Local Voting Networks for Human Face Recognition

Metin Artiklar, Xiaoyan Mu, 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
Detroit, MI 48128

watta@umich.edu

Abstract

We investigate a template matching-based classifier system which uses local distance computations and a voting scheme. The proposed system has a mechanism to reject unknown patterns, and provides invariance to small amounts of translation. Experimental results are presented for 3 different types of face recognition problems: *classification experiments* where we measure the ability of the system to identify known individuals; *false positive experiments* where we measure the ability of the system to reject images of unknown individuals; and *temporal experiments*, where we measure the ability of the system to recognize images of known individuals taken over a period of time (6 months). The results show that the proposed system performs well on all 3 of these problems.

1. Introduction

The automated face recognition problem is formulated as follows [1]. We are given a set of image samples of known individuals: **DB**. The task is to design a fully automated recognition system such that for any input image, the system does one of the following:

1. Identify the input with one of the known individuals, or

2. Reject the input as not known to the system.

Rejections can arise either because the input image is not a face, or else it is a face of an individual who is not in the database. Of course, the trick is to reliably recognize known individuals (that is, achieve a high correct classification rate) and reject all unknown individuals (that is, achieve a low false positive rate).

A block diagram of a typical template matching-based face recognition system is shown in Figure 1. The *preprocessor* is responsible for extracting the face portion of the image (from, conceivably, a much larger image containing background and clutter, etc.) and normalizing the image to a standard position, scale, etc. After segmenting and normalizing the face, typically some *feature extraction* is applied to the pixel data in order to reduce the dimensionality and/or provide some invariance to common distortions, such as translation, scaling, and rotation. Common types of feature extraction methods include eigenfaces [2], Fisherfaces, [3], and wavelets [4]. Once chosen, the methods of preprocessing and feature extraction are applied to all images in **DB** to create a *memory set* or *database* **DB-M**. Hence for each image in **DB**, **DB-M** contains the corresponding feature vector. Of course, the preprocessing and feature extraction needed to construct **DB-M** can be done in advance, and hence is an off-line computation. We will assume that **DB-M** contains multiple samples of each known individual. If there are M known individuals and K samples per individual, then

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

Once the feature extraction method is chosen and the database **DB-M** is constructed, a classifier system is needed to map the input image to the proper output. In this paper, we focus on template matching-based classifiers. In this case, the classifier has two main computational steps: a *matching* stage and a *decision* stage. Of course before matching can be done, we must apply the preprocessing and feature extraction to the input image (typically, the same processing that was applied to the memory set). The matching stage involves comparing the input feature vector to each of the stored feature vectors in **DB-M**. Finally, the decision stage examines the results of the matching stage and either identifies the input with a known individual, or else rejects the input.

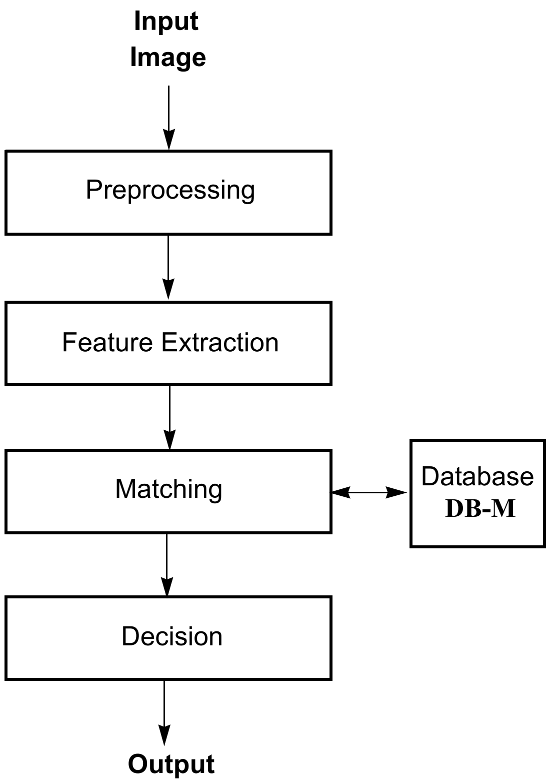


Figure 1. Main computational stages of a face recognition system.

Note that since we require the system to reject unknown individuals, the decision network must necessarily be more sophisticated than a simple min select, as used by the classical nearest neighbor algorithm (which just identifies the input with the best matching database pattern). In order to determine whether a pattern should be identified or rejected, the decision algorithm typically uses some type of *threshold*. If the best matching distance is lower than the chosen threshold, then the input is classified; otherwise it is rejected. In Section 2, we will devise a training method for computing suitable thresholds.

Even though many sophisticated neural net-based classifiers and associative memory models have been developed in the last few decades, there are several reasons why this simple type of template matching approach is so often used in practical systems. First, template matching systems typically require far less training than neural net-based approaches. Second, template matching systems are much easier to maintain. Individuals can be added to or deleted from the system by simply adding or deleting the corresponding image samples from the database. Some neural network approaches require complete retraining, even when adding or deleting a single individual.

On the other hand, there are several disadvantages associated with template matching. First, in the case of large databases, since the input has to be compared to each prototype, the system may operate too slowly. Second, template matching is sensitive to common types of distortions, such as shift, rotation, and scaling.

To address these disadvantages, a template matching-based system was devised which uses local distance measures and a voting mechanism [5, 6]. This algorithm can be called *local matching and voting network* (LMVN) and combines the parallel and distributed processing (PDP) paradigm of neural networks with the flexibility and practical advantages of template matching.

2. Review of the Local Matching and Voting Network

To address the speed problem associated with template matching systems, local distance computations are introduced which can be performed independently and hence in parallel [5]. This type of parallel and distributed processing approach is well suited for VLSI hardware implementation and fine-grained processor arrays. In addition, the proposed system offers tolerance to small amounts of image shift by allowing each window to be optimally positioned in its surrounding neighborhood.

2.1 Local Distance Calculations

To localize the distance computation, we partition both the input image $\mathbf{x}$ and the database images $\mathbf{x}_{mk}$ using non-overlapping windows, as shown in Figure 2. Matching is then determined locally between corresponding windows. For example, consider the highlighted window of the input image in Figure 2. We compute the city-block distance between this window and the corresponding windows of the database images (also highlighted). The local window determines the database image which gives the smallest distance, and casts a vote for that image. We repeat this process for all the local windows and keep track of all the votes v_{mk} received by each image in the database.

Once the local voting is done, we determine how many votes were cast for each individual by summing among all the image samples for that individual:

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

Note that, depending on the type of image samples used, we may not want to include all of the samples in the sum in (1). Rather, we can select just a portion of the samples to use. For example, for the simulations presented in Section 4, we store 4 image samples per person in the database. But when computing the vote sum, we only include the best 3 matching prototype images and not all 4. The number of prototypes to include can easily be determined by running trial simulations [6].

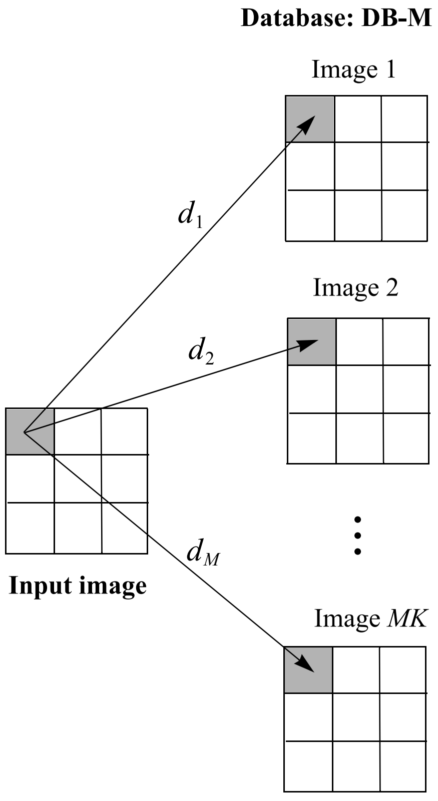


Figure 2. Schematic diagram of the voting process. Each window computes local distances and then casts a vote for the best matching image in the database.

Note that this voting network was studied both theoretically and experimentally for memory sets consisting of random binary images [5, 6]. It was found that local window size had a profound effect on system performance. In particular, it was found that the system performed best for very small and very large window sizes. In this paper, we consider the case of grayscale face images.

2.2 Optimizing Window Position for Shift Invariance

As mentioned above, template matching is sensitive to image shift. In order to provide robustness to small amounts of shift, we propose the following technique to optimize the position of the local windows during the matching process. The shift process is applied when computing the (local) distance between an input window and a database window.

First, the distance between the input window and the corresponding database window is computed and recorded. Then, we shift the input window by 1 pixel in 4 directions: north, south, east, and west and measure the resulting distance with the prototype. If the smallest of these 4 distances is smaller than the recorded distance, then we move the input window in that direction and repeat the process. The process terminates when none of the 4 distances yield a smaller value than the best recorded distance. In addition, the process terminates when a maximum number of shifts have occurred. In the experiments reported here, a maximum of 5 steps are used. A more detailed explanation of the shifting method can be found in [7].

2.3 Threshold Calculation

As mentioned above, in order for the system to determine whether to classify or reject an input, suitable thresholds are needed. In the case of the voting network, we determine closeness of match between the input and the database individuals according to number of votes received. Hence the threshold will be used to determine if the best matching individual received a sufficiently large number of votes in order to make an identification.

Unlike a typical nearest neighbor-based system where a separate threshold is needed for each individual in the database, the voting network requires only a single threshold. The threshold T is used in the following way. From the ordered list of best matching prototypes, we select the individual who received the most votes; call this individual i^*. We compare v_{i^*} to the threshold T. If v_{i^*} exceeds T, then we identify the input as individual i^*; otherwise, we reject it [6].

Now how does one determine the value of the threshold T? We propose that T be computed so as to ensure good performance on the classification task, as well as the rejection task. Hence, for the training phase, we construct a training set **TS** which contains two parts: **TS-C** (for classification training set) and **TS-FP** (for false positive training set). **TS-C** contains additional image samples of each individual in the database and is used to determine how many votes known individuals are likely to receive (addresses the correct classification ability). **TS-FP** contains image samples of unknown individuals and is used to determine how many votes strangers are likely to receive (addresses the performance on false positive experiments).

In the training phase, we store all of the desired prototypes in **DB-M**. Then for each image in **TS-C**, we perform the voting template matching process as outlined above (complete with the window alignment procedure). Since this is a training phase, we know the true identity of each image in **TS-C**. Hence we record the number of votes received by the true person. Similarly, for each image in **TS-FP**, we record the number of votes received by the samples of the best-matching database individual.

Figure 3 shows the average number of votes received by the best matching image samples in **DB-M** when the input is a known person (circle on top) and a stranger (circle on bottom). The standard deviation is also shown. For example, consider the results for the 1st best match case. This result says that when the input is a sample of, say, known individual *i**, the best matching image sample of *i** in **DB-M** receives (on average) 19.1 votes with standard deviation 7.0. However, when the input is a stranger (in **TS-FP**), the average number of votes received by the best matching image is 3.2 with standard deviation 1.2.

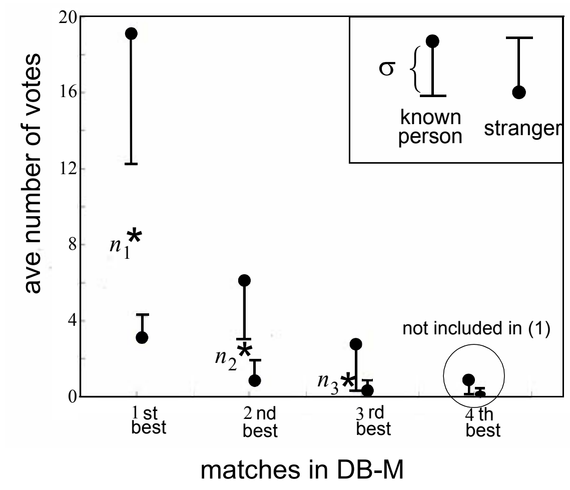


Figure 3. Average number of votes received by the best matching images in **DB-M** when the input is a known person (circle on top) and unknown person (circle on bottom). The standard deviation is indicated by the length of the attached line.

In order to find a threshold, we look for separation between the average vote values of known people and strangers. For the first 2 samples, there is clear separation in mean, and separation even if the standard deviation is taken into account. For the 3rd sample, the separation starts to disappear. However, there is such a large separation in mean (2.7 for known people and 0.3 for strangers) that we also include the 3rd sample in the sum. We don't include the 4th sample in the sum (see Eq. 1) because there is very little separation in mean.

Finally, the threshold is determined from Figure 3 by summing the average number of votes in the middle of each separation area (indicated on the graph by n_1, n_2, and n_3): $T = n_1 + n_2 + n_3$. Note that, in general, the threshold is a function of the given database $T = T(\mathbf{DB\text{-}M})$. However, at least for the relatively well-behaved CNNL database (described later) we found that *T* is only a function of the number of people included: $T = T(M)$. For example, if we start with a gallery of 1000 individuals and randomly choose 500 individuals to comprise **DB-M**, then a threshold of $T = 10$ will result, no matter which 500 individuals are chosen. Also, *T* changes relatively slowly as *M* increases. For example, for $M = 200$, $T = 11$, while for $M = 1000$, $T = 9$.

3. Weighted Voting

In the voting method outlined above, if we neglect the unlikely case of a tie in the distance calculation, then a local window casts only a single vote for the best matching image. All other images don't receive any votes. The idea here is that the correct person may not be first on the list, but may be second or third. We propose a weighted or fuzzy-type of voting whereby the vote is not cast as a 0-1 binary decision; rather, the vote is cast as a real number in the interval [0, 1].

To determine the proper weight of the vote, we construct a graph like the one in Figure 4.

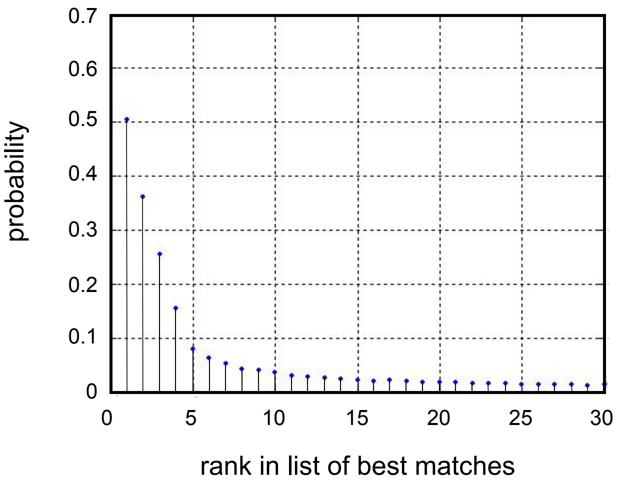


Figure 4. Probability that the correct person appears 1st, 2nd, 3rd, etc. on the list of best matching images.

This Figure shows (for each local window) the probability that the correct person is in position *i* on the (sorted) list of best-matching images. Note that the experiment is run one window at a time and then the results over all the windows are averaged. Consider the result for the best matching image (rank 1). Here we see a surprising result: Each window votes for the correct person only 50% of the time! But even though the local decisions are so unreliable, the overall system can achieve a very high correct classification rate. How? Well, it's due to the voting mechanism itself. The local windows will choose the correct person 50% of the time. However, the other 50% of the time, the votes are split among several different alternate candidates. When the votes are tallied, the correct person ends up with the most votes.

The weighted voting scheme is based on giving each candidate image a vote weighted by the probabilities given in Figure 4. So instead of the best matching image getting a vote of 1, it now gets a vote of 0.5. The second best matching image gets a vote of 0.36, and so on. We will compare the performance of this weighted voting scheme to the basic voting model.

4. Testing Methodology

4.1 Performance Evaluation

In this paper, we use a rigorous testing methodology to study the performance of the voting network. We will assess system performance by performing 3 different types of experiments:

1. **Correct classification experiments**. Measure the ability of the system to correctly identify known individuals. There are 3 important measures to consider here: percent correct, percent rejected, and percent misclassified. Of course since these quantities sum to 100%, only two need to be reported. We will report percent rejected and percent misclassified.

2. **False positive experiments**. Measure the ability of the system to reject individuals who are not part of the database. Here the measure of merit is the number of false positive matches (i.e., incorrectly identified as a known individual).

3. **Temporal experiments**. Measure the ability of the system to correctly classify individuals over time. In many of the publicly available face databases, the database images and the test images of each person are collected on the same day. However, in a practical setting, the database samples will nearly always be collected on a different day from the input image. Unfortunately, there are surprisingly few studies in which these type of experiments are considered.

From previous work, we have found that it is not terribly difficult to design systems that perform well on one of the above problems. But designing a system to perform well on all three problems simultaneously is quite a challenge. Unfortunately, many face recognition algorithm designs reported in the literature just report performance on the correct classification experiments. However, it is clear that a practical system must have good performance on all 3 types of problems [1].

4.2 The Database of Face Images

The CNNL face database used for the experiments reported here was collected at Wayne State University and contains 1400 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. A few sample images are shown in Figure 5. A detailed discussion of the construction of the face database can be found in [8]. For a few selected individuals, additional images were captured periodically over 6 months. These images will form the temporal test set.

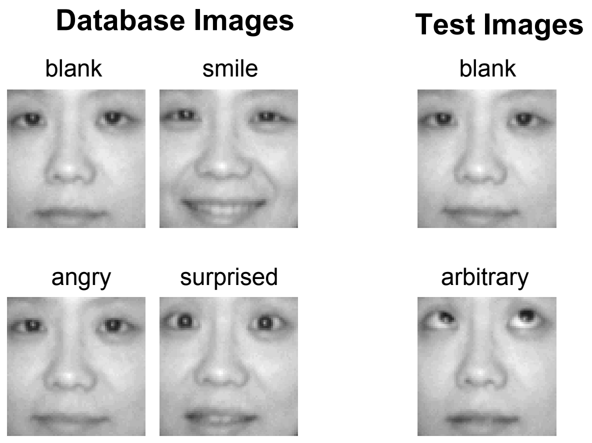


Figure 5. Samples of the database and test images.

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. The training set **TS-C** will contain 4 blank images per person; **TS-FP** will contain 200 people (not in **DB-M**) and all 10 samples for each person (hence 2000 images total). The correct classification test set will contain 2 samples per person: a blank expression and the arbitrary expression. The false positive test set will contain 200 people (not in **DB-M** and not in **TS-FP**) and all 10 samples per person (hence 2000 images).

5. Results

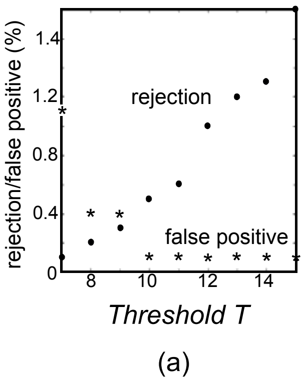
The results of the correct classification experiments are shown in Table 1. We see that the results on the blank test images are very reliable, with 100% correct classification at less than 3% rejection. As expected, since the arbitrary test image is a much harder test set, the number of rejections are much higher. Here we have 100% correct classification with less than 23% rejection.

The results of the false positive experiments are also shown in Table 1. Both the basic voting network and the weighted voting network achieve less than 1% false positive matches.

Finally, for the temporal database, images of 3 individuals in **DB-M** were captured periodically over a period of 6 months (about 30 different sessions for each person and 10 samples taken each time). The classification results on this database are: 100% correct classification (0% misclassification) with 9% rejection for the basic voting method and 5.7% rejection for the weighted voting method. The high rejection rate here is indicative of the difficulty of the temporal classification problem [9].

Test Set	**Basic Voting (%)**		**Weighted Voting (%)**	
	Reject	Misclass.	Reject	Misclass.
Classification blank expression	1.1	0	0.8	0
Classification arbitrary Expression	26.7	0	22.0	0
False Positive	-	0.5	-	0.7
Temporal	9.0	0	5.7	0

Table 1. False positive performance for the voting network. The results are reported as %Rejection and %Misclassification.



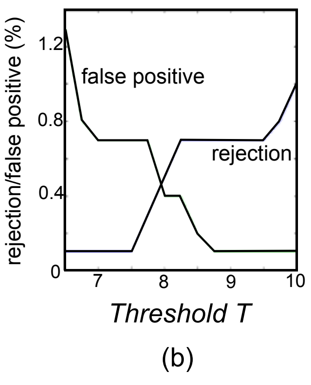


Figure 6. The false positive and reject rates for (a) the basic voting method and (b) weighted voting.

As with most practical systems, there is no one optimal solution or operating point. In the case of the proposed voting network, system performance is determined by the value of the threshold. Figure 6 shows how the performance of the system varies as a function of the threshold *T*. Notice that the threshold plot is discrete for the standard voting and continuous for the weighted voting. Although both voting methods perform nearly equally well, the weighted voting offers more flexibility in that there are more operating points to choose from (continuous choice of *T* rather than discrete).

References

1. Chellappa, R., Wilson, C., and Sirohey, S. (1995). "Human and Machine Recognition of Faces: A Survey," *Proceedings of the IEEE*, **83**(5), 705-740.

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

3. 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.

4. Duc, B. and Fischer, S. (1999). "Face Authentication with Gabor Information on Deformable Graphs," *IEEE Transactions on Image Processing*, **8**(4), 504-516.

5. Ikeda, N., Watta, P., Artiklar, M, and Hassoun, M., (2001). "A Two-level Hamming Network for High Performance Associative Memory," *Neural Networks*, **14**, 1189-1200.

6. Artiklar, M. (2002). *Capacity Analysis of Voting Networks with Application to Human Face Recognition*, Ph D. Thesis, Dept. Electrical and Computer Engineering, Wayne State Univbersaity, Detroit, MI.

7. 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.

8. Watta, P., Artiklar, M., Masadeh, A., and Hassoun, M. H. (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, .

9. Phillips, P., Moon, H., Rizvi, S., and Rauss, P. (2000). "The FERET Evaluation Methodology for Face-Recognition Algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(10), 1090-1104.