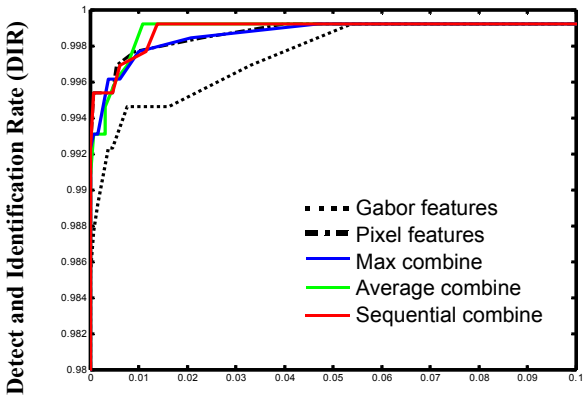
In this Section, we examine the difference in performance between simple voting and weighted voting. In addition, we examine the difference in performance between using a single classifier and using a combination of classifiers approach. The first individual classifier uses pixel features, while the second use Gabor features. When combining classifiers, A is taken to be the pixel-based classifier and B is taken to be the Gabor-based classifier.

Figure 6 shows the results of the watchlist test when the blank test images are used with (a) simple voting and (b) weighted voting. In each case, the individual Gabor-based and pixel-based classifiers are shown, as well as 3 different combination schemes: average, max/min, and the proposed sequential combination strategy. The system threshold T is typically implicit in the ROC graph. For these curves, the threshold was varied from $T = 4$ to $T = 8$, and is explicitly labeled in Figure 6(a) for the Gabor features classifier.

The results show that weighted voting performs slightly better than simple voting. In both cases, the sequential combination scheme gives similar performance to the average combination scheme.



(a) False Acceptance Rate (FAR)

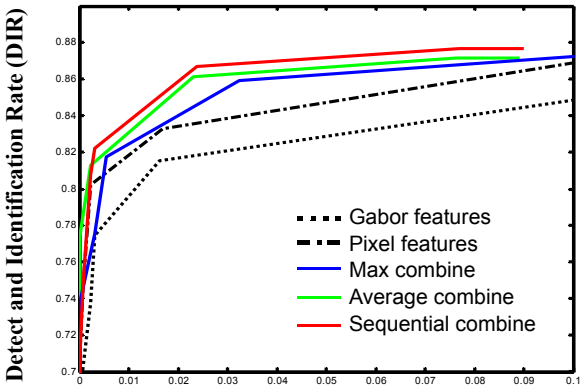


(b) False Acceptance Rate (FAR)

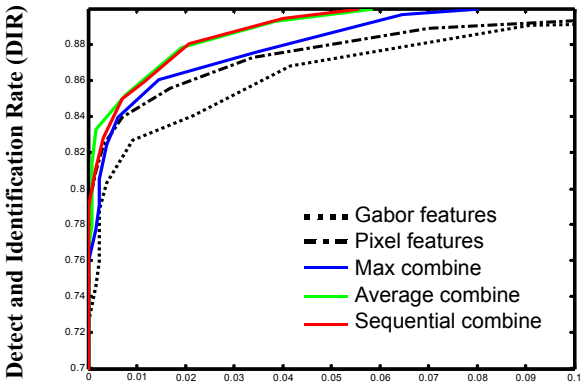
Figure 6. ROC curves on the 1300 person database with the blank test images. (a) Simple Voting and (b) Weighted voting. The results of the individual Gabor and pixel-based classifiers are shown, along with 3 combination schemes: average, max/min, and the sequential classifier.

Figure 7 shows the results for (a) simple voting and (b) weighted voting when the arbitrary test images are used. Clearly, the arbitrary test set is more difficult than the blank test set, as evidenced by smaller detection and identification rates. The performance improvement in using weighted voting is more apparent.

Also more apparent in Figure 7 is that by combining classifiers, it is possible to obtain results better than either of the individual classifiers can achieve. Of course, the price to pay for the increased classification performance is in terms of longer computation time.



(a) False Acceptance Rate (FAR)



(b) False Acceptance Rate (FAR)

Figure 7. ROC curves on the 1300 person database with the arbitrary expression test images. (a) Simple Voting and (b) Weighted voting.

Table 1 shows the time required to classify a single image for each of the individual classifiers, as well as the 3 different combination schemes. As expected, the time required to compute the output for the average or max/min classifier is the sum of the times required by the individual classifiers. The sequential classifier, though, gives similar performance to both combination schemes, but requires much less computation time.

It is interesting that the sequential combination scheme requires about twice as much time for the arbitrary expression test images than the blank images. This indicates that the second classifier is called upon much more often for the more difficult images.

Test Set	**Individual Classifiers**		**Combination**		
	A	B	Ave	Max	Seq
Blank	3.1	15.1	18.2	18.2	3.1
Arbitrary	3.1	15.1	18.2	18.2	6.1

Table 1. Average classification time (in seconds) per test image. 2 different test sets are used: blank facial expression and arbitrary facial expression. Here, classifier A uses pixel features and weighted voting classification, and classifier B uses Gabor-based features with weighted voting classification.

7. Summary

Experimental results on a large database of face images (1300 people) show that the proposed weighted voting-based classifier outperforms simple voting. In addition, we showed that a sequential combination scheme provides an effective way to combine the results of two different weighted voting classifiers. The sequential combination strategy achieves classification results as high as the traditional combination schemes of averaging and max/min, but requires far less computation.

References

1. Anonymous, "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. Anonymous, "Generalizations of the Hamming Net for High Performance Associate Memory," *Neural Networks*, 14(9), pp.1189-1200, 2001.

3. 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.

4. 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.

5. Anonymous, (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, Pittsburgh Pennsylvania, 465-469, May 15-17, 2000.

6. 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.

7. 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.

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

9. X. Lin, S. Yacoub, J. Burns and S. Simske, "Performance Analysis of Pattern Classifier Combination by Plurality Voting", *Pattern Recognition Letters*, 24(12), pp. 1959-1969, 2003.