Principal Component Analysis and Neural Network Based Face Recognition

Qing Jiang

Mailbox

Abstract

People in computer vision and pattern recognition have been working on automatic recognition of human faces for the last 20 years. Our approach to the face recognition problem is to combine the **principal component analysis** and **neural network**. Here the faces are vertically oriented frontal view with wide expression change. They are extracted from the images by the face detection code [2] first. It will eliminates background influence as much as possible. The face space is described by a set of eigenfaces. Each face is efficiently represented by its projection onto the space expanded by the eigenfaces and has a new descriptor. Neural networks are used to recognize the face through learning correct classification of these new descriptors. A real-time system has been created which combines the face detection and recognition techniques. It continuously takes image from camera, finds the face inside the image and recognizes it. A recognition rate of more than 95% has been achieved over real tests. It is also shown that our face database can be easily expanded to accommodate more individuals.

1. Introduction

The problem can be described as following. Given an image of human face, compare it with models in the database and report who it is if a match exists. Here the image is gray scale, vertically oriented frontal view. Normal expression variation is allowed and the image is prepared under roughly constant illumination.

Because usually images are bigger than the actual faces, the first problem is to find the face in the image, or face detection which is another closely related problem. We use the face detection code by Kah-Kay Sung from MIT Artificial Intelligence Lab. We make some change to the code to make it run faster and locate face more accurately.

Principal component analysis is applied to find the aspects of face which are important for identification. Eigenvectors (eigenfaces) are calculated from the initial face image set. New faces are projected onto the space expanded by eigenfaces and represented by weighted sum of the eigenfaces. These weights are used to identify the faces.

Neural network is used to create the face database and recognize the face. We build a separate network for each person. The input face is projected onto the eigenface space first and get a new descriptor. The new descriptor is used as network input and applied to each person's network. The one with maximum output is selected and reported as the host if it passes predefined recognition threshold.

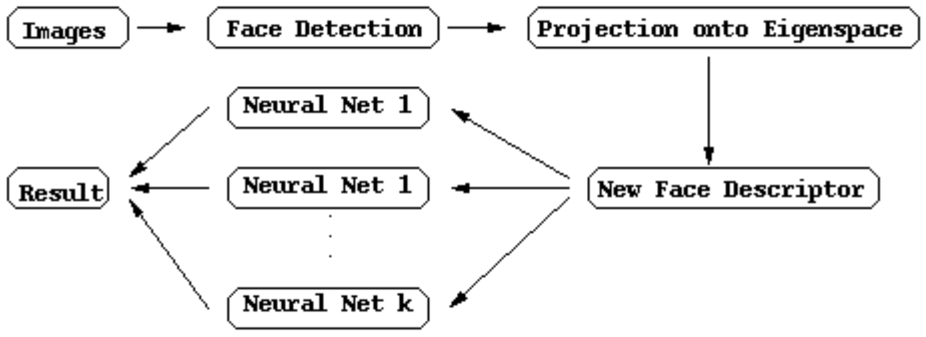

Figure 1: Face recognition structure

2. Face Detection

The problem before face recognition is face detection which finds the face in the image. We use the code from the MIT Artificial Intelligence Lab [2] (below we call it face detection code) to locate the face inside the image and cut it out for recognition.

Using the face detection code to prepare the face not only facilitates the recognition problem, but also set a uniform standard as to which part of the image should be used as face which can't be achieved through hand segmentation.

The face detection code search the face by exhaustively scanning the image at all possible scales. Kah-Kay Sun started the face pattern size from 20x20 pixels and increased it by a scalar (0.1). In our situation, we search a specific portion of the image (usually from 40% to 80%) for the face. It will help to speed the face detection procedure. If the actual face size is big, linear increase of the face pattern size will lead to coarse face location. So we increase the face pattern size arithmatically by 4 pixels at each step to locate the face more accurately.

For each window pattern, a series of 3 templates is applied sequentially to determine whether the input image is a face. If the output of any test fails to pass the predefined threshold (usually 0.5), it is rejected immediately. A face is reported only after the window pattern passes all the tests and the minimum of the 3 test results will be selected as the final output. In our environment, we can assume that there is no more than one face in the image. We set the threshold dynamically by replacing it with the maximum output up to the searching point. This greatly reduces the time cost on searching by avoiding unnecessary template tests.

If the image contains more than one person, we have to set a predefined threshold for face finding. For each face inside the image, most of the time the code will find multiple face templates for it with small location shift and size change. Finally all the face templates are packed to give only one face for each person in the image.

In our tests, most of the time the face detection code can find the face. It gives higher outputs on upright face images than those from images with orientation change. But the objective of the face detection code is to find faces inside images. Its aim is not to cut faces from images for recognition purpose. We have found that sometimes the face templates located by the code are smaller than their actual size or not central to the actual faces.

2.1 Face Model Resize

The face model used by the face detection code is 19x19. This is a little small for recognition. We use a model of 46x46 for recognition. It is still a small size and it can keep all necessary details for recognition. After the face is found, it is resized to this standard size.

2.2 Edge Removal

After the face is found and resized, a binary mask (Figure 2) is applied to eliminate those pixels on the edges. This is done in order to remove those pixels from the background. Another reason is that the model used by the face detection code is square. Our mask will shape it to rectangle which is closer to the shape of human face.

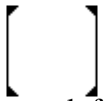
Figure 2: Binary mask for edge removal

2.3 Illumination Normalization

The input face should have roughly the same lighting as those in the database. To avoid strong or weak illumination, each face is normalized. The image is treated as a vector in the high dimensional space. Its vector length is adjusted to the vector length of average face in the face space.

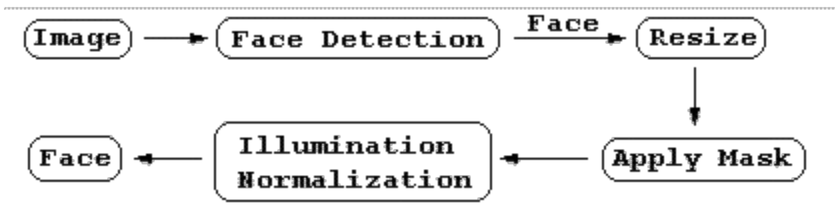

Figure 3: Face detection structure

3. Face Recognition

While the face detection problem emphasizes the commonality among faces and their difference from non-faces, the interest in face recognition is the face variation among different individuals. What we need is a mathematical description and explanation of the phenomenon that face A and face B are the same person, or face A and face C are different people.

Unlike the face detection where there are only two classes of objects, faces and non-faces, here each individual is a separate class. All faces have the same facial features and are basically very similar in overall configuration. It makes the face recognition a difficult and fine discrimination problem. Another thing which makes it more complicated is that each individual's face can have many variations because of the change in orientation, expression and lighting. While we hope that the system could handle a wide range of variation in real tests, the number of examples in learning a specific individual's face is always limited.

A face image is a two dimensional array of intensity values. In our experiments the standard size is 46x46. It can also be treated as a vector or a point in a space of dimension 2116. But face images are not randomly distributed in this high dimensional space. The fact that all faces are basically very similar to each other and have the same facial features such as eyes, nose and mouth makes all the faces a subset of the whole image space, in other words, the dimension of the face space is smaller than that of the image space.

Sirovich and Kirby [6] and Kirby and Sirovich [7] first applied the principal component analysis in efficient face representation. In this technique a new coordinate system is created for the faces where coordinates are part of the eigenvectors of a set of face images. New faces can be approximately reconstructed with only part of their projection onto the new low-dimensional space.

Matthew Turk and Alex Pentland [1] expanded the idea to face recognition. Faces are encoded by a small set of weights corresponding to their projection onto the new coordinate system, and are recognized by comparing them with those of known individuals.

3.1 Eigenspace Representation

First we prepare an initial set of face images [X1, X2, ... , Xn]. The average face of the whole face distribution is

$$X = (X1 + X2 + ... + Xn)/n$$

Then the average face is removed from each face,

$$Xi' = Xi - X, i = 1, 2, ... , n$$

The eigenvectors are calculated from the new image set [X1', X2', ... Xn'] as [Y1, Y2, ..., Yn]. These eigenvectors are orthonormal to each other. They do not correspond directly to any face features like eyes, nose and mouth. Instead they look like sort of face and are refered as eigenfaces. They are a set of important features which describe the variation in the face image set. The dimension of the complete eigenspace is n-1 because the eigenvalue of the remaining eigenface is 0. Our eigenface space is created with 593 face images (Figure 4).

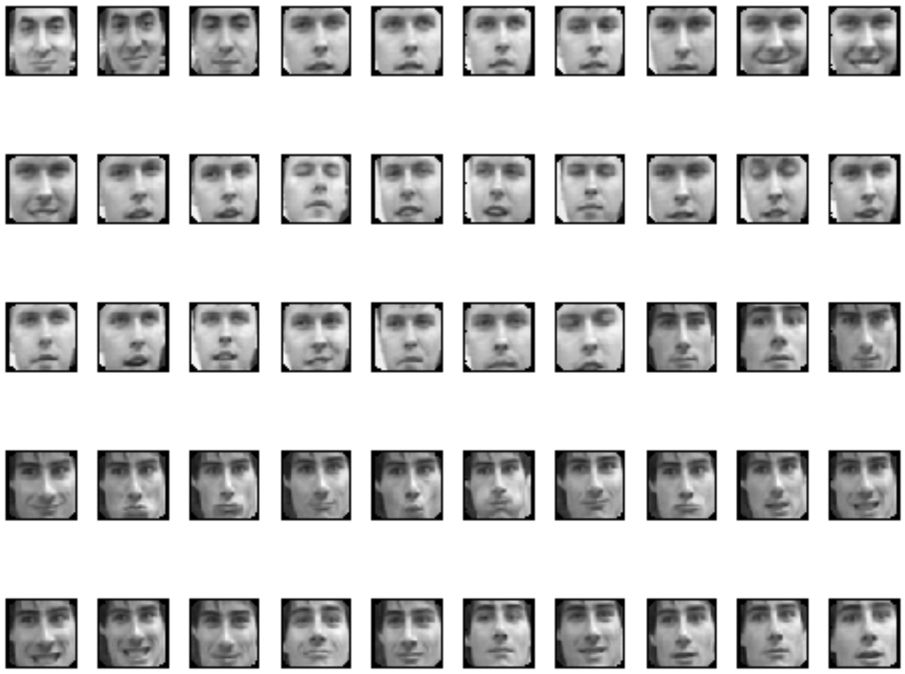
Figure 4: Part of the face images used to create eigenspacee

As a property to the eigenvector, each of them has an eigenvalue associated with it. More important, eigenvectors with bigger eigenvalues provide more information on the face variation than those with smaller eigenvalues. This is in contrast to the Euclidian space representation where all axes are of the same importance. Figure 5 and 6 show the eigenfaces with high and low eigenvalues respectively.

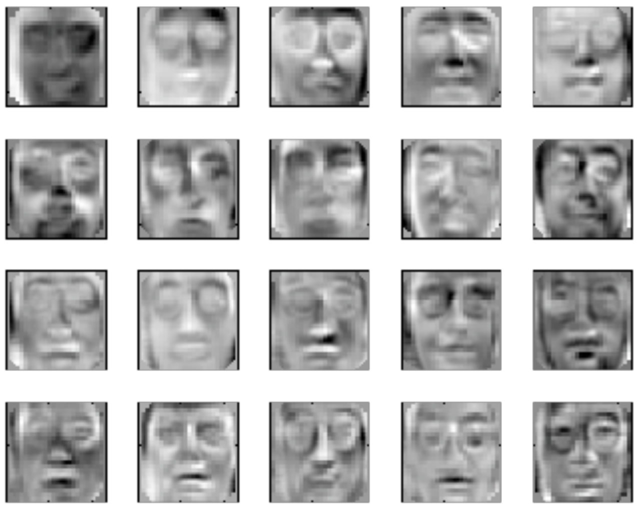
Figure 5: The first 20 eigenfaces with the highest eigenvalues

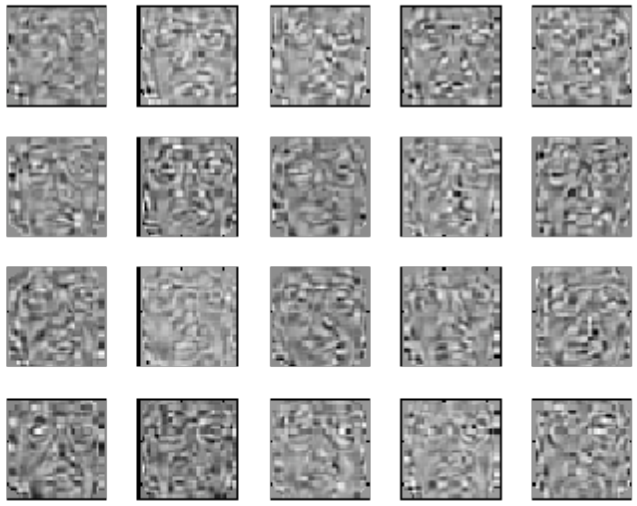
Figure 6: Eigenfaces with eigenvalues ranked from 141 to 160

From figure 7 we can see that the eigenvalue curve drops very quickly.

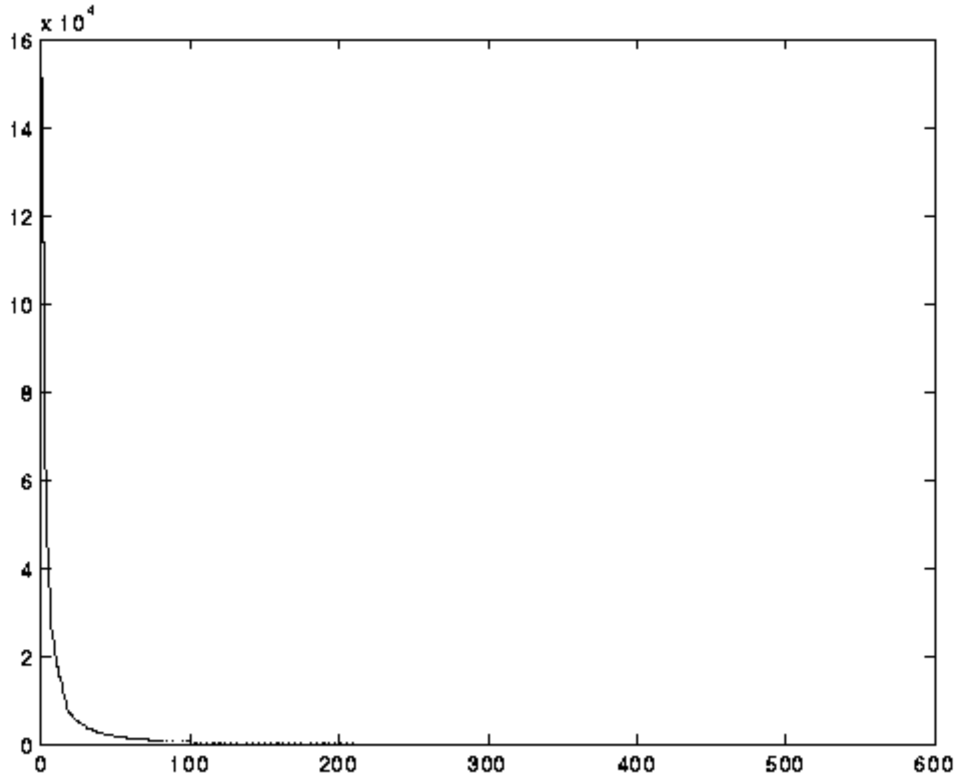

Figure 7: Eigenvalues of eigenfaces

After the eigenfaces are extracted from the covariance matrix of a set of faces, each face is projected onto the eigenface space and represented by a linear combination of the eigenfaces, or has a new descriptor corresponding to a point inside the high dimensional space with the eigenfaces as axes.

If we use all the eigenfaces to represent the faces, those in the initial image set can be completely reconstructed. But these eigenfaces are used to represent or code any faces which we try to learn or recognize. Figure 8 show the faces reconstructed from eigenfaces with high eigenvalues, while Figure 9 using those with low eigenvalues. It's clear that we should use eigenfaces with higher eigenvalues to reconstruct the faces because they provide much more information on the face variation.

Figure 8 also illustrates that while small set of eigenfaces can not reconstruct the original face, using too many eigenfaces will introduce noise to the reconstructed face. We use the first 100 eigenfaces with the highest eigenvalues. The face which we try to recognize is projected onto the 100 eigenfaces first. It produces a new description of the face with only 100 real numbers.

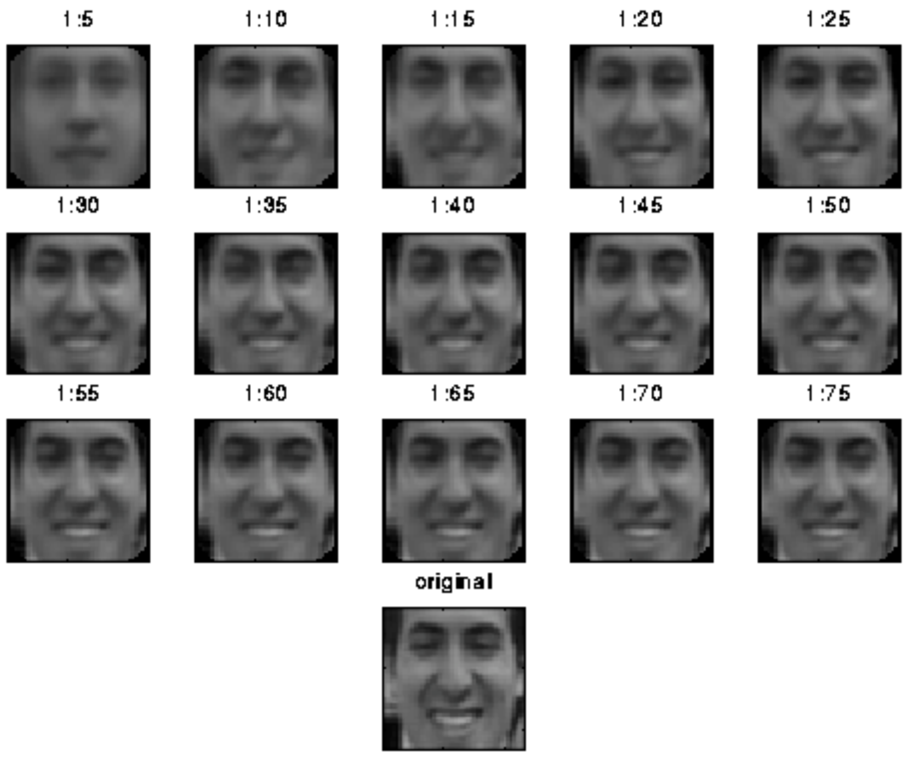

Figure 8: Faces reconstructed using eigenfaces with high eigenvalues. The label above each face is the range of eigenfaces used.

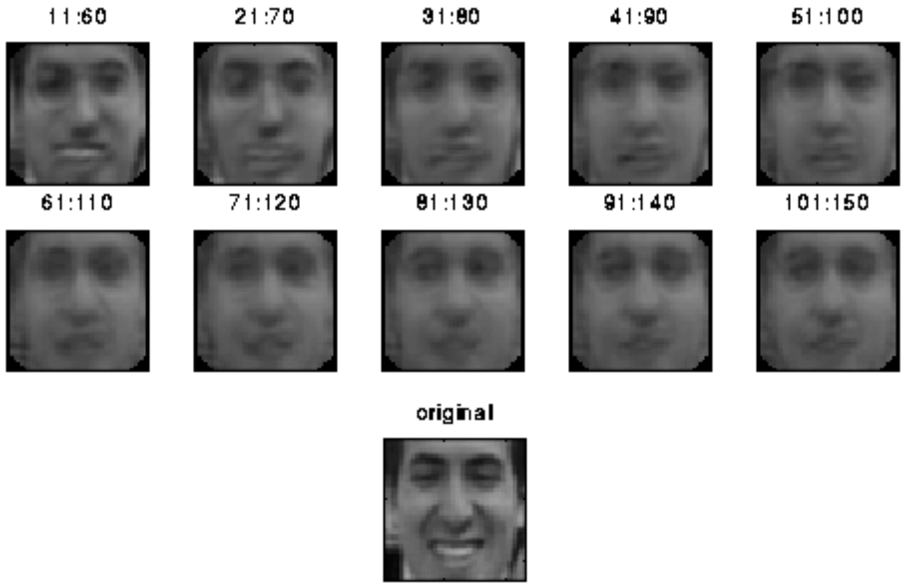

Figure 9: Faces reconstructed using eigenfaces with low eigenvalues. The label above each face is the range of eigenfaces used.

Because projection onto the eigenface space describes the variation of face distribution, it's natural to use these new descriptors of faces to classify them. Faces are recognized by comparing the new face descriptor with the face database which has been encoded in the same way. One approach to find the face pattern is to calculate the Euclidian distance between the input face descriptor and each known face model in the database. All faces of the same individual are supposed to be close to each other while different persons have different face clusters. But actually we don't have any prior knowledge on the distribution of the new face descriptors. We can't assume it to be Gaussian distribution and each individual will make one cluster. We have found that usage of this method is not sufficient in real tests.

A better approach is to recognize the face in unsupervised manner using neural network architecture. We collect typical faces from each individual, project them onto the eigenspace and neural networks learn how to classify them with the new face descriptor as input.

3.2 Neural Network

The neural network has a general backpropagation structure with three layers. The input layer has 100 nodes from the new face descriptors. The hidden layer has 10 nodes. The output unit gives a result from 0.0 to 1.0 telling how much the input face can be thought as the network's host.

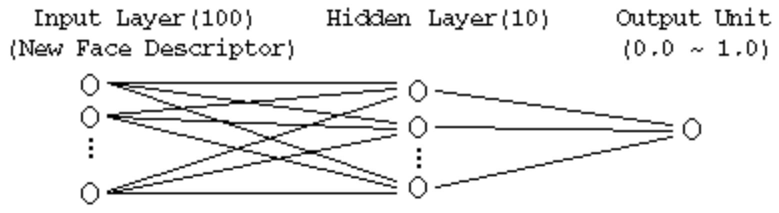

Figure 10: Neural network structure

In order to make the training of neural network easier, one neural net is created for each person. Each neural net identifies whether the input face is the network's host or not. The recognition algorithm selects the network with the maximum output. If the output of the selected network passes a predefined threshold, it will be reported as the host of the input face. Otherwise the input face will be rejected.

3.3 Training Set

After the neural network structure is set, the most important thing is to prepare the training examples. In the beginning of the training, we select a number of face images from each person that are well aligned frontal view. Any of them can represent their host clearly. All the faces here are extracted or cut by the face detection code. These faces will be used as positive examples for their own networks and negative examples for other networks.

Here we only deal with images which assume that they are always faces. In our tests, most of the time the face detection code can find the face if it exists. The database is not supposed to handle non-face images because in our situation it's unnecessary and it will make network training very difficult.

After the basic neural networks are created, we run them over new faces from the individuals in our database. If the image fails to pass the face detection test, it will be ignored. If the face detection code reports a face in the image, it will be applied to the face recogniton code. We check the recognition result to find more faces for training. Here each face will fall into one of the four categories as following:

1. Faces have high output on their own networks and low output on other networks. No action here.

2. Faces have high output on both their own networks and some other networks. These faces will be used as negative examples for other networks.

3. Faces are well-aligned frontal view and clearly represent their hosts. They have low output on their own networks. These faces will be used as both positive examples for their own networks and negative

examples for other networks.

4. Faces are not well cut and can't represent their hosts clearly. They have low output on their own networks. If they have high output on some other networks, they will be included as negative examples for other networks. Otherwise they will be ignored.

Once we get these new faces, we add them to training examples and retrain the neural networks. Recognition errors will be corrected and the total performance will be improved. While adding some examples from a specific individual will improve the performance of his own network, it will also influence the performance of other networks. In our experiments, the network training process continues until no significant recognition errors are found.

3.4 Normalize Training Set

If we use the original face descriptors from the training examples as neural network input, it will be difficult to make the network converge. What we do is to make the average of the training set to zero and unify its standard derivation.

4 Experiments

Combining the techniques listed above, we have built a real-time system which continuously gets image from camera, finds face with face detection code and recognizes it using the database which has been created with neural network. If the face is recognized as one in the database, the speaker will say his name, otherwise it will say ``no match". The experiments are conducted under roughly the same illumination condition. Wide expression variations are incorporated. The faces are basically frontal view without significant orientation change.

In our tests, each image is 200x200 pixels. We search an area from 40\% to 80% for the face. If a face is located, the recognition can be conducted instantly. A recognition rate of more than 95% has been achieved.

We can improve the recognition rate by recognizing the same person multiple times. We don't report the recognition result until most of the recognitions during a short period of time give the same result.

In the beginning our database contains 5 persons. After the initial database is created, we have added another person to the database. It is shown that the database can be expanded easily.

Individuals	***Traning Examples Number***	***Recognition Rate(%)***
Alain	142	100
Ari	202	95
Charlie	107	92.5
Josh	118	95
Peter	103	97.5
Mike	179	95
Average	142	96

5 Related Works

Henry Rowley et. al. [3] have built another neural network-based face detection system. They first preprocess the image window and then pass it through neural network to see whether it is a face. Their networks have three types of hidden units: 4 looking at 10x10 pixel subregions, 16 looking at 5x5 pixel subregions and 6 looking at 20x5 pixel subregions. These subregions are chosen to represent facial features that are important to face detection. Overlapping detections are merged. To improve the performance of their system, multiple networks are applied. They are trained under different initial condition and have different self-selected negative examples. The outputs of these networks are arbitrated to produce the final decision.

Roberto Brunelli and Tomaso Poggio [6] develop two algorithms for face recognition: geometric feature based matching and template matching. The geometric feature based matching approach extracts 35 facial features automatically such as eyebrow thickness and vertical position, nose vertical position and width, chin shape and zygomatic breadth. These features form a 35-D vector and recognition is performed with a Bayes classifier. In the template matching approach, each person is represented by an image and four masks representing eyes, nose, mouth and face. Recognition is based on the normalized cross correlation between the unclassfied image and the database images, each of which returns a vector of matching scores (one per feature). The person is classified as the one with the highest cumulative score. They also perform recognition based on single feature and features are sorted by decreasing performace as eyes, nose, mouth and whole face template.

A face recognition system Visionics FaceIt wins the ``FERET'' face recognition test 1996 hold by the US Army Research Laboratory. It was originally developed from the Computational Neuroscience Laboratory at The Rockefeller University. Their face recognition is based on factorial coding which transforms the image pattern into a large set of simpler statistically independent elements. The system finds and recognizes face in real time. User can create their own face database and add new person to the database. People can be gathered into different groups. A useful functionality is to unlock the screen if the system recognizes the person. Currently it runs under Windows 95/NT with VFW or MIL video capture device driver and IRIX system 5 with an external video camero such as SGI IndyCam.

Acknowledgements

I would like to thank to Kah-Kay Sung at MIT for providing us with the face detection code. I also thanks Alain A. Roy, Josh D. Flachsbart, Charles Frankel, Michael Russell, Roger Kahn and Peter Prokopowicz for providing the face images and helping to test the database.

References

[1] M. Turk and A. Pentland, ``Eigenfaces for recognition'', Journal of Cognitive Neuroscience, vol. 3, no. 1, pp.71-86, 1991

[2] K. Sung and T. Poggio, ``Example-based learning for view-based human face detection'', A.I. Memo No.1521, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1994

[3] Henry A. Rowley, Shumeet Baluja and Takeo Kanade, ``Human face detection in visual scenes'', CMU-CS-95-158R, Carnegie Mellon University, November 1995

[4] David Beymer and Tomaso Poggio,``Face recognition from one example view'', A.I.Memo No.1536, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1995

[5] A. Pentland, B. Moghaddam and T. Starner, ``View-based and modular eigenspaces for face recognition'', Proceedings IEEE Conf. on Computer Vision and Pattern Recognition, p.84-91, June 1994

[6] Roberto Brunelli and Tomaso Poggio, ``Face recognition: feature versus templates'', IEEE

Transactions on Pattern Analysis and Machine Intelligence, 15(10):1042-1052, 1993

[7] L. Sirovich and M. Kirby, ``Low-dimensional procedure for the characterization of human faces", Journal of the Optical Society of America A, 4(3), 519-524, 1987

[8] M. Kirby and L. Sirovich, ``Application of the Karhunen-Loeve procedure for the characterization of human faces", IEEE Transaction on Pattern Analysis and Machine Intelligence, 12(1), 1990