PCA, 2DPCA, KPCA for Face Recognition

Gayathri P
AM.EN.U4AIE20126

Jithin John
AM.EN.U4AIE20135

Lakshmi Warrier
AM.EN.U4AIE20143

M Devika
AM.EN.U4AIE20144

Nivedita Rajesh
AM.EN.U4AIE20153

B.Tech Computer Science, AI, Amrita School of Engineering, Amritapuri, Kerala

***Abstract*—This paper describes the different ways Principal Component Analysis (PCA) can be used in recognising faces. The three ways we have considered is 1D PCA, 2D PCA and Kernel PCA (KPCA). This paper discusses the difference of implementation of the mentioned approaches. The efficiency of the algorithms are also calculated and compared using Python programs.**

***Index Terms*—face recognition, principal component analysis, 2D PCA, KPCA, eigenfaces, kernel functions**

I. Introduction

Face recognition is one of the very common and important branches of computer vision. An efficient face recognition program would recognise faces in real-time from a large database which makes reducing the features an important part of the face recognition process. This algorithm may be further improvised for gender or emotion detection or more accurate outputs by including different face profiles or lighting or adding accessories like hats or spectacles, but the training data can lack these features. This, however, is outside the scope of the existing implementation.

Here we developed Python scripts to recognise faces using three kinds of PCA - 1DPCA, 2DPCA and KPCA. Principal Component Analysis or PCA is one of the very common techniques that is widely used for image recognition and compression. PCA is also a suitable model for signal processing, image processing, system and control theory, communications, because of its excellence in the linear domain. PCA is also found to be effective in prediction, redundancy removal, feature extraction, data compression, etc. [1].PCA based face recognition algorithms are simple,faster and is insensitive to slight changes made to the face which makes it efficient over other algorithms. [2]

II. LITERATURE REVIEW

PCA uses the eigenface approach to detect faces [2]. The face recognition program has three steps: face detection, feature extraction and face recognition [6]. Face detection includes the steps like appearance and location of the face in a given image, prepossessing and normalisation of the image [6]. Feature extraction includes extraction of useful features for face patches which is an output of face detection and to get significant features and feature vector by undergoing dimensionality reduction [6]. Face recognition includes a comparison of the features of the input image and the extracted feature of the image in the database and the most identical face class undergo classification [6]. However, here we work on just extraction and recognition. PCA will select and reduce the face feature based on eigenvalues of correlation matrix data [3].

Eigenvectors are referred to as principal components in PCA. Principal components, as the name suggests, contains the most relevant features of a dataset. The amount of information that an eigenvector carry is deduced from its eigenvalue. Greater the eigenvalue, greater the information content. The eigenvector associated with the largest eigenvalue is one that reflects the greatest variance in the image. That is, the smallest eigenvalue is associated with the eigenvector that finds the least variance. They decrease in an exponential fashion, meaning that roughly 90% of the total variance is contained in the first 5% to 10% of the dimensions [3].

III. FACE RECOGNITION AND PCA

The concept of face recognition using PCA is common for all the three mentioned methods. Faces are transformed to a small set of main components from the training set, called eigenfaces. Projecting a new image to the eigenface subspace can classify a person by comparing its position in eigenface space with the position of known individuals [2]. The scope of this paper can be observed as two phases:

- Initialization process:
  1) Forming mean-centered training set from the initial set of face images.
  2) Calculating eigenfaces keeping only the k highest eigenvalues.
  3) Calculating distribution in this k-dimensional space for person in the dataset by projecting the face images onto this face space.

  The face space remains the same for a given dataset and hence can be re-used to save some computation time.
- Recognition process:
  1) Calculating a set of weights for the input image by projecting the input image onto eigenfaces.
  2) Calculate the distance between the projection of the test image and each of the projections from the training set. The distance would be very less for

similar faces. When the difference is least, or, if the distance is lesser than the threshold θ, the input image belongs to that category of faces.

IV. FACE RECOGNITION USING PCA

A. *One-dimensional PCA (Fig. 2)*

1) **Formation Of Image Matrix**
Let there be T number of images in the dataset and each image be of dimension m×n.
Each image is converted to a vector of dimension mn×1 by appending column below column. These one-dimensional vectors are known as **face vectors**. From face vectors, create an image matrix, I, of dimension mn×T by stacking face vectors as its columns.

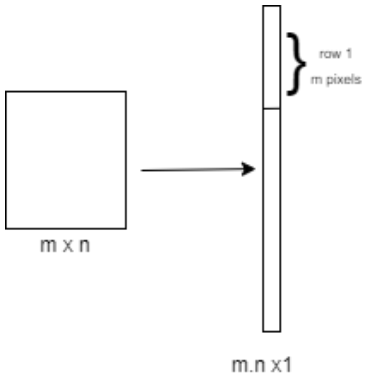

Fig. 1. Image matrix to vector

2) **Finding the mean vector and mean-subtracted image matrix**
Mean vector is computed by the formula

$$M = \frac{1}{T}\sum_{i=0}^{T} X_i \tag{1}$$

where T is the total number of images X_i corresponds to pixel values in each row of the image matrix.
Now, subtract the mean vector from the image matrix.

$$A = I - M \tag{2}$$

where, I = Image matrix, M = Mean Vector

3) **Computing the eigenvectors and eigenvalues from the covariance matrix**
Covariance matrix is calculated as,

$$C = A.A^T$$

Covariance matrix is of dimension (mn)×(mn)
The eigenvalues and eigenvectors of the covariance matrix is calculated from its characteristic equation,

$$C\vec{x} = \lambda\vec{x} \tag{3}$$

where, λ = eigenvalue, $\vec{x}$ = eigenvector

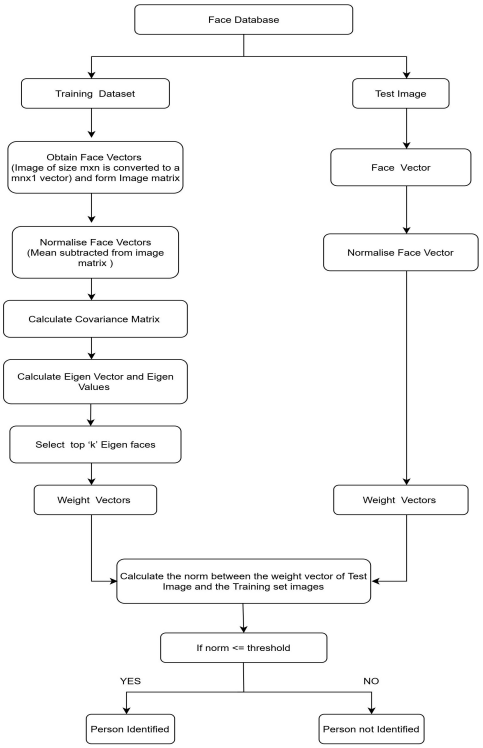

Fig. 2. 1D-PCA algorithm.

4) **Choosing eigenfaces**
The top k eigenvectors with highest eigenvalues are known as **eigenfaces**. Rest of the eigenvectors are neglected due to lack in their information content.

5) **Finding the weight matrix**
Let u be the eigenface matrix and A be the mean-subtracted image matrix. Projecting a face onto an eigenface results in a vector with scalar values known as weights.

$$D = u.A^T \tag{4}$$

D is the new feature subspace with dimension k×T.

6) **Recognising the face**
Let the image matrix of the new test face be I

$$W = u.(I - M)^T \tag{5}$$

where u = eigenfaces matrix, M = mean subtracted matrix and W = weight matrix for the current image.
The **euclidean distance** between the weight matrix (D) and W is calculated, and the image is classified into a category with lowest distance d.

B. *Two-dimensional PCA (Fig. 3)*

Rather than working on one-dimensional vectors like in one-dimensional PCA, the image processing is done on two-dimensional matrices in two-dimensional PCA.

1) **Formation Of Image Matrix**
Let there be T images in the training set with a pixel dimension of m×n. Stack these images, one on top of the other, to form an image matrix, I, of dimension T×m×n.

2) **Finding the Mean Face and Subtract it From Image Matrix**
The mean face is a matrix of same dimension as the face images, i.e, m×n. Mean face is computed using the formula

$$M = \frac{1}{T}\sum_{i=0}^{T} X_i \tag{6}$$

where, T is the number of images and X_i corresponds to the value at each pixel.

Mean face is subtracted from each image in the image matrix so as to form a mean-subtracted image matrix.

$$A = I - M \tag{7}$$

where, I is the image matrix and M is the mean face. Dimension of A is mxn×T.

3) **Computing the Covariance Matrix**
Covariance matrix is calculated using the formula,

$$C = \frac{1}{T}\sum_{i=0}^{T} A_i^T.A_i \tag{8}$$

where, T is number of images, A corresponds to each of the mean-subtracted image from the image matrix. The covariance matrix is of dimension n×n.

4) **Computing the Eigenvalues and Eigenvectors of Covariance Matrix**
The eigenvalues and eigenvectors of the covariance matrix is calculated from its characteristic equation.

$$C\vec{x} = \lambda\vec{x} \tag{9}$$

where, λ = eigenvalue, $\vec{x}$ = eigenvector

5) **Choosing top k eigenvectors and creating the eigenface matrix**
The top k eigenvectors with the highest value of eigenvalues are chosen from the list. This matrix with k eigenvectors is called eigenface matrix(E)

6) **Find Weight Matrix**
Weight matrix is computed by projecting image matrix onto the eigenface matrix. Let the eigenface matrix be E and the image matrix be I, then weight matrix is computed by the formula,

$$D = I.E \tag{10}$$

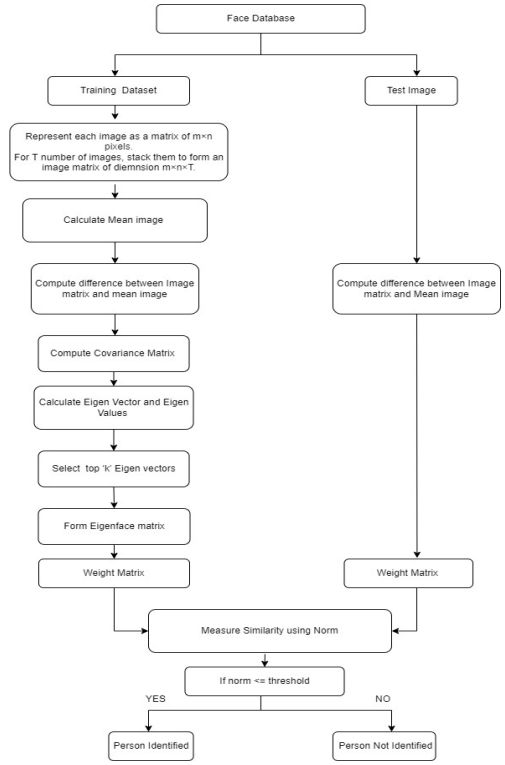

Fig. 3. 2D-PCA algorithm.

The dimension of this new feature subspace is Txmxk. This is of a much lower dimension as $k < n$.

7) **Recognising a test face**
Every test image matrix is projected onto the eigenfaces matrix(E) that was obtained through training to get the weight matrix of the current image. The norm of current weight matrix and each image matrix from the weight matrix (see eqn 10) is found and kept in track. The image will be classified into the class which has an image having with the lowest distance.

C. Kernel PCA (Fig 4)

Kernel PCA is the nonlinear form of PCA, which is promising in exposing the more complicated correlation between original high-dimensional features [4]. A first-order polynomial kernel function is nothing but the classical PCA [8].

1) **Formation of Kernel Matrix**
We transform the original D-dimensional space to an f-dimensional space using a nonlinear transformation, say $\phi(x)$, where f>D. Then each data point x_n is projected to a point $\phi(x_i)$. Assuming the new features are mean-centered, covariance of the new matrix is

$$C = \frac{1}{N}\sum_{i=0}^{N} \phi(x_i)\phi(x_i)^T \tag{11}$$

and its eigenvalues and eigenvectors are given by

$$C\vec{\mathbf{v_i}} = \lambda_i\vec{\mathbf{v_i}} \tag{12}$$

On substituting (11) in (12), we get,

$$\frac{1}{N}\sum_{i=0}^{N}\phi(x_i)\phi(x_i)^T\vec{\mathbf{v_i}} = \lambda_i\vec{\mathbf{v_i}} \tag{13}$$

Finally,

$$\vec{\mathbf{v_i}} = \frac{1}{N}\sum_{i=0}^{N}a_{ki}\phi(x_i) \tag{14}$$

where,

$$a_{ki} = \frac{1}{\lambda_k}\phi(x_i)^T\vec{\mathbf{v_i}} \tag{15}$$

We see that the eigenvector can be expressed as a linear combination of features. On substituting (14) in (13) and multiplying both sides with $\phi(x_i)$ and re-writing it after defining a matrix κ, we get,

$$\kappa^2\vec{\mathbf{a}} = N\lambda\kappa\vec{\mathbf{a}} \tag{16}$$

where the kernel matrix κ has $\kappa_{ij} = K(\phi(x_i)\phi(x_j))$
We substitute κ with $\tilde{K}$ to make the projected dataset $(\phi(x_n))$ mean-centered. This new matrix is called the Gram matrix which is given by

$$\tilde{\kappa} = \kappa - 1_N\kappa - \kappa 1_N + 1_N 1_N \tag{17}$$

where 1_N is an N×N matrix with every element $\frac{1}{N}$. Assume that the matrix is mean-centered in R^f.

2) **Choosing a kernel function**
The power of kernel methods is that we do not have to compute $\phi(x_n)$ explicitly. The kernel PCA algorithm is expressed only in terms of dot products, which allows us to construct kernels only from training data [5]. The two commonly used kernels are polynomial kernel and Gaussian kernel, out of which, we focus on Gaussian kernel with a parameter σ as:

$$K(x,y) = exp(\frac{-||x-y||^2}{2\sigma^2}) \tag{18}$$

σ is usually considered to be a value that is thrice the value of standard deviation. Kernel functions are positive semi-definite.

3) **Finding the eigenvalues, eigenvectors and eigenfaces**
Equation (16) can be simplified into a characteristic equation format as

$$\kappa\vec{\mathbf{a}} = N\lambda_i\vec{\mathbf{a}} \tag{19}$$

We can project the vectors in R^f to a lower dimensional space spanned by the top k eigenvectors y^k, and let x be a face whose projection is $\phi(x)$ in R^f, then the projection of $\phi(x)$ onto the eigenvectors y^k is the non-linear principal components corresponding to ϕ.

$$y^k.\phi(x) = \sum_{i=0}^{N}a_i(\phi(x_i).\phi(x)) = \sum_{i=0}^{N}a_i\kappa(x_i,x) \tag{20}$$

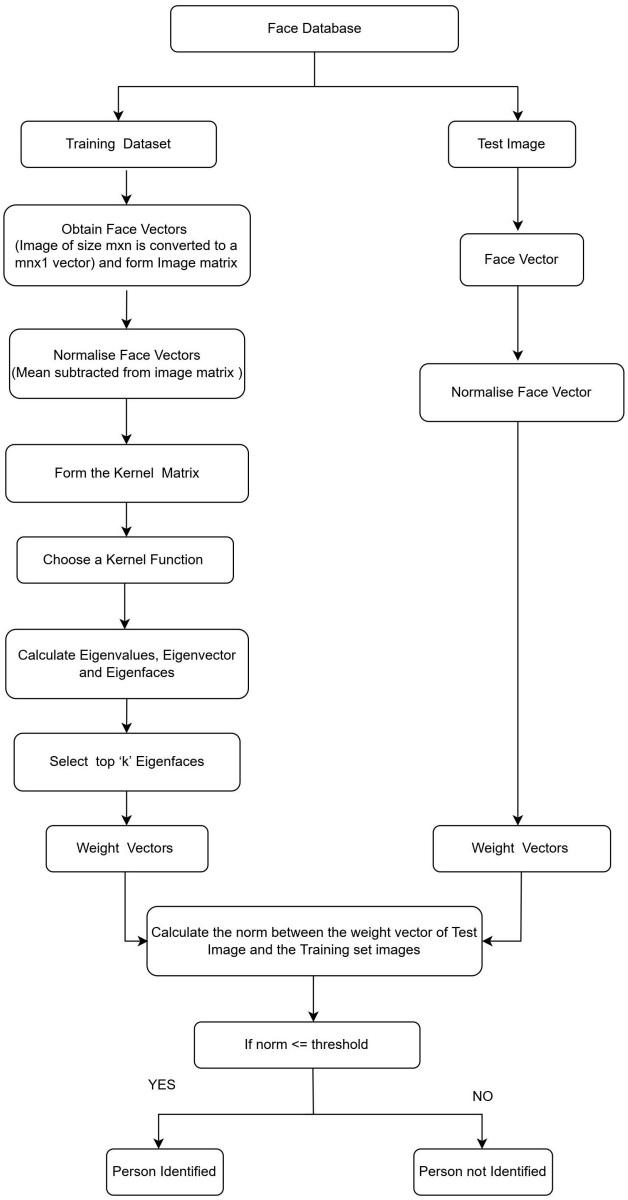

Fig. 4. Kernel-PCA algorithm.

4) **Finding Weight Matrix**
Weight matrix is computed by projecting the kernel matrix (κ) matrix onto the eigenface matrix y^k by the formula,

$$D = (y^k)^T.\kappa \tag{21}$$

5) **Recognising face**
Let the eigenface matrix be y and A be the image matrix of the new test image. The weight matrix of the current image (W) can be calculated as:

$$W = (y^k)^T.A \tag{22}$$

The **euclidean distance** between the weight matrix (D) (see eqn. 21) and W is calculated, and the image is classified into a category with lowest distance d.

V. Dataset

We used the ORL database of faces for training and testing. The dataset contains 10 gray scaled faces of 41 people each with varying lighting and facial expressions of dimensions 80x70. All the images have a dark homogeneous background with the face in an upright, frontal position.

Out of 10 pictures of each person, 8 were taken for training and 2 were taken for testing.

VI. Code implementation

The following libraries were used in code implementation:

- **Scikit-learn:** A standard Python library for a different supervised and unsupervised learning techniques.
- **NumPy:** A library that handles multi-dimensional arrays and contains functions to operate on these arrays.
- **Matplotlib:** Python package that creates static, animated, and interactive visualisations.
- **OpenCV:** A python library dealing with image processing, computer vision and machine learning.

VII. Results and Discussion

A. Evaluation Metrics

1) **Accuracy**: The percentage of correct predictions

$$Accuracy = \frac{Correct\ predictions}{Total\ predictions}\%$$

2) **Time**: The time taken for training and testing

B. Output

First 16 eigenfaces when 40 principal components were chosen is plotted as shown in fig. 5. Fig 6 shows a correct match the program found after performing 1DPCA on the dataset. Even though the face profile is different, the face was recognised correctly. 1DPCA as well as 2DPCA displays the count of correct and wrong predictions, accuracy and the time taken for the execution of the program.

C. Analysis

From table I it is noticed that when the number of principal components (N) is 10, i.e., 14% of the data is taken, the accuracy of face recognition is 93.902% in 1DPCA and 95.121% in 2DPCA. When N is 40, i.e., 57% of the data is taken, the accuracy of face recognition is 95.122% in 1DPCA and 96.341% in 2DPCA. But on considering the time taken for execution, it is seen that 2D PCA takes significantly more time than 1D PCA.

TABLE I
Analysing the accuracy and time taken

	Accuracy		**Time**	
	N=10	N=40	N=10	N=40
1DPCA	93.902%	95.122%	0.60s	0.653s
2DCPA	95.121%	96.341%	1.6s	7.3s

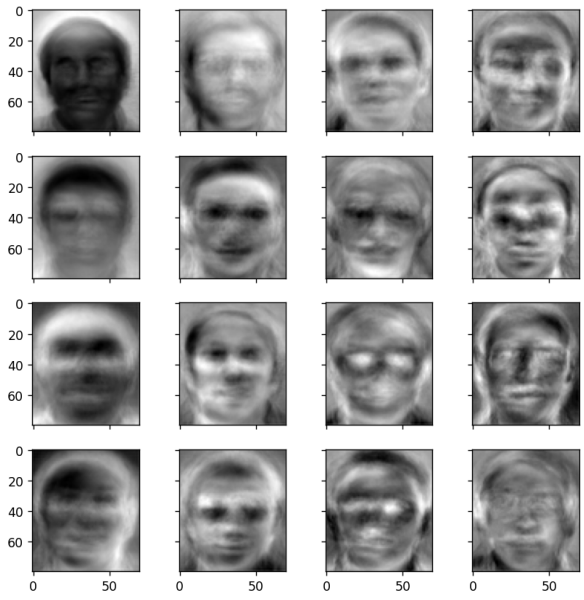

Fig. 5. Eigenfaces in 1DPCA when 40 principal components were chosen

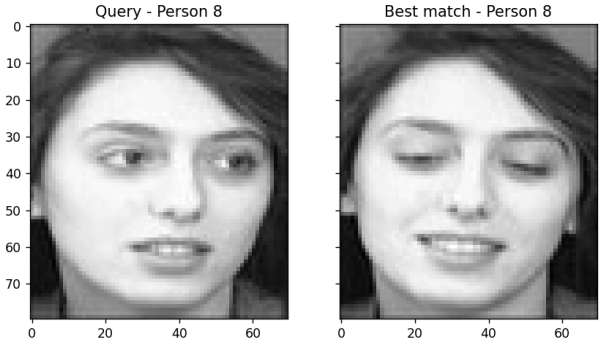

Fig. 6. Finding the best match (1DPCA)

D. Observation:

For an experiment focusing on the quality of face recognition over time taken, 2D PCA is a better choice. Even if the time taken is higher, the recognition is pretty accurate. The accuracy increment is steep among first few principal components, and later is reduced. It is also seen even that at very low values of n, we get highly accurate results.

VIII. CHALLENGES FACED

- PCA does not work well in face detection. So the face should be placed in center of the image, occupying the frame completely.
- The face image should be normalised and be of frontal-view
- Training is computationally expensive
- Deciding a threshold requires a lot of experience and patience

- Since deciding threshold is not easy and can be inaccurate, handling non-faces and unknown faces can also be inaccurate.

IX. Conclusion

We implemented and studied the different types of PCA and how it is used in face recognition. In 1D PCA, images are converted to 1-dimensional vectors, in 2D PCA, the images are let to be in 2D matrix, and in KPCA, a kernel function is used to make image matrix. We also came to a conclusion that 1D PCA takes lesser time to execute when compared to 2D PCA, but 2D PCA returns more accurate solutions.

References

[1] Face Recognition using Principle Component Analysis, Kyungnam Kim

[2] Paul, Liton & Suman, Abdulla. (2012). Face recognition using principal component analysis method. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET). 1. 135-139.

[3] B. Sugandi, I. Dewita and R. P. Hudjajanto, "Face Recognition Based on PCA and Neural Network," 2019 2nd International Conference on Applied Engineering (ICAE), 2019, pp. 1-5

[4] Wang, Quan. (2012). Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models.

[5] Ezukwoke, Kenneth & Zareian, Samaneh. (2019). KERNEL METHODS FOR PRINCIPAL COMPONENT ANALYSIS (PCA) A comparative study of classical and kernel PCA.

[6] Jasem, Esra. (2019). FACE RECOGNITION SYSTEM USING PCA, LDA, KERNEL PCA AND KERNEL LDA. 6. 249-6831.

[7] Mohammad Mohsen Ahmadinejad,Elizabeth Sherly, "A Comparative Study on PCA and KPCA Methods for Face Recognition", International Journal of Science and Research (IJSR),Index Copernicus Value (2013): 6.14.

[8] Ming-Hsuan Yang. (2002). Kernel eigenfaces vs Kernel Fischerfacees: Face recognition using kernel methods