Face Recognition of actors of action movies with Eigenfaces approach and PCA

Pedro Mendoza[1]

Abstract— **This final project mainly is about an appropiate and efficient MATLAB program for face recognition using eigenfaces approach and PCA and perform tests with a dataset of face images of actors of action movies. The face recognition task is performed by comparison of given test image with training images and obtain an image recognized. In the experiment, the user enter the access routes to the stored dataset and the test image with which he desires to validate the correct operation of the training set. Finally some results are showed (one for each actor: Brad Pitt, Arnold Schwarzenneger, Dwayne Johnson, Will Smith and Sylvester Stallone). The results demonstrate that the program recognizes some test face correctly, but performing modifications between the training folder and the test folder, a coincidence doesn't always happen. This is because the database was of very few image and then, the algorithm is not optimal.**

I. INTRODUCTION

Currently, face recognition has become an important area of research in computer vision and image analysis. Face recognition is used at all times in science fiction movies: "The protagonist looks at a camera, and the camera scans his or her face to recognize the person" [1]. Facial recognition systems are widely used in security systems and can be compared to other biometrics such as fingerprint or eye iris recognition systems [4]. It can be said that face recognition constitutes a recognition task, in which training and test images are received as inputs and as a result of comparing a test image with training images in a folder, an image recognized by the system is obtained. [1].

Defining our problem, face recognition in this case, is the challenge of recognition of a person's face by comparison of test images with training images of people's faces received as input images. This task is different than face detection where the challenge is determining if there is a face in the input image. With face recognition, we need an existing database of faces or at least a database created by ourselves. Given a test image of a face, we need to compare that image with training images and obtain an image recognized [1].

An approach to solve the problem defined is to take the new image, flatten it into a vector, and calculate the Euclidean distance between it and all of the other flattened images in the chosen database. This approach has a disadvantage and is that the larger the dataset, the slower the algorithm. However, more faces will also produce better results. The mentioned previously will be used to train in the selected dataset. To accomplish the task of face recognition, a technique called eigenfaces will be used. At the heart of eigenfaces is an unsupervised dimensionality reduction technique called principal component analysis (PCA), and it will see how it can apply this general technique to the specific task of face recognition [1].

"If we had a single $m \times n$ image, we would have to flatten it out into a single $m\dot{n} \times 1$ vector as input. For large image sizes, this might hurt speed!" [1]. This is related to the high dimensionality of the images (i.e, an $m \times n$ image is really a $m\dot{n} \times 1$ vector). "A new input might have a ton of noise and comparing each and every pixel using matrix subtraction and Euclidean distance might give us a high error and misclassifications!" [1].

Based on the above, in this work it wants to take high dimensionality images and reduce them to a smaller dimensionality, conserving the essence of the images. The aim is to study and develop an appropiate and efficient MATLAB program for face recognition using eigenfaces approach and PCA and perform tests with a dataset of face images of actors of action movies.

II. BACKGROUND / RELATED WORK

Below, after reviewing the literature, a section highlighting the most relevant work and background on facial recognition using Eigenfaces approach and PCA is presented.

A. *Face Recognition with Eigenfaces [1]*

Face Recognition with Eigenfaces is an article available on the web site `https://pythonmachinelearning.pro/face-recognition-with-eigenfaces/`.
This article discusses a popular approach to face recognition called **eigenfaces**. The essence of eigenfaces approach is an unsupervised dimensionality reduction algorithm called **Principal Components Analysis (PCA)**, which is used to reduce the dimensionality of images into something smaller. After having a smaller representation of the faces, a classifier that takes the reduced-dimension input and produces a class label is applied. For the classifier, the author of the article uses a single-layer neural network.
It should be noted that for the case of the article, a facial recognition algorithm was encoded using scikit-learn. For this, a set of data was needed. For the purposes of the article, an out-of-the-box dataset by the University of

*This work was not supported by any organization

[1]P.Mendoza is with Master's Program in Engineering, Technological University of Bolvar, Colombia

Massachusetts called Labeled Faces in the Wild (LFW) was used.

B. Face recognition using PCA [5]

This work is about a program that recognizes a face from a database of human faces using PCA. The code of the program was implemented on software MATLAB, version 7.6 (R2008a). The principal components were projected onto the eigenspace to find the eigenfaces and an unknown face was recognized from the minimum euclidean distance of projection onto all the face classes. The program uses one function called facerecog and the main file .m called face_recognition. This work is available on web site `https://www.mathworks.com/matlabcentral/fileexchange/45750-face-recognition-using-pca` of MathWorks Community.

C. Face Recognition using Principal Component Analysis Method [6]

This article mainly addresses the building of face recognition system by using Principal Component Analysis (PCA). ”PCA is a statistical approach used for reducing the number of variables in face recognition. In PCA, every image in the training set is represented as a linear combination of weighted eigenvectors called eigenfaces. These eigenvectors were obtained from covariance matrix of a training image set. The weights were found out after selecting a set of most relevant Eigenfaces. Recognition was performed by projecting a test image onto the subspace spanned by the eigenfaces and then classification was done by measuring minimum Euclidean distance. A number of experiments were done to evaluate the performance of the face recognition system. This paper used a training database of students of Electronics and Telecommunication Engineering department, Batch-2007, Rajshahi University of Engineering and Technology, Bangladesh. The article is available on International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), vol. 1, no. 9, pp. 135-139.

III. APPROACH

A. Dimensionality Reduction

Everything mentioned in the introduction of this work motivates the reason for using a dimensionality reduction technique. This technique is a type of unsupervised learning where we want to take higher-dimensional data, like images, and represent them in a lower-dimensional space [1]. An example to understand visually the technique is shown in the figure 1:

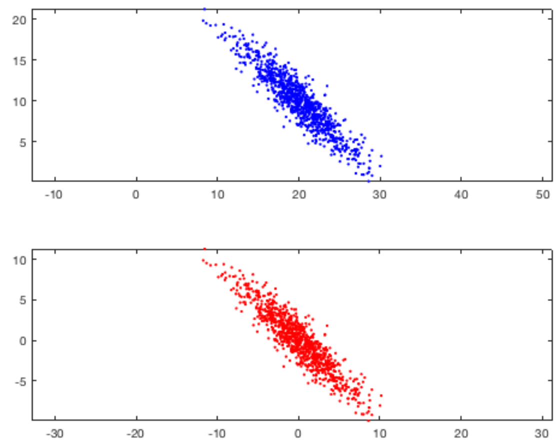


Fig. 1. Plots to understand visually dimensionality reduction technique.*

*Taken from [1].

The plots show the same data, except the bottom chart zero-centers it. Notice that these data do not have any labels associated with them because this is unsupervised learning! In this work, dimensionality reduction will reduce the data from a 2D plane to a 1D line. In the particular case in which we had 3D data, we could reduce it down to a 2D plane or even a 1D line [1].

The dimensionality reduction can be defined as the process of reducing the number of random variables under consideration by obtaining a set of principal variables [7]. The idea of all dimensionality reduction techniques is to find some hyperplane, a higher-dimensional line, to project the points onto. To understand better this affirmation, we can imagine a projection as taking a flashlight perpendicular to the hyperplane were project onto and plotting where the shadows fall on that hyperplane. For example, in the data of the above plots, if we wanted to project our points onto the x-axis, then we pretend each point is a ball and our flashlight would point directly down or up (i.e, perpendicular to the x-axis) and the shadows of the points would fall on the x-axis. This is a projection [1].

In the case of face images (i.e 2D case), it wants to find a line to project points onto. After projecting the points, it has data in 1D instead of 2D. ”The different types of dimensionality reduction are all about figuring out which of these hyperplanes to select: there are an infinite number of them!” [1].

B. Principal Component Analysis (PCA)

One of the techniques of dimensionality reduction is Principal Component Analysis (PCA). This technique is optimal in the sense that it represents the variability of the training data with as few dimensions as possible. For example, a tiny 100×100 pixel grayscale image has

10,000 dimensions, and this can be considered a point in a 10,000-dimensional space. Other example is a megapixel image that has dimensions in millions. Thus, ”with such high dimensionality, it is no surprise that dimensionality reduction comes in handy in many computer vision applications” [2]. After applying PCA, a projection matrix is obtained, which can be seen as a change of coordinates to a coordinate system where the coordinates are in descending order of importance [2].

In the case of image data, to apply PCA, it is necessary that the images are converted to a one-dimensional vector representation using, for example, Numpy’s flatten() method of python. However, there are other ways to convert images to one-dimensional vector representation, such as Numpy’s reshape() method of python [2]. In this work, reshape() function of MATLAB will be used.

Using Numpy’s flatten() method of python, ”the flattened images are collected in a single matrix by stacking them, one row for each image. The rows are then centered relative to the mean image before the computation of the dominant directions” [2]. To find the principal components, singular value decomposition (SVD) is usually used, but in the case that the dimensionality is high, a useful trick is used instead the SVD, because SVD computation will be very slow [2]. In the section called *Experiment* of this work, one way (different to SVD) to find the principal components will be shown.

The principal idea of PCA is to select one hyperplane such that when all the points are projected onto it, they are maximally spread out. I.e, it wants to find the axis of maximal variance. Considering the figure 1, a potencial axis is the x-axis or y-axis, but, in both cases, there is not the best axis. Despite this, if we pick a line that cuts through the data diagonally, that will be the axis where the data would be most spread [1]. This fact can be shown in the following figure (2):

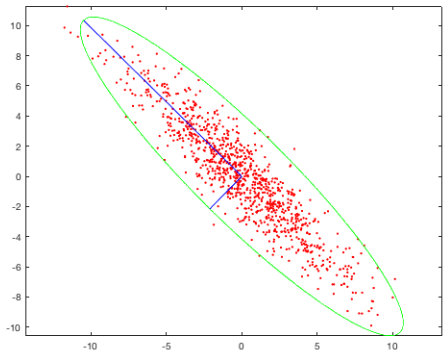


Fig. 2. Line that cuts through the data diagonally. Taken from [1].

In the figure, it observes that the longer blue axis is the correct axis. Projecting the points onto this axis, they would be maximally spread. To discover this axis, eigenvectors are used. From here, the name of eigenfaces comes [1].

In this work, it computes the covariance matrix of the data and consider that covariance matrixs largest eigenvectors, using the Kaiser’s rule that will be explained later. Those eigenvectors are the principal axes and the axes that it projects the data of the experiment onto to reduce dimensions. Thus, it can take high-dimensional data and reduce it down to a lower dimension by selecting the largest eigenvectors of the covariance matrix and projecting onto those eigenvectors [1].

Since we’re computing the axes of maximum spread, the most important aspects of the data must be retained. The PCA technique will help speed up the computations and be robust to noise and variation [1].

C. Singular Value Decomposition (SVD)

The Singular Value Decomposition (SVD), in the field of linear algebra, is defined as a factorization of a real or complex matrix. ”It is the generalization of the eigendecomposition of a positive semidefinite normal matrix (for example, a symmetric matrix with positive eigenvalues) to any $m \times n$ matrix via an extension of the polar decomposition” [8].

Then, any matrix A with size $m \times n$ can be written as the product of 3 matrices, thus [3]:

$$A = UDV^T \tag{1}$$

where:

- $U \rightarrow$ An $m \times m$ orthonormal matrix ($UU^T = I$).
- $D \rightarrow$ An $m \times n$ diagonal matrix. Where its diagonal elements, $\sigma_1, \sigma_2, ...$, are called the **singular values** of A, and satisfy $\sigma_1 \geq \sigma_2 \geq ... \geq 0$.
- $V \rightarrow$ An $n \times n$ orthonormal matrix ($VV^T = I$).

For example: If $m > n$:

$$\begin{array}{ccccccc} A & & U & & D & & V^{\mathrm{T}} \\ \begin{bmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{bmatrix} & = & \begin{bmatrix} \uparrow & \uparrow & \uparrow & & \uparrow \\ | & | & | & & | \\ u_1 & u_2 & u_3 & \cdots & u_m \\ | & | & | & & | \\ \downarrow & \downarrow & \downarrow & & \downarrow \end{bmatrix} & & \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_n \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & & \begin{bmatrix} \leftarrow & v_1^T & \rightarrow \\ \vdots & \vdots & \vdots \\ \leftarrow & v_n^T & \rightarrow \end{bmatrix} \end{array}$$

Fig. 3. Example of SVD when $m > n$. Taken from [3].

A better visualization of the matrix multiplications in SVD is shown in the figure 4:

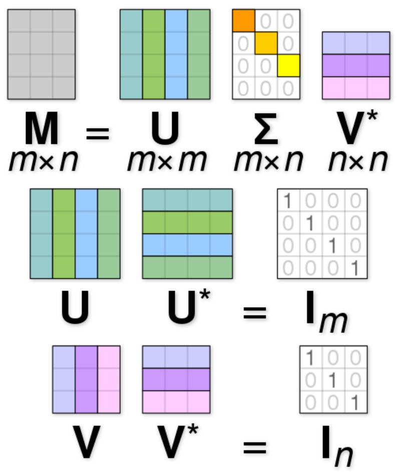


Fig. 4. Visualization of the matrix multiplications in singular value decomposition.**

**Taken from [8].

The Singular Value Decomposition has some important properties. These properties are [3]:

- ”The rank of matrix A is equal to the number of nonzero singular values σ_i”.
- ”A square $(n \times n)$ matrix A is singular iff at least one of its singular values $\sigma_1, ..., \sigma_n$ is zero”.

Interpreting geometrically the singular value decomposition, if A is a square $n \times n$ matrix,

$$\underset{A}{\begin{bmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{bmatrix}} = \underset{U}{\begin{bmatrix} \uparrow & \cdots & \uparrow \\ u_1 & \cdots & u_n \\ \downarrow & \cdots & \downarrow \end{bmatrix}} \underset{D}{\begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_n \end{bmatrix}} \underset{V^T}{\begin{bmatrix} \leftarrow & v_1^T & \rightarrow \\ \vdots & \vdots & \vdots \\ \leftarrow & v_n^T & \rightarrow \end{bmatrix}}$$

Fig. 5. SVD case with a square $n \times n$ matrix. Taken from [3].

- $U \rightarrow$ Unitary matrix: rotation.
- $D \rightarrow$ Scale matrix.
- V and $V^T \rightarrow$ Unitary matrices.

According to the above, an arbitrary n-D linear transformation equals to a rotation (plus perhaps a flip), followed by a scale transformation, followed by a rotation [3].

D. Aside on Face Detection

In addition to the detection of faces, a calculation of eigenfaces is done taking into account the eigenvalues and the eigenvectors of each face image, as will be shown later (Experiment section). In the final part of this work, two faces (one training face image and one test face image) are compared by projecting the images into facespace and measuring the Euclidean distance between them [5].
Coming up next, Kaiser’s rule will be enunciated, which will be used to find how many principal components should be taken.

E. Kaiser’s Rule

In the factorial analysis of a dataset, the factorial matrix indicates the relationship between the factors and the variables. This matrix can present a number of factors greater than necessary to explain the structure of the original data. In general, it is observed that there is a reduced set of factors, the first ones, which explain the most of the total variability. The rest of the factors contribute relatively little. Usually, it presents a problem related to the determination of the number of factors that must be conserved. To solve it, there is a well-known and used criterion called the Kaiser’s rule that states the following: ”keep only those factors whose eigenvalues are greater than unity” [9].

IV. EXPERIMENT [5]

Initially, different images of the selected training set will be aligned (X1, X2, X3, X4, ..., XN). For this, all the images of the training set will be taken in 2D format and transformed into 1D columns in order to facilitate a displacement in the form of rows of all these columns 1D and achieve a 2D Matrix called ’X’.

In the MATLAB code, *Datapath* is the path of the data images used for training and X is a 2D matrix containing all 1D image vectors. All P images in the training database have the same size of $M \times N$, so the length of 1D column vectors is $M \times N$ and ’X’ will be a $(M \times N) \times P$ 2D matrix. For this project, images of 300 x 300 pixels will be used. The total number of images stored in the route of our training set is determined. It has a folder of $P \times 1$ where four fields are found: name, byte, date and isdir of all files present in the directory Datapath.

Then, the 2D matrix ’X’ that will contain the images of the training set will be created. In this part, the following steps are performed:

1) The file of image is read.
2) It converts the image from RGB format to grayscale to process it more easily.
3) It stores the dimensions of the training image.
4) Reshaping 2D images into 1D image vectors.
5) ’X’ is the image matrix with columns getting added for each image.

After, it calculates Average, A and EigenFaces. The descriptions are below:

- Average $\rightarrow (M \times N) \times 1$. Mean of the training images.
- A $\rightarrow (M \times N) \times P$. Matrix of image vectors after each vector getting subtracted from the mean vector Average.
- EigenFaces $\rightarrow (M \times N) \times P'$. It has P’ eigenvectors of covariance matrix (C) of training database X, and where P’ is the number of eigenvalues of C that best represent the feature set.

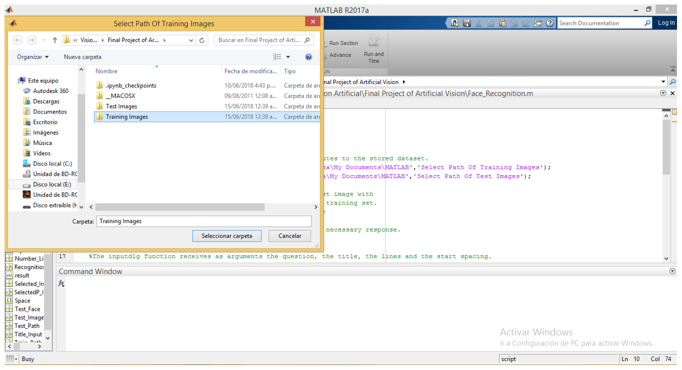
It proceeds to calculate ’Average’ as the mean image vector of ’X’. Next, ’A’ matrix is calculated after subtraction of all image vectors from the mean image vector ’Average’.

Followed by the above, the calculation of EigenFaces is performed taking into account the following aspects:

- It knows that for a $M \times N$ matrix, the maximum number of non-zero eigenvalues that its covariance matrix can have is $min[M-1, N-1]$.
- As the number of dimensions (pixels) of each image vector is very high compared to number of test images, the number of non-zero eigenvalues of C will be maximum P-1 (P being the number of test images).
- If it calculates eigenvalues and eigenvectors of $C = A * A'$, then it will be very time consuming as well as memory.
- So it calculates eigenvalues and eigenvectors of $L = A' * A$, whose eigenvectors will be linearly related to eigenvectors of C.
- These eigenvectors being calculated from non-zero eigenvalues of C, will represent the best feature sets.
- Kaiser's rule is used to find how many principal components (eigenvectors) must be taken. If corresponding eigenvalue is greater than 1, then the eigenvector will be chosen for creating eigenface.

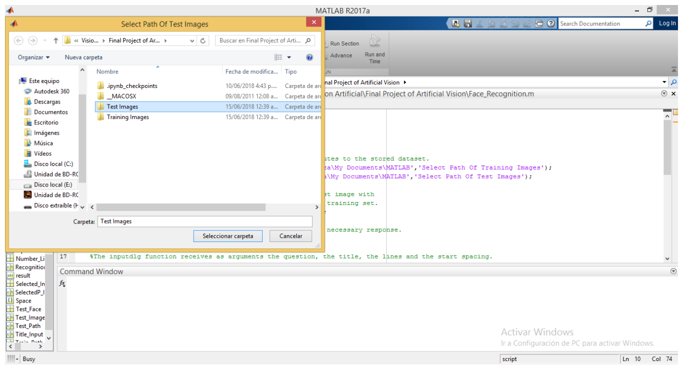
In the final part of the experiment, it compares two faces by projecting the images into facespace and measuring the euclidean distance between them, extracting before PCA features of the test image.

To perform the experiment, the user must enter the access routes to the stored dataset (training path and test path). After, the user will be asked to enter the test image with which he desires to validate the correct operation of the training set. This is shown in the following figures:



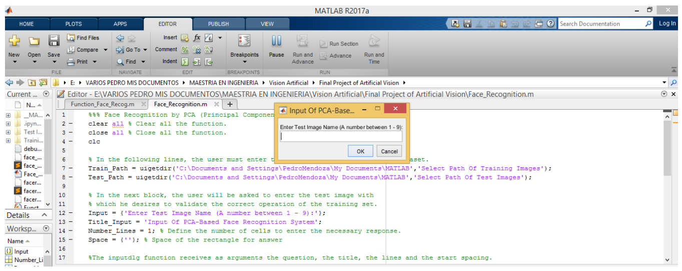
Nombre	Fecha de modifica...	Tipo
.ipynb_checkpoints	10/06/2018 4:43 p....	Carpeta de ar
_MACOSX	09/08/2011 12:08 a....	Carpeta de ar
Test Images	15/06/2018 12:39 a....	Carpeta de ar
Training Images	15/06/2018 12:39 a....	Carpeta de ar

Fig. 6. Selection of path of training images.



Nombre	Fecha de modifica...	Tipo
.ipynb_checkpoints	10/06/2018 4:43 p....	Carpeta de ar
_MACOSX	09/08/2011 12:08 a....	Carpeta de ar
Test Images	15/06/2018 12:39 a....	Carpeta de ar
Training Images	15/06/2018 12:39 a....	Carpeta de ar

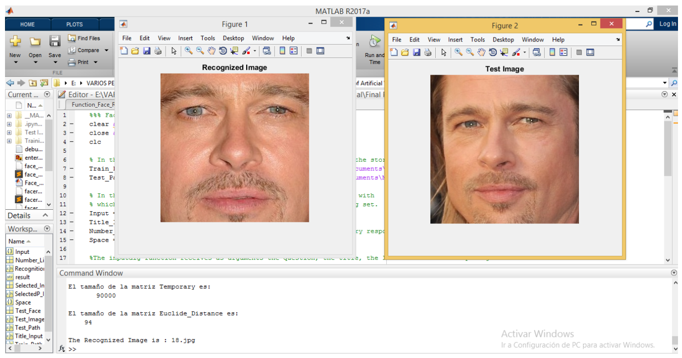
Fig. 7. Selection of path of test images.


MATLAB R2017a
Editor - E:\VARIOS PEDRO MIS DOCUMENTOS\MAESTRIA EN INGENIERIA\Vision Artificial\Final Project of Artificial Vision\Face_Recognition.m
Function_Face_Recog.m
Face_Recognition.m
Input Of PCA-Base...
Enter Test Image Name (A number between 1 - 9):
OK
Cancel

```
%%% Face Recognition by PCA (Principal Componen
clear all % Clear all the function.
close all % Close all the function.
clc

% In the following lines, the user must enter t                                    aset.
Train_Path = uigetdir('C:\Documents and Settings\PedroMendoza\My Documents\MATLAB','Select Path Of Training Images');
Test_Path = uigetdir('C:\Documents and Settings\PedroMendoza\My Documents\MATLAB','Select Path Of Test Images');

% In the next block, the user will be asked to enter the test image with
% which he desires to validate the correct operation of the training set.
Input = {'Enter Test Image Name (A number between 1 - 9):'};
Title_Input = 'Input Of PCA-Based Face Recognition System';
Number_Lines = 1; % Define the number of cells to enter the necessary response.
Space = {''}; % Space of the rectangle for answer

%The inputdlg function receives as arguments the question, the title, the lines and the start spacing.
```

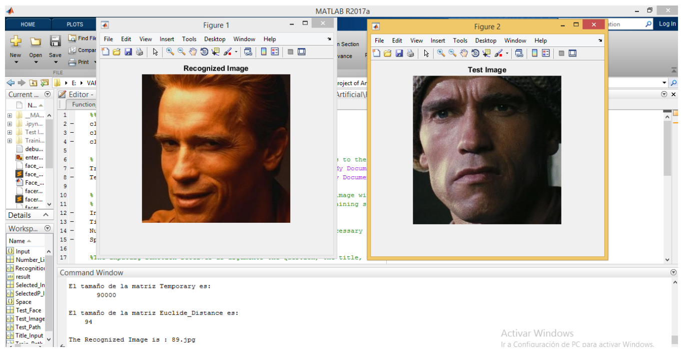
Fig. 8. Entry of test image name.

Some results are shown below:



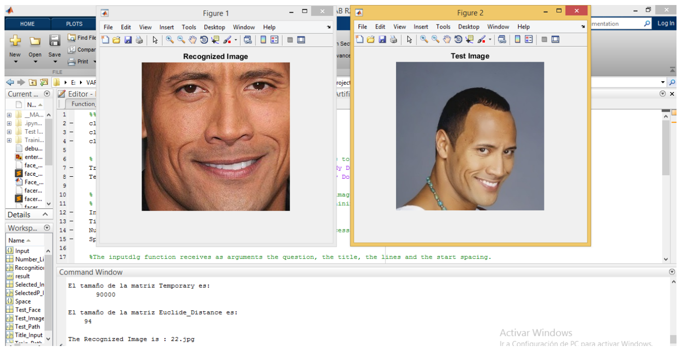
```
El tamaño de la matriz Temporary es:
      90000

El tamaño de la matriz Euclide_Distance es:
      94

The Recognized Image is : 18.jpg
```

Fig. 9. Result for Brad Pitt.



```
El tamaño de la matriz Temporary es:
      90000

El tamaño de la matriz Euclide_Distance es:
      94

The Recognized Image is : 89.jpg
```

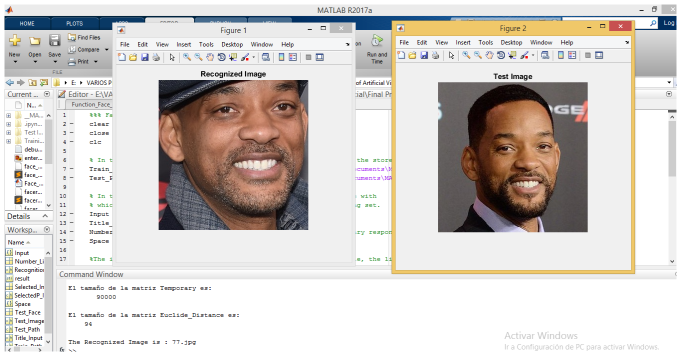
Fig. 10. Result for Arnold Schwarzenegger.



```
%The inputdlg function receives as arguments the question, the title, the lines and the start spacing.
```

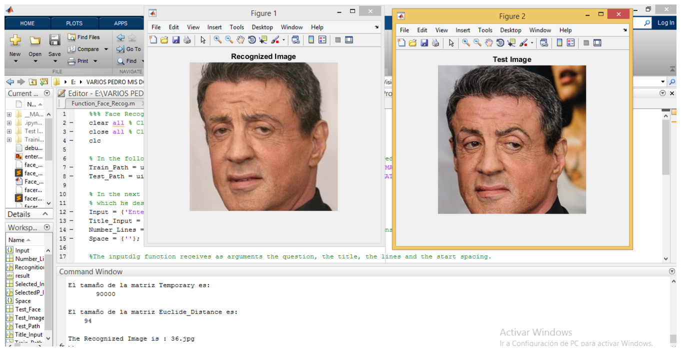
```
El tamaño de la matriz Temporary es:
      90000

El tamaño de la matriz Euclide_Distance es:
    94

The Recognized Image is : 22.jpg
```

Fig. 11. Result for Dwayne Johnson.



```
El tamaño de la matriz Temporary es:
      90000

El tamaño de la matriz Euclide_Distance es:
    94

The Recognized Image is : 77.jpg
```

Fig. 12. Result for Will Smith.



```
%The inputdlg function receives as arguments the question, the title, the lines and the start spacing.
```

```
El tamaño de la matriz Temporary es:
      90000

El tamaño de la matriz Euclide_Distance es:
    94

The Recognized Image is : 36.jpg
```

Fig. 13. Result for Sylvester Stallone.

These results show that the program works to recognize some test faces correctly when compared to the training faces. However, modifications were made exchanging some images of faces of the training folder and the test folder, noting that in all cases there was not a coincidence in the recognition of the faces.

V. CONCLUSIONS

From the results obtained from the experiment and the implementation of the program, it could be noted that this is not totally optimal for all the test images. This is because the database taken for the implementation was of very few images so that the algorithm did not have a good training. Occasionally, Kaiser's rule tends to overestimate the number of factors necessary to explain the total variability of a dataset, therefore other methods can be used, such as the likelihood ratio method, which is a goodness-of-fit criterion for the use of the method of extraction of maximum likelihood, which is distributed according to Chi-square.

REFERENCES

[1] M. Deshpande, *Face Recognition with Eigenfaces* [online]. Python Machine Learning, ZENVA, 2017. Disponible en: https://pythonmachinelearning.pro/face-recognition-with-eigenfaces/

[2] J. E. Solem, "Chapter 1: Basic Image Handling and Processing", in *Programming Computer Vision with Python: Tools and algorithms for analyzing images*, First Edition, Andy Oram and Mike Hendrickson, Ed. O'Reilly Media, Inc., Sebastopol, 2012.

[3] T. K. Marks, *Linear Algebra Review* [online]. University of California San Diego (UCSD). Disponible en: `http://cseweb.ucsd.edu/classes/wi05/cse252a/linear_algebra_review.pdf`

[4] Wikipedia, *Facial recognition system* [online]. Wikipedia, the free encyclopedia. Last edited on 2 June 2018, at 06:21. Disponible en: `https://en.wikipedia.org/wiki/Facial_recognition_system`

[5] B. Dash, *Face recognition using PCA* [online]. MathWorks. Updated 04 Mar 2014. Disponible en: `https://www.mathworks.com/matlabcentral/fileexchange/45750-face-recognition-using-pca`

[6] L. C. Paul, A. Al Sumam, "Face recognition using principal component analysis method", *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 1, no. 9, pp. 135-139, November 2012.

[7] Wikipedia, *Dimensionality reduction* [online]. Wikipedia, the free encyclopedia. Last edited on 20 April 2018, at 20:00. Disponible en: `https://en.wikipedia.org/wiki/Dimensionality_reduction`

[8] Wikipedia, *Singular-value decomposition* [online]. Wikipedia, the free encyclopedia. Last edition on 6 June 2018, at 21:02.

[9] Secci de Geografia, Departament de Geografia, Història de L'Art, *Análisis factorial* [online]. Universitat de Girona. Disponible en: `http://www3.udg.edu/dghha/cat/secciogeografia/prac/models/factorial(5).htm`