Using Eigenfaces to Perform Facial Identification on a College Directory

Janet Burt and David Tengdin
MSCS
St. Olaf College
1500 St. Olaf Ave, Northfield, MN 55057
burtj@stolaf.edu, dreamingfurther@gmail.com

Abstract

Face identification is a complicated puzzle that computer scientists have been working on for decades. As with many other computer vision problems, face identification has neither a well-defined testing structure or a straightforward way to compare various methods.

We implemented the eigenface approach to face recognition described by Turk and Pentland (1991) [4] in C++. We tested this method on a subset of the id picture database of St. Olaf College students, faculty, and staff. We used PCA to obtain some of the eigenvectors of our dataset, which we scaled into eigenfaces (pgm files). Given a new probe image, we pre-processed the image. We then reconstructed the image by adding linear combinations of eigenfaces to the average image. Finally, we compared our newly processed image to the images in the database to find a probable match. This comparison revealed that our smallest dataset did not produce satisfactory results.

Introduction

Face identification has been a longstanding problem for computer vision. Since before 1991 different groups and individuals have been researching good ways to robustly take various images of faces and identify them. One of the reasons that this function is of such interest is because humans have a strong and innate ability to identify faces with the set of faces that they have been exposed to from the time of birth. Indeed, studies have shown that there are entire regions of the brain that are solely concerned with face identification.

Computers excel in replicating certain functions that people perform especially well, in particular when it comes to repetitive computations. However a "face" is a rather unique object, much harder to classify than something like a box or even something as potentially variable as a table. As such, a brute force method for face identification that compared a newly found input image against a database of hundreds of thousands of facial images would be impractical. The root of the problem is that we need a way to represent face images with fewer parameters than simply the value of every pixel of given images.

Since computer scientists have been working on the face identification problem, various methods for "classifying" faces have been tried with varying levels of success. Simple things such as measuring distance between key locations on a face prove to be inadequate due to the fact that these details can change even if the face is still the same "face". Although it is significantly more computationally heavy, 3D imaging is another way that face classification has been attempted. After all, the face is a significantly 3D object so perhaps the way our brains succeed so well in face identification is through 3D object comparison. On the other hand, humans can recognize faces previously known rather well from 2D pictures. This suggests that there must be some method for treating a face as a 2D image and finding parameters to identify it as a 2D image.

One of the classic ways to go about reducing the number of values used to classify a facial image is called Principle Component Analysis (PCA). This method uses a detailed method for extracting the information that is different in each face for a given set of faces. Although basic PCA has been created and tested before, we believed that it would be valuable to re-create code to do this in C++. Additionally in our case, we worked on getting and running this code on a unique set of images, the id pictures of all the students, faculty, and staff of St. Olaf College located in Northfield, MN.

Image Pre-Processing

In order for our eigenface program to run more successfully, the images we use must all have the same dimensions. By pre-processing images, such images can be obtained. The pre-processing program takes any image and converts it into a cropped 230x230 image

portraying just the face from the original image. This allows us to remove much of the background which is not necessary in identifying individuals and ultimately leads to incorrect identifications.

The first step in this process is locating the eyes in the original images. To do this, we use the face and feature tracking program developed and described by George Arceneaux, Allison Schedin, and Andrew White. [1] If we do not find exactly two eyes, or the located eyes are in the wrong spot, we discard the image. Once we have successfully found 2 eyes in the original image, we use trigonometry to determine how much the image must be rotated for the eyes to be horizontally aligned. We then rotate the original image this amount, creating a rotated image.

Next, we find the eyes in the rotated image. Assuming the center of the original image to be the origin, we subtract this center from the locations of the eyes to calculate vectors representing the locations of the eyes with respect to the center of the image. We multiply these vectors by a rotation matrix to determine the locations of the straightened eyes (and the lengths of the representative vectors) with respect to the center of the original image. Finally, we add these vectors to the center of the rotated image to find the locations of the eyes in the rotated image.

From these locations, we calculate the distance between the eyes (j) for use as a unit in determining where to crop the face. We crop the faces to 2jx2j images where the upper left corner of the cropped image is located j/2 above and j/2 to the left of the location of the left eye. This gives us images that are all cropped correctly but do not have the same dimensions. To change all the images so they have the same dimensions, we scale them different amounts so that each image is 230x230. These 230x230 cropped images are the images that we used as input for our eigenface program.

Figure 1: Cropped Images

Principal Component Analysis (PCA)

To further understand the details of our project it would be valuable to explain the math involved in the PCA method. To begin, we store the image data in an r by c matrix, where r is the number of pixels in each image and c is the number of total images that we have. Each

column of the matrix represents a single image from the original database of pgm facial images and contains all the pixel data from these images.

We calculate an average image of all the images by averaging each row of the image matrix. This creates a vector which represents the average image.

Figure 2: Average Image

We then subtract this image from each image in our image matrix. We will call the resulting matrix our subAverage matrix, since it contains the data from the images after subtracting the average image . To find the eigenvectors of the subAverage image data, we must first calculate the covariance matrix of the subAverage matrix using the equation

$$q_{ij} = \frac{1}{N-1} \sum_{k=1}^{N} (x_{ik} - \bar{x}_i)(x_{jk} - \bar{x}_j)$$

where q_{ij} is the entry in the ith row and the jth column of the covariance matrix, N is the number of images (and the number of columns in the subAverage matrix), x is the entry in the subAverage image matrix, and x-bar is the average image for a given row.

Then we must find the eigenvalues and resulting eigenvectors of the covariance matrix. However, the dimensions of this covariance matrix would be rxr, where r is again the number of pixels in the original image, a large enough number that it is immensely time consuming to calculate this matrix directly. Instead we use the power method to find the eigenvalues and corresponding eigenvectors that we want.

The power method used to calculate eigenvectors of the covariance matrix for our subAverage image matrix begins with a random vector the size of a final eigenvector. We then multiply the transpose of our subAverage matrix by this vector, and then multiply the resulting vector by the subAverage matrix. The result is that the image vector that starts out as a random set of values becomes the eigenvector corresponding to the highest eigenvalue of the covariance matrix of the subAverage matrix. This involves a lot of math. However, there is still more to come.

We needed to find more than just the eigenvector which corresponds to the largest

eigenvalue. To find more eigenvectors, once an initial eigenvector is calculated, we remove it from the subAverage image data. To remove the eigenvector, we project all the subAverage images into the subspace orthogonal to the eigenvector being removed. The subAverage images, once projected into this subspace, are put into a new image matrix. We then use the power method again on this matrix of images to find the eigenvector corresponding to the largest eigenvalue of the new image matrix. This gives us the second eigenvector. This process is repeated until we have the number of eigenvectors that we want.

At this point we have a specified number of eigenvectors for a given dataset which are our principal components.

Eigenfaces and Coefficients

Now that we have our principle components it would be useful to be able to display these in a meaningful way for humans to visualize. We found that by multiplying the resulting eigenvectors by -8000 and adding 127 to result of the multiplication, the resulting values created pixels that can be displayed as human viewable pgm images. These images are called the eigenfaces of our dataset.

To bring this back to the original issue of figuring out variables to define an image, consider the eigenvectors the significant "dimensions" of the image dataset. These dimensions define a subspace of the original space called a facespace. At this point the eigenfaces are the human viewable versions of these eigenvectors, so we use these terms interchangeably for the remainder of this paper.

Now, by simply computing the dot product of each subAverage image and each eigenvector we get a unique set of coefficients for each image that we can use to define that image in terms of the eigenvectors. To test how well our eigenvectors map the given images we can re-create each input image from the average image with only the coefficients for that image and the eigenvectors for our dataset. How well this reconstructed image matches the original image is a measure of how accurately our eigenvalues describe the face space we have created. Below is an example of a reconstructed image (set next to its original image for comparison) from a dataset of 16 images and 7 eigenfaces.

Figure 3: Original (left), Reconstruction (right)

More eigenvectors result in better and better re-creation, or definition of the facespace. For a dataset of 16 images we can perfectly reconstruct each image with the coefficients for 15 eigenvectors. And while this is more eigenvectors than we need to get a tolerable reconstruction, even comparing coefficients for the maximum number of eigenvectors is a reduction in data for the images of a small dataset. For any given dataset there is a point, however, where the increased detail gained from computing more eigenvectors and their coefficients is marginal. This means that for any given set of images the number of eigenfaces calculated will be significantly smaller than the number of image in the set.

Identification

Our next test was to see whether or not our program could find a correlation between a new picture of a member of our sample dataset and their picture in the established dataset. To do this we appropriately pre-processed the new image, rotating, cropping, and scaling it to match the images in our small dataset of 16. Then we subtracted the average image created by our sample dataset and calculated the coefficients of this subAverage image and our eigenvectors. Finally we ran a distance function on these coefficients and each set of coefficients for images in the original dataset, taking the square root of the sum of the squares of the distances between each of the seven coefficients for our images.

If there was a noticeable similarity between the coefficients of one of the images in our dataset and the new image, that would result in a lower distance for that particular image than the distances for the other images in the dataset. Additionally if there was a measurable closeness to several images this could present several “probable” matches for the identity of the face. This could be applied on a much larger scale for a much larger dataset.

Results

There are a number of measurable results from all of the different computations we coded to implement PCA. We have the results of this code for the data presented by a sample set of 16 images from the St. Olaf profile image database and for a similar set of 70 images the the St. Olaf profile database. In the span of this project we calculated 7 eigenfaces, the resulting reconstructed faces, and attempted to identify several probe images against the set of 16 images. For the set of 70 images we calculated 7 eigenfaces to map the face space and the resulting reconstructed faces. We then calculated 40 eigenfaces and the resulting reconstructed faces of the same dataset. We did not attempt to identify any probe images against the larger dataset. This was in part due to the results of the first probe identification attempts.

As you can see in the above images for the set of 16 faces, 7 eigenfaces resulted in a number of coefficients that accurately reconstructed each face from the average face. However, when we ran our checking program with a new image for one of the individuals whose face was in the small data set, the results were disappointing. This image did not generate a distance to any of the faces that was close enough to make a matching suggestion. We did however run a similar identification on an image that was part of the set of 16 images. This was to check that our function would generate a very close distance to its identical counterpart in the small data set. It did this successfully which indicates that the distance checking algorithm is correctly implemented.

Our comparison between generating two different numbers of eigenvectors for the set of 70 images did yield some enlightening results. Below are two reconstructed images. r.1 was generated with only 7 eigenvectors to define its facespace, while r.2 was generated with 40 eigenvectors to define its facespace. We also include a copy of the initial face that is being reconstructed for reference.

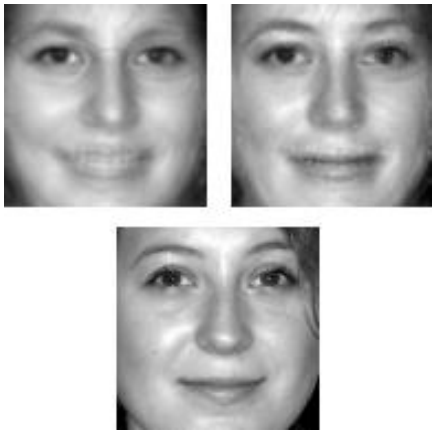

Figure 4: 7 eigenfaces (r.1) (top right), 40 eigenfaces (r.2) (top left),
Original Image (bottom)

The visual result clearly shows that 40 eigenvectors define a clearer face space (r.2) than 7 eigenvectors do (r.1). However r.1 is not terribly bad. Furthermore when we had an error calculation algorithm designed by Phinehas Bynum, and Wan-Chih Tsai [2] run between each of the reconstructed images and the original with a threshold of 5 (for pixel variation). We found that r.1 had 76% error and r.2 had 67%. Only 9% difference in error between these two images indicates an expected truth about the eigenfaces. Ultimately there are diminishing returns for the percent increase in correct reconstruction for generating more and more eigenfaces on a given dataset.

The large disappointment with the results was that we were unable to get image matching correlations between a handful of probe images checked against the small dataset of 16

images. However the reconstructed image for the probe image using the eigenvectors of the small data set was significantly worse than other reconstructed images. Therefore we conclude that the reason we had trouble correctly matching new images to any of the images in the small dataset was not due to a shortcoming in our code. We believe that the shortcoming is that there is too much variation, primarily in expression, in the set of 16 images to find good quality principal components for these faces to match new images of the same individuals as those in our dataset.

Once again, this is not surprising as much of the research on PCA indicates that it has a much harder time matching faces that vary significantly in expression and/or pose.

Conclusion

It is highly unlikely than any one method will be able to fully solve the problem of face identification. The PCA method we have implemented in C++ is not a particularly new method. However we built it specifically for use on a unique dataset that is meaningful to us. There is a great deal of room for improvement both in our own work and in the field of face identification as a whole.

More testing needs to be done with our code before comprehensive results can be presented about using this method on the St. Olaf database. Future work will indicate whether or not running PCA on this set of images is a useful way of trying to represent them.

There is another method for face identification that has been pioneered recently. This method might add more strength to the ability of an identification algorithm to identify faces that vary across pose. In a paper by Castillo and Jacobs (2009) [3], a method for using stereo matching in conjunction with epipolar geometry on 2D facial images produces some successful results. They find elements of a face that can be matched even when the faces in question are significantly rotated.

This is a particularly exciting way we might be able to improve on a PCA algorithm for several reasons. One reason is that PCA is reasonably good at identifying images that do not have rotational pose variation, however it breaks down when images to be identified are rotated horizontally. Stereo matching can identify images that have rotational pose variation and a good PCA algorithm will match images that do not have rotational pose variation. Combining these could greatly improve on the weak points of both of these methods. Castillo and Jacobs do not talk much about how successful their method is on images without pose variation so it is possible that this method breaks down where PCA is strong.

This still leaves the question of how to find correlation between images that have significant expression variation, such as a large smile. But like we said, there is much room for research and work on the problem of automated computerized face identification.

References

[1] G. Arceneaux, A. Schedin, A. White, Real Time Face and Facial Feature Detection and Tracking for Expression Recognition, Jan. 2010.

[2] P. Bynum, W. Tsai, Post-Processing of Stereo Results Using Iterations of a Bilateral Filter, Jan. 2010.

[3] C. Castillo, D. Jacobs, Using Stereo Matching with General Epipolar Geometry for 2D Face Recognition Across Pose, IEEE Transactions of Pattern Analysis and Machine Intelligence, Vol. 31, No. 12, Dec. 2009, pp. 2298-2303.

[4] M. Turk, A. Pentland, Eigenfaces for Recognition, Journal of Cognitive Neuroscience, Vol. 3, No. 1, Win. 1991, pp. 71-86.