International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:3, No:4, 2009

Face Recognition Using Eigen face Coefficients and Principal Component Analysis

Parvinder S. Sandhu, Iqbaldeep Kaur, Amit Verma, Samriti Jindal, Inderpreet Kaur, Shilpi Kumari

***Abstract*—**Face Recognition is a field of multidimensional applications. A lot of work has been done, extensively on the most of details related to face recognition. This idea of face recognition using PCA is one of them. In this paper the PCA features for Feature extraction are used and matching is done for the face under consideration with the test image using Eigen face coefficients. The crux of the work lies in optimizing Euclidean distance and paving the way to test the same algorithm using Matlab which is an efficient tool having powerful user interface along with simplicity in representing complex images.

***Keywords*—**Eigen Face, Multidimensional, Matching, PCA.

I. Introduction

THE Principal Component Analysis (PCA[1][6])[1][6] is one of the most powerful techniques that have been used in image recognition or in compression. PCA[1][6] is a statistical method under the broad title of factor analysis. The function of PCA[1][6] is to reduce the large size of the data space (variables) to the smaller intrinsic dimensionality or size of feature space (independent variables), that are needed to describe the data cost efficiently. This is the case when there is a strong correlation between observed variables. In [1] [6] various functions of PCA are discussed.

Because PCA[1][6] is a classical technique which can perform functions in the linear domain, thus the applications having linear models are much suitable.

The field of Face recognition has so many areas of application as in security, biometric systems, banks and many more that are beyond the list. Moreover, face recognition can be partitioned into Face identification, Face classification, sex determination, people surveillance in crowded areas, Video content indexing, Personal identification (e.g. Driver's License), Mug shots matching and Entrance security.

The main idea of using PCA[1][6] for face recognition is to express the large 1-D vector of pixels constructed from 2-D facial image into the compact principal components of the feature space. This can be called projection of eigenspace. Eigenspace is calculated by identifying the eigenvectors of the covariance matrix derived from a set of facial images(vectors). Once the eigenfaces have been computed, several types of decision can be made depending on the application. Face recognition is a broad term which is categorized as identification where the labels of individuals must be obtained, categorization where the face must be assigned to a certain class. Recognition of a person, where it must be decided if the individual has already been seen, PCA[1][6] computes the basis of a space which is represented by its training vectors. These basis vectors, actually eigenvectors, computed by PCA[1][6] are in the direction of the largest variance of the training vectors called eigenfaces. Each eigenface[5] can be viewed a feature. When a particular face is projected onto the face space, its vector into the face space describes the importance of each of those features in the face. The face is expressed in the face space [5] by its eigenface coefficients. We can handle a large input vector, facial image, only by taking its small weight vector in the face space. This means that we can reconstruct the original face with some error, since the dimensionality of the image space is much larger than that of face space.

Each face in the training set is transformed into the face space and its components are stored in memory. The face space has to be populated with these known faces. An input face is given to the system, and then it is projected onto the face space. The system computes its distance from all the stored faces.

Parvinder S. Sandhu is working as Professor with the Rayat & Bahra Institute Of Engineering & Bio-Technology, Mohali-India. E-Mail: parvinder.sandhu@gmail.com,

Amit Verma, Iqbaldeep Kaur and Inderpreet Kaur are Assistant Professor with the Rayat & Bahra Institute Of Engineering & Bio-Technology, Mohali-Sahauran14004.

Samriti Jindal is Lecturer with Swami Vivekananad Institute of Engineering & Technology, Banur Punjab, India.

Shilpi Kumari is an M.E Student at PEC Chandigarh-India

II. PCA Implementation

Principal component analysis (PCA) has been called one of the most valuable results from applied linear algebra. PCA is used abundantly in all forms of analysis - from neuroscience to computer graphics - because it is a simple, non-parametric method of extracting relevant information from confusing data sets. With minimal additional effort PCA provides a roadmap for how to reduce a complex data set to a lower dimension to reveal the sometimes hidden, simplified structure that often underlie it.

III. PCA: Eigenvectors Of Covariance

Researchers derive algebraic solution to PCA using linear algebra. This solution is based on an important property of eigenvector decomposition. The data set is X which is an $m \times n$

matrix, where *m* is the number of measurement types and n is the number of samples. The goal is summarized as follows:
Find some orthonormal matrix *P* where $Y = PX$ such that:

$\mathrm{CY} \equiv \frac{1}{\mathrm{n}-1} \times YYT$ is diagonalized. The rows of P are the principal components of *X*. We begin by rewriting *CY* in terms of our variable of choice *P*.

$$\mathrm{CY} = \frac{1}{\mathrm{n}-1} \times YYT \quad (1)$$

$$\mathrm{CY} = \frac{1}{\mathrm{n}-1} \times (PX)(PX)T \quad (2)$$

$$\mathrm{CY} = \frac{1}{\mathrm{n}-1} \times PXXTPT \quad (3)$$

$$\mathrm{CY} = \frac{1}{\mathrm{n}-1} \times P(XXT)PT \quad (4)$$

$$\mathrm{CY} \equiv \frac{1}{\mathrm{n}-1} \times PAPT \quad (5)$$

Note that, we defined a new matrix $A \equiv XX^T$, where *A* is symmetric The roadmap is to recognize that a symmetric matrix (A) is diagonalized by an orthogonal matrix of its eigenvectors. For a symmetric matrix:

$$A = EDE^T \quad (6)$$

Where, *D* is a diagonal matrix and *E* is a matrix of eigenvectors of *A* arranged as columns. The matrix *A* has $r \leq m$ orthonormal eigenvectors where *r* is the rank of the matrix. The rank of *A* is less than *m* when *A* is degenerate or all data occupy a subspace of dimension $r \leq m$. Maintaining the constraint of orthogonality. We can remedy this situation by selecting $(m - r)$ additional orthonormal vectors to "fill up" the matrix *E*. These additional vectors do not effect the final solution because the variances associated with these directions are zero. We select the matrix *P* to be a matrix where each row *pi* is an eigenvector of *XXT*. By this selection, $P \equiv E^T$. Substituting into Equation, we find $A = P^T DP$. With this relation ($P^{-1} = P^T$) we can finish evaluating *CY*.

$$\mathrm{CY} = \frac{1}{\mathrm{n}-1} \times PAP^T \quad (7)$$

$$\mathrm{CY} = \frac{1}{\mathrm{n}-1} \times P(P^T DP)P^T \quad (8)$$

$$\mathrm{CY} = \frac{1}{\mathrm{n}-1} \times (PP^T)D(PP^T) \quad (9)$$

$$\mathrm{CY} = \frac{1}{\mathrm{n}-1} \times (PP^{-1})D(PP^{-1}) \quad (10)$$

$$\mathrm{CY} = \frac{1}{\mathrm{n}-1} \times D(1) \quad (11)$$

It is evident that the choice of *P* diagonalizes *CY*. This was the goal for PCA.

IV. Eigen Faces

Eigen face method for human face recognition is remarkably clean and simple. The basic concept behind the Eigen face method is information reduction. When one evaluates even a small image, there is an incredible amount of information present. From all the possible things that could be represented in a given image, pictures of things that look like faces clearly represent a small portion of this image space. Because of this, we seek a method to break down pictures that will be better equipped to represent face images rather than images in general. To do this, one should generate 'base-faces' and then represent any image being analyzed by the system as a linear combination[10-12] of these base faces. Once the base faces have been chosen we have essentially reduced the complexity of the problem from one of image analysis to a standard classification problem. Each face that we wish to classify can be projected into face-space and then analyzed as a vector. A k-nearest-neighbor approach, a neural network or even a simply Euclidian distance measure can be used for classification. The technique discussed in [13-14] can be broken down into the following components:

a. Generate the eigenfaces.

b. Project training data into face-space to be used with a predetermined classification method.

c. Evaluate a projected test element by projecting it into face space and comparing to training data.

V. Generation Eigen Faces

Before any work can be done to generate the Eigen faces, sample faces are needed. These images will be used as examples of what an image in face-space looks like. These images do not necessarily need to be images of the people the system will later be used to identify (though it can help); however the image should represent variations one would expect to see in the data on which the system is expected to be used, such as head tilt/angle, a variety of shading conditions, etc. Ideally these images should contain pictures of faces at close to the same scale, although this can be accomplished through preprocessing if necessary. It is required that all of the images being used in the system, both sample and test images, be of the same size. The resulting Eigen faces will also be of this same size once they have been calculated.

It is assumed that all images being dealt with are grayscale images, with pixel intensity values ranging from 0 to 255. Suppose, there are *K* images in our data set. Each sample image will be referred to as X_i where *n* indicates that we are dealing with ith sample image ($1 <= i <= K$). Each X_i is a column vector. Generally images are thought of as pixels, each having (*x*, *y*) coordinates with (0, 0) being at the upper left corner (or one could think of an image as a matrix with *y* rows and *x* columns). Converting this to a column form is a matter of convenience, it can be done in either column or row major form, so long as it is done consistently for all sample images it will not affect the outcome. The size of the resulting X_i column vector will depend on the size of the sample images. If the sample images are *x* pixels across and *y* pixels tall, the column vector will be of size ($x * y$) x 1. These original image sizes must be remembered if one wishes to view the resulting Eigen

faces, or projections of test images into face-space. This is to allow a normal image to be constructed from a column vector of image pixels. Let X^- be the mean of all X_i ($1 <= i <= K$). This is the step to calculate an average face of the database. If one were to reinterpret the vector as a normal image, it would appear as one might expect, as shown in Fig. 1.

Fig. 1 Addition of faces

The next step is to calculate difference faces U_i such that $U_i = X_i - X^-$ (where X^- is mean) and form a matrix U, such that $U = [U_1 U_2 \ldots U_K]$. Our goal now is to generate the Eigen faces which is done by calculating the eigenvectors of the covariance matrix UU^T .This cannot be done directly as the size of UU^T is $(x * y)*(x * y)$ which is very large. Clearly, doing these calculations on a resulting matrix of this size is going to be taxing on all but the most specialized, advance hardware. To avoid this problem, a trick from linear algebra is applied[17-18]. The eigenvectors of the UU^T matrix can actually be found by considering linear combinations of the eigenvectors of the U^TU matrix. This is extremely usefully when one realizes that the size of the U^TU matrix is K x K. For practically all real world situations $K << (x*y)$. The eigenvectors wj of this matrix can be readily found through the following formula:

$$w_j = \frac{\sum_{l=1}^{K} U_l E_{lj}}{\sqrt{\lambda_j}} \tag{12}$$

Where E_{lj} is the l^{th} value of the j^{th} eigenvector of U^TU and λj is the corresponding Eigen value of w_j and E_j . The linear algebra part of this trick is given below: Let the eigenvectors of U^TU be E_j ($1 <= j <= K$) and the corresponding Eigen values be λ_j . Hence, we can write:

$$U^T U E_j = \lambda_j E_j \tag{13}$$

Multiplying both the sides by U:

$$U \times U^T U E_j = \lambda_j U E_j \tag{14}$$

Thus, $w_j = UE_j$ is the j^{th} eigenvector of UU^T with corresponding Eigen value λ_j. The fact that the Eigen values for the UU^T and U^TU are the same (though if we were going to calculate all of the Eigen values of the UU^T matrix, we could get more values, the eigenvectors of the U^TU only represent the most important subset of the Eigen values of the UU^T matrix).

VI. Procedure and Working

Function L = Create Database (TrainDatabasePath): It Align a set of face images as the training set [3] from L1 to LM) This function reshapes all 2D images of the training database into 1D column vectors. Then, it (As from table I) puts these 1D column vectors in a row to construct 2D matrix.

TABLE I
Functions

Argument	Train Database Path	Path of the training database
Returns	L	A 2D matrix, containing all 1D image vectors.
Key	Size column vector	Suppose all Z images in the training database have the same size of PxQ. So the length of 1D column vectors is PQ and 'T' will be a PQxZ 2D matrix.

We use Principle Component Analysis (PCA) [1][6] to determine the most discriminating features between images of faces. This function (function [m, A, Eigenfaces] = EigenfaceCore(L) gets a 2D matrix, containing all training image vectors and returns 3 outputs which are extracted from training database. In the argument, L is a 2D matrix, containing all 1D image vectors

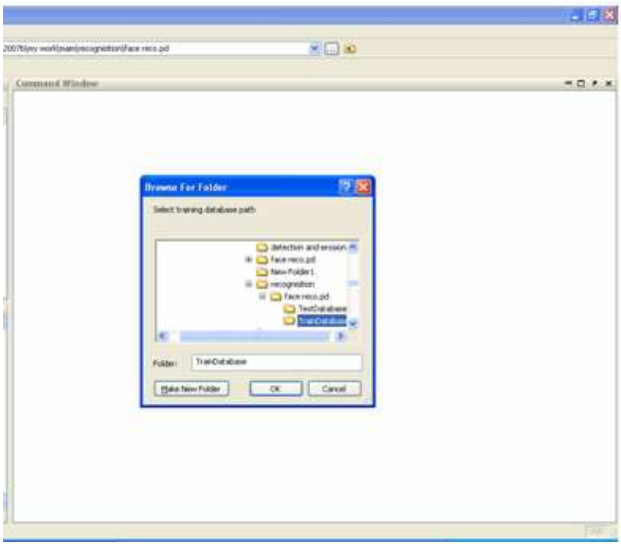

Fig. 2. Train Data Base Selection

Suppose all P images in the training database have the same

size of *MxN*.

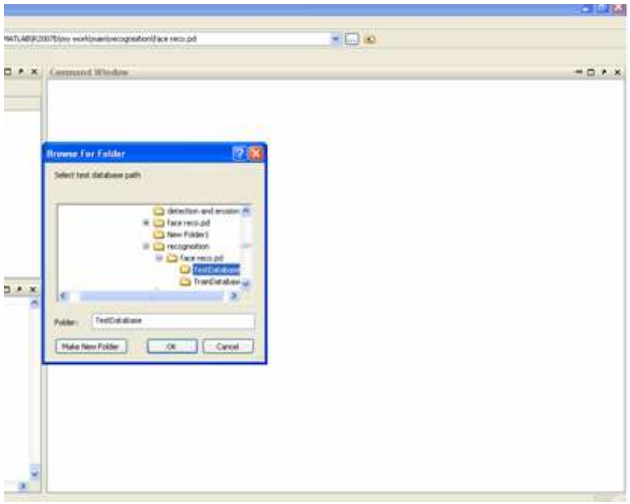

Fig. 3 Test Data Base Selection

So, the length of 1D column vectors is *M * N* and *'L'* will be a *PQxZ 2D* matrix where as m returns (*P*Qx1*) Mean of the training database Eigen faces (*P*Qx(Z-1)*) Eigen vectors of the covariance matrix of the training database *A - (P*QxZ)* Matrix of centered image vectors[15-16]. After calculating mean we calculate deviation of each image from the mean value. The next step is to Sort and eliminate eigen values.

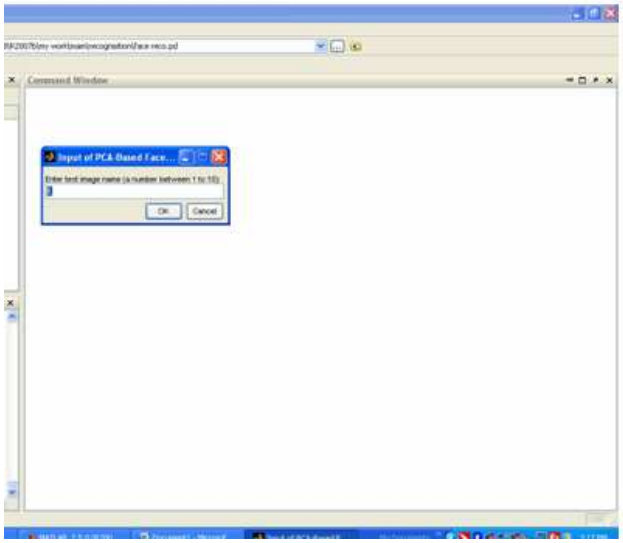

Fig. 4. Assigning fields

All Eigen values of matrix A are sorted and those that are less than a specified threshold, are eliminated. So the number of non-zero eigenvectors may be less than (Z-1).Then Calculating the eigenvectors of covariance matrix 'C' Eigenvectors of covariance matrix C (or so-called "Eigenfaces") can be recovered from A's Eigenvectors. Recognition is done by Projecting centered image [4] vectors into face space All centered images are projected into facespace by multiplying in Eigenface basis. Projected vector of each face would be its corresponding feature vector.

VII. Results and Conclusion

After the above detailed steps, the eventual step is to extract the PCA[1][6] features from test image. The crux of the work lies in calculating Euclidean distances. Euclidean distances between the projected test image and the projection of all centered training images are calculated. Moreover the objective of the whole procedure remains to have minimum distance with its corresponding image in the training database. The following figure illustrates the fore stated step.

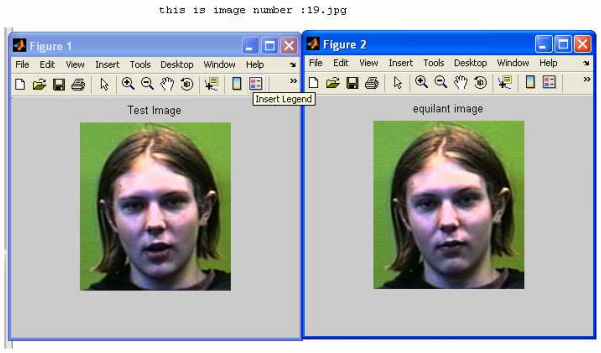

Fig. 5. Output with assigned field

VIII. Limitation of Using PCA

. Both the strength and weakness of PCA is that it is a non-parametric analysis. One only needs to make the Non-Gaussian distributed data causes PCA to fail. In exponentially distributed data the axes with the largest variance do not correspond to the underlying basis. There are no parameters to tweak and no coefficients to adjust based on user experience - the answer is unique and independent of the user. This same strength can also be viewed as a weakness. If one knows a-priori some features of the structure of a system, then it makes sense to incorporate these assumptions into a parametric algorithm - or an algorithm with selected parameters. Consider the recorded positions of a person on a ferris wheel. The probability distributions along the axes are approximately Gaussian and thus PCA finds (p1, p2), however this answer might not be optimal. The most concise form of dimensional reduction is to recognize that the phase (or angle along the ferris wheel) contains all dynamic information. Thus, the appropriate parametric algorithm is to first convert the data to the appropriately centered polar coordinates and then compute PCA. This prior non-linear transformation is sometimes termed a kernel transformation and the entire parametric algorithm is termed kernel PCA. Other common kernel transformations include Fourier and Gaussian transformations. This procedure is parametric because the user must incorporate prior knowledge of the structure in the selection of the kernel but it is also more optimal in the sense that the structure is more concisely described. Sometimes though the assumptions themselves are too stringent. One might envision

situations where the principal components need not be orthogonal. Furthermore, the distributions along each dimension (xi) need not be Gaussian. The largest variances do not correspond to the meaningful axes; thus PCA fails. This less constrained set of problems is not trivial and only recently has been solved adequately via Independent Component Analysis (ICA). The formulation is equivalent. Find a matrix P where $Y = PX$ such that CY is diagonalized. However, it abandons all assumptions except linearity, and attempts to find axes that satisfy the most formal form of redundancy reduction – statistical independence. Mathematically ICA finds a basis such that the joint probability distribution can be factorized $P(y_i, y_i) = P(y_i)P(y_i)$ for all i and j, $i \neq j$. The downside of ICA is that it is a form of nonlinear optimization, making the solution difficult to calculate in practice and potentially not unique. However ICA has been shown a very practical and powerful algorithm for solving a whole new class of problems.

REFERENCES

[1] Wendy S. Yambor Bruce A. Draper J. Ross Beveridge, "Analyzing PCA based Face Recognition Algorithms: Eigenvector Selection and Distance Measures", July 1, 2000. Available at: http://www.cs.colostate.edu/~vision/publications/eemcvcsu2000.pdf
[2] Peter Belhumeur, J. Hespanha, David Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection", IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7):771 – 720, 1997.
[3] L. Breiman. Bagging predictors. Technical Report Technical Report Number 421, Dept. of Statistics, University of California, Berkeley, 1994.
[4] D. Swets and J. Weng, "Hierarchical discriminant analysis for image retrieval", IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(5):386–401, 1999.
[5] Wendy S. Yambor, "Analysis of PCA Based and Fisher Discriminant-Based Image Recognition Algorithms", M.S. Thesis, July 2000 (Technical Report CS-00-103, Computer Science).
[6] Kyungnam Kim, "Face Recognition using Principle Component Analysis",. International Conference on Computer Vision and Pattern Recognition, pp. 586-591, 1996.
[7] http://scien.stanford.edu/class/ee368/projects2001/dropbox/project16/
[8] http://www.irc.atr.jp/%7Emlyons/pub_pdf/fg98-1.pdf
[9] http://www.kasrl.org/jaffe.html
[10] James R. Parker, J R Parker , "Algorithms for Image Processing and Computer Vision", John Wiley & Sons, 1996.
[11] Sankar K. Pal, Ashish Ghosh, Malay K. Kundu, "Soft Computing for Image Processing", Studies in Fuzziness and Soft Computing, Vol. 42, 2000.
[12] Rafael C. Gonzalez, Richard E. Woods, "Digital Image Processing", Pearson Publications, 2000.
[13] Image Processing Handbook by John C. Russ
[14] Handbook of Pattern Recognition and Image Processing by K.S. Fu and T.Y. Young
[15] Li Ma , Tieniu Tan , Yunhong Wang , Dexin Zhang " Personal Identification Based on Iris Texture Analysis" , IEEE Transactions on Pattern Analysis and Machine Intelligence , Vol. 25 No. 12, December 2003.
[16] John Carter, Mark Nixon, "An Integrated Biometric Database", available at: ieeexplore.ieee.org/iel3/1853/4826/00190224.pdf.
[17] Arun Rose, Anil Jain and Sharat Pankanti, "A Prototype Hand Geometry Based Verification System", 2nd International Conference on Audio and Video Based Person Authentication, Washington D. C., pp. 166-171, 1999.
[18] Boreki, Guilherm, Zimmer, Alessandro, "Hand Geometry Feature Extraction through Curvature Profile Analysis", XVIII Brazilian Symposium on Computer Graphics and Image Processing, SIBGRAPI, Brazil, 2005.