An Efficient Technique for Global Facial Recognition using Python and OpenCV in 2D Images

JOSÉ CADENA, MANUEL VILLA, MAIRA MARTÍNEZ, JAIME ACURIO, LUIS CHACÓN
Universidad Técnica de Cotopaxi,
Facultad de Ciencias de la Ingeniería y Aplicadas,
Av. Simón Rodríguez, Latacunga
ECUADOR

Abstract: - The present work is an investigation that deals with the use of efficient techniques for global facial recognition using Python and OpenCV carried out in the Information Systems career of the Faculty of Engineering and Applied Sciences of the Technical University of Cotopaxi. We work with a database of 2D faces corresponding to the students of the Information Systems career that served for the analysis and comparison of the three techniques used (Fisherfaces, EigenFaces, LBPH). The objective of our work is to determine an efficient technique that contributes to the area of global facial recognition, contributing significantly to the field of university security, taking into consideration the saving of resources for future implementations. Finishing this work with the respective analysis and interpretation of the research results, it has been determined that of the three techniques studied, LBPH has the best results both in training time and in face recognition efficiency, reaching near 100% in our tests.

Key-Words: - Facial recognition, OpenCV, Python, Fisherfaces, LBPH.

Received: October 9, 2022. Revised: January 13, 2023. Accepted: February 15, 2023. Published: March 23, 2023.

1 Introduction

This research project describes the facial recognition techniques used, in addition to working with Python and OpenCV, the same ones that apply artificial intelligence algorithms in 2D images.

The objective of this work is to find an efficient technique for a 2D facial recognition process and to test its effectiveness through a prototype designed with this technique.

Face recognition systems are a problem that is still the subject of research because a large part of the problem is due to factors that can affect the system at the time of recognition.

According to [1], some factors hinder the facial recognition process, such as occlusions that prevent good recognition, whether due to gestures made by the person, elements that cover the face, lighting, distance, etc. scars, and aging among other factors.

The Windows operating system was installed with the Python programming language, the OpenCV, os, and numpy libraries, and the PyCharm IDE in which the face recognition algorithm was developed using the technique that provides the best result to determine an efficient technique.

For [1], to check the image processing capacity for facial recognition, three techniques were used: Eigenfaces, LBPH (Local Binary Patterns Histograms), and Fisherfaces.

The problem of facial recognition is very complex since a person's face can vary depending on their age, gestures, emotions, half-covered faces, light intensity, etc., this has insisted that the process of recognition does not reach 100% efficiency.

At present many techniques allow these facial recognition processes to be carried out, they try to reach a high percentage of effectiveness, with this work a very acceptable level was reached, through a bibliographic review of databases, feature extraction techniques, and pattern classification algorithms in Python and OpenCV.

Finally, to check if the applied technique meets the expected expectations, different field tests will be carried out with the detection of faces and the comparison of results in the three techniques described.

2 Problem Formulation

Nowadays, the use of facial recognition has been one of the most common problems worldwide due to the availability of software, the cost, and the hardware needed for its implementation in a public or private organization. World powers such as the United States, Russia, China, Japan, Germany, and

North Korea allocate large amounts of economic resources to the implementation of facial recognition tools in companies in these countries so that they have greater security in the authentication of workers.

According to [2], due to the current situation of violence and crime that Ecuador is going through, it is necessary to propose solutions that reduce insecurity rates, a facial recognition platform was implemented, designed to be exposed to the public through services web, the logic deployed was structured to satisfy the requirements in the context of the country, seeking integration with the entities in charge of citizen security by having control of the development of the platform, different techniques and changes to the face detection algorithms were combined. and face recognition.

According to [3], in primitive civilizations people lived in small communities, where they were easily recognized by their limited number of inhabitants; however, due to the rapid population increase and human mobility, identification has become a complicated process, in such a way that innovative societies have found it necessary to implement sophisticated identification recording techniques.

[4], stated, in 1882, the French policeman Alphonse Bertillon (1853-1914) presented the first system of identification of people based on physical characteristics, that is, on biometric features, and called anthropometry and this is considered the first scientific system used by the police to identify criminals, it works by classifying the shape of the nose, face or body of people: The following image shows the different types of noses that were published in Pearson's Magazine. It is presented in Figure 1.

Fig. 1: Image published by Pearson's Magazine [4].

[5], affirms that the concept of facial detection began in the '60s. Between the years 1964 and 1965 Woodrow Wilson, Helen Chan, and Charles Bisson developed the first semi-automatic facial recognition system through the use of the computer, in the 70s Goldstein, Harmon, & Lesk, used 21 physical characteristics among them the color of the hair, the thickness of the lips, etc. To improve facial detection, and to identify these features, a manual sequence had to be followed.

In the early 1990s, [6], using the "eigenfaces technique" (method discovered by Kirby and Sirovich) demonstrated that: "the error was used in the identification of faces in images, this was a finding that allowed perform face detection systems in real-time. That is why the similarity was forced by environmental factors, however, this caused significant interest in the further development of these systems.

[7], stated in 2001, surveillance cameras were used in a Super Bowl game where the procedure was the collection of images through the cameras to later be compared with digitized images of delinquents who were in a database that stored said information.

For [8], Machine Learning is a technique associated with the automatic detection of relevant patterns within a data set. In recent years, it has become a very common tool in practically all tasks that require extracting information from data. From large amounts of data.

According to [9], on a day-to-day basis, we are surrounded by technology based on Machine Learning: email filtering, recommendation systems, facial detection, smartphone speech recognition, weather forecasting, etc., or consulting traffic on the road, it is also used in other fields such as medicine, marketing, logistics or the maintenance of industrial equipment.

[9] state, due to the complexity of all these applications, a human being is not capable of programming a series of specific specifications to carry out said tasks, but rather has to provide the computers themselves with the ability to learn from experience and adapt to new situations.

According to [10], learning refers, as has been seen, to a wide spectrum of situations in which the learner increases his knowledge or his abilities to accomplish a task, learning applies inferences to certain information to build an appropriate representation. Of some relevant aspect of reality or some process, a common metaphor in the area of machine learning within Artificial Intelligence is to consider problem-solving, according to [11], as a type of learning that consists once a type of problem has been solved. A problem in being able to recognize the problematic situation and react using the learned strategy, currently the greatest distinction that can be drawn between an animal and a problem-solving mechanism is that certain animals are capable of improving their performance, in a broad set of tasks, as a result of having solved a certain problem.

For [11], computer vision, also called artificial vision, machine vision, computational vision, image analysis, or scene interpretation, is the process of extracting information from the real world from images using a computer as a tool.

According to [12], face detection is a technique that allows finding the face of one or more people in an image, while ignoring the background of the image or other objects that are present within it.

According to [13] is present in many applications that we use every day, for example on Facebook the images that we upload already detect the face, and what often asks us is to add the name of the person or persons that appear there. , we also have Instagram filters where the face needs to be detected so that the filters that we want to apply are used. It is even present in some banking applications or when we use a smartphone to take a photo and it shows the faces of the people who appear there in a box or circle.

For [12], object detection using cascade classifiers based on de Haar functions is an effective object detection method proposed by Paul Viola and Michael Jones, it is an approach based on machine learning in which a cascade function is trained on many positive and negative images, then used to detect objects in other images. The method used by Haar can be described as a screen of pixels, of different orientations and sizes, separated by rectangles, each rectangle being either positive or negative.

According to [14], facial recognition is an area that is part of pattern recognition, in recent years it has gained great interest, especially due to the wide range of applications it has in different fields such as security, surveillance, cards, smart, among and others.

On the other hand, [15], facial recognition is a part of computer vision. Facial recognition has been used for many decades, mainly by the armed forces. According to [16], it is understood that facial recognition is a part of computer vision, it is necessary to know its definition to have a better understanding of the subject, it is a way to obtain, and process images of the real world to obtain numerical information that can be handled by a computer.

For [7], facial recognition is based on patterns that can be checked if an image contains a face. This is achieved by training the neural network methods with images that contain faces and others that do not show images.

For [17], facial detection systems aim to detect the presence of a person through their facial features in a digital image, the main advantage of these detection systems is that it is not intrusive, so it does not require the collaboration of the user beyond being in front of the camera used in the security system, in addition to the fact that it only requires a single capture device.

For [18], the local binary pattern method was designed for the description of textures. According to [19], the use of local descriptions in some regions of the face provides more information than others, so texture descriptors tend to average the information they describe, which is not convenient when describing faces since maintaining information on spatial relationships is important. For [20], to form the global description, the image of the face is divided into different regions, to which a histogram is applied with which the LBPH operator is obtained, which describes independent information by region, these local descriptions are then concatenated. To build a global description of the face.

[21], Fisherface is a face recognition technique, which takes into account light and facial expressions. This is in charge of classifying and reducing the dimensions of the faces using the FLD (Discriminant Linear Fisher) method. For [21], Fisher's discriminant analysis tries to project the data in such a way that its new dispersion is optimal for classification, while PCA looks for the vectors that best describe the data, LDA (Discriminant Linear Analysis) looks for the vectors that provide better discrimination between classes after the screening.

According to [22], Fisherfaces performs an LDA, where it seeks to take advantage of the available information about the classification of the training images, to find a projection that maximizes the separation between images of different people (or classes) and minimizes the distance between images of the same class, thus concentrating the images, significantly improving the recognition rate. Several techniques are used for facial recognition, but these have shortcomings when implemented in some companies such as response time and analysis of 2D image patterns, generating a loss of time at the time of user authentication and thus damaging the arrest of the faces.

[23], suggests that Python is a high-level language since it contains some implicit data structures such as lists, dictionaries, sets, and tuples, which allow performing some complex tasks in a few lines of code and in a readable way.

The Python Standard Library, [24] states the precise syntax and semantics of the Python language, this library reference manual describes the standard library that ships with Python, and it also describes some optional components that are usually included in Python distributions.

3 Problem Solution

The focus of this research is quantitative, therefore, it was possible to store and analyze the data obtained in a way that facilitates the research, to better understand the object of study on global facial recognition using Python and OpenCV, with this information raised the problem, formulation of the problem, theoretical foundation, processing and discussion of the results.

To determine the sample, three facial recognition techniques were used: Eigenfaces, Fisherfaces, and LBPH with a total of 31 students of different characteristics and ethnicities will be used in this research, students from the Information Systems career were used to conduct research, obtaining data and therefore the results of the research.

This research will handle 2D facial images. Facial images have unique characteristics that differentiate them from one another. A human face is established in two dimensions by taking the plane and coordinates (x, y) in such a way that we can obtain images, with gestures and emotions (happy, sad, angry, etc.). It is presented in Figure 2.

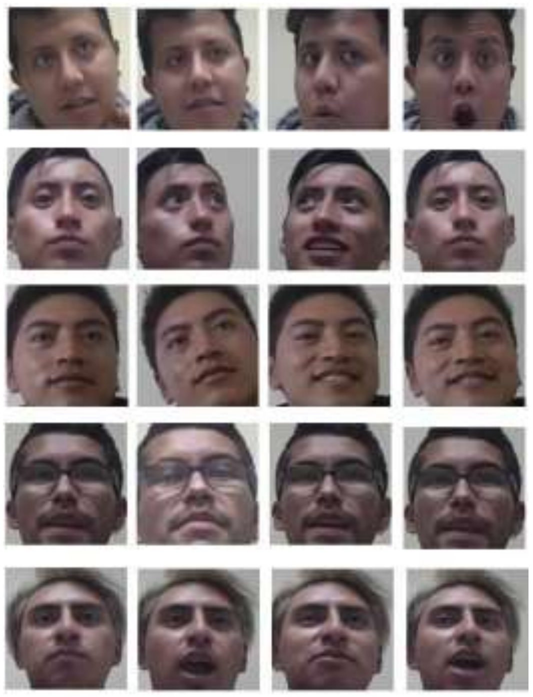


Fig. 2: Emotions and gestures.

As part of the experimental research applied in this research work, for the analysis and processing of the data obtained, algorithms are used to extract pickpockets from images and algorithms for their recognition within the 2D recognition processes.

Face detection is a technique that makes it possible to find the face or faces of numerous people in an image while ignoring the background of the image or other objects that are present in it.

It is present in many applications that we use every day, for example, in the social network Facebook with the images that are uploaded detect the face, and what it often asks us is to add the name of the person or persons that appear there. Also in the social network Instagram, we have filters where the face needs to be detected to be applied.

3.1 Detecting a Face in an Image

It is not an easy task for the computer, so we need it to 'learn', which is why we use Machine Learning or automatic learning for face detection.

Initially, a large number of images are needed to train a classifier, so that it can differentiate between the presence of an object and its non-presence. For example, to perform a face detector, positive images (containing faces) and negative images (images containing no faces) are required.

Then we will proceed to extract features from all images, to then use a Machine Learning approach, and proceed with training. Finally, the classifier can be obtained. It is presented in Figure 3.

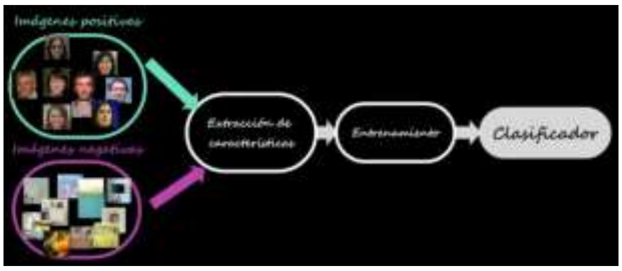


Fig. 3: Process to create a face classifier, [34].

Now that we have this process in mind, let's briefly look at facial detection using the Haar Cascade. This object detector uses the method proposed in the paper “Rapid Object Detection using a Boosted. It is presented in Figure 4.

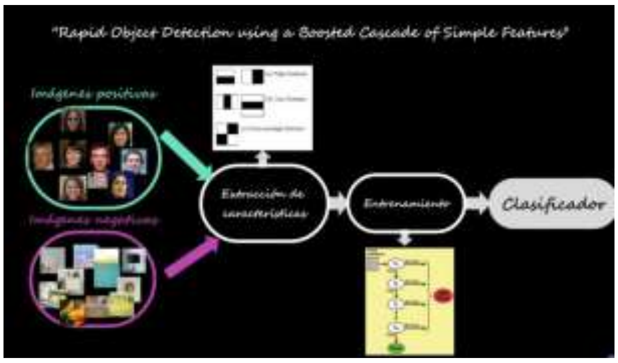

"Rapid Object Detection using a Boosted Cascade of Simple Features"
Imágenes positivas
Imágenes negativas
Extracción de características
Entrenamiento
Clasificador

Fig. 4: Process to create a face classifier using Haar-Cascade, [34].

3.2 Haar Cascades Python – OpenCV

OpenCV offers us pre-trained classifiers not only for people's faces, which we will use in this work but also for eyes, smiles, and whole bodies, among others. You can find these XML files in the folder: OpenCV/data/haarcascades/ or in the OpenCV repository on GitHub.

3.3 DetectMultiScale

To use face detection with haar cascade in OpenCV we are going to need the DetectMultiScale module that will help detect objects according to the classifier used. This will allow us to obtain a delimiting rectangle where the object to be found within an image is located, for this, we must specify some arguments that are detailed below.

3.4 Scale Factor

This parameter specifies how much the image will be reduced. For example, if 1.1 is entered, it means that the image will be reduced by 10%, with 1.3 it will be reduced by 30%, thus creating a pyramid of images. It must be taken into account and that is that, if we give a very high number, some detections are lost. While for very small values such as 1.01 (that is, to reduce the image by 1%), it will take longer to process, since there will be more images to analyze, in addition to the fact that they can increase false positives (which are detections presented as objects or faces, but in reality, are not). It is presented in Figure 5.

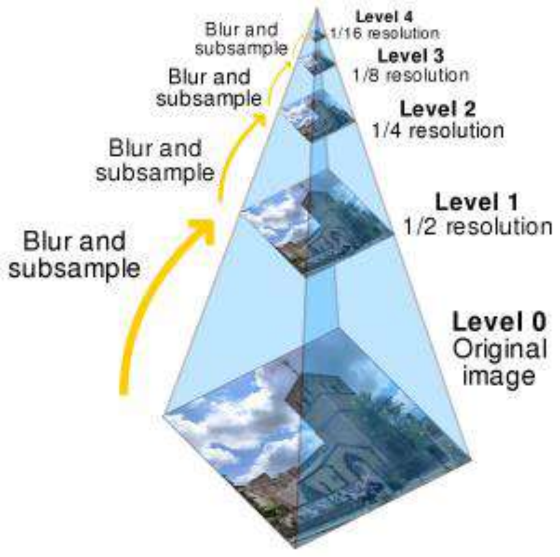


Fig. 5: Example of an image pyramid, [24].

An image pyramid is made due to the size of the faces in the image, some may occupy more or less area than others so to try to detect all in their different sizes, the image pyramid is applied.

3.5 MinNeighbors

This parameter specifies how many neighbors each candidate rectangle must have to retain it. We have a small window that will go through an image looking for faces, so you may find that at the end of the whole process, it has identified several faces (Figure 7), but many of them may correspond to the same person. So, this parameter relates to all those delimiting rectangles of the same face. Therefore, minNeighbors specifies the minimum number of bounding boxes, or neighbors, that a face must have to be detected as such. It is presented in Figure 6.

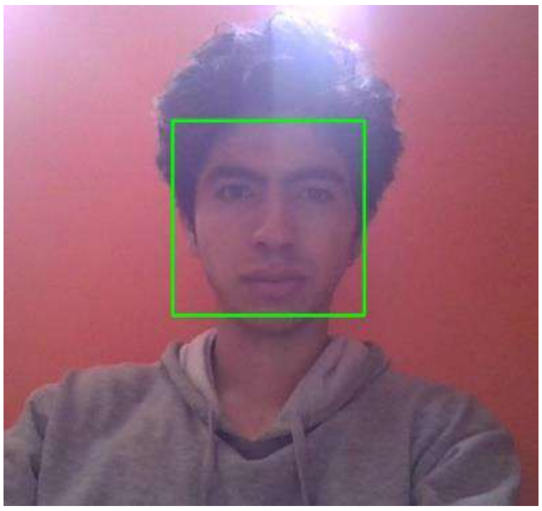


Fig. 6: Face detection.

Now, we must take into account that the higher the value that we set, the fewer faces will be detected, while if a very low value is given, false positives can occur.

3.6. Detection of Faces on an Image

To begin with, the XML file containing the frontal face classifier (haarcascade_frontalface_default.xml) is saved in the same folder where the script is stored. We are going to detect the faces that appear in the following image. The process is presented in Figure 7, Figure 8, and Figure 9.

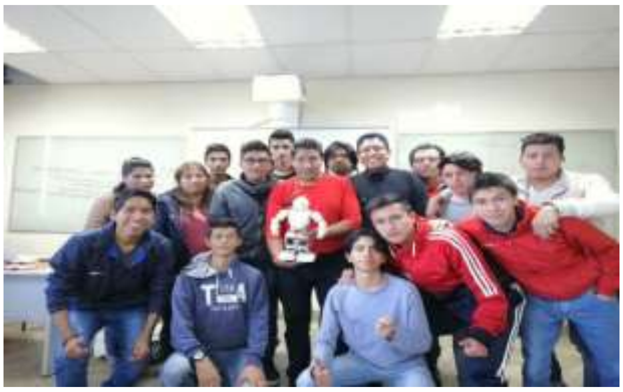
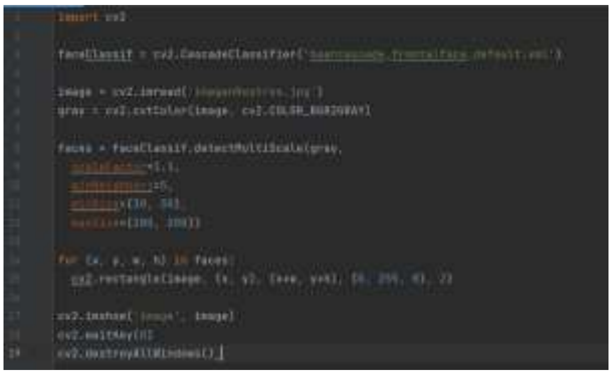


Fig. 7: Input image for face detection.

The programming given for face detection will be as follows:



```
import cv2

faceClassif = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

image = cv2.imread('imagenMuestras.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

faces = faceClassif.detectMultiScale(gray,
  scaleFactor=1.1,
  minNeighbors=5,
  minSize=(30, 30),
  maxSize=(200, 200))

for (x, y, w, h) in faces:
  cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)

cv2.imshow('image', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Fig. 8: Code to detect the face.

Line 1: Import OpenCV.
Line 3: Load the classifier with an XML extension with the help of cv2.CascadeClassifier, inside quotes specify the name and extension of the file.
Lines 5 and 6: Read the image where the faces will be detected and then proceed to transform it to grayscale.
Line 8: Load the classifier in line 3 it is necessary to apply it to the image, for this purpose detectMultiScale is used, which must be followed by the variable with which the load of the classifier was assigned and this is where the recently raised arguments are: image, scaleFactor, minNeighbors, minSize, and maxSize.

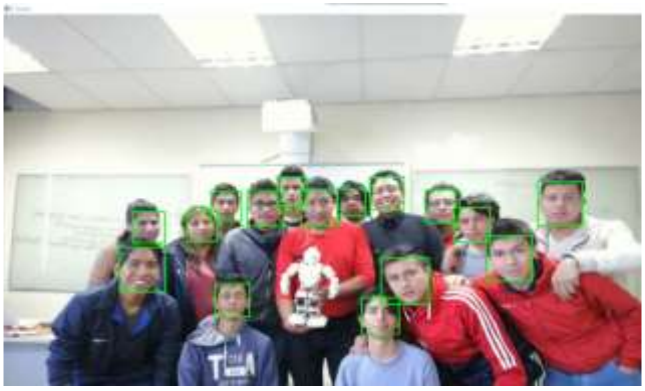


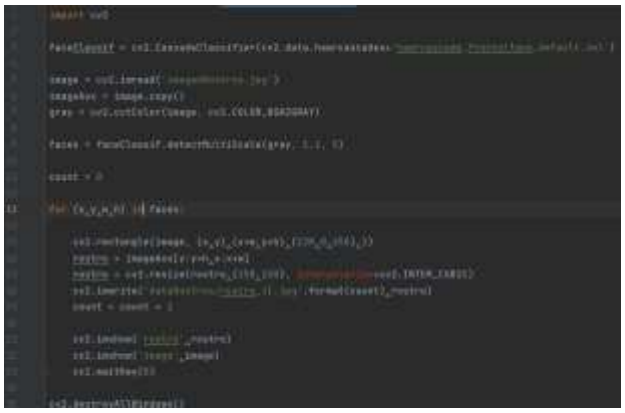
Fig. 9. Image with face stop.

3.7 Storing Faces

We proceed to reuse the face detection code in an image with the use of haarcascades, we will use the same procedure adding new lines of code to save the faces. It is presented in Figure 10.

Fig. 10: Input image to store faces.

The code to detect and store the faces of the image will be the following (Figure 11).



```
import cv2

faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')

image = cv2.imread('imagenMuestras.jpg')
imageAux = image.copy()
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

faces = faceClassif.detectMultiScale(gray, 1.1, 5)

count = 0

for (x,y,w,h) in faces:

    cv2.rectangle(image, (x,y),(x+w,y+h),(128,0,255),2)
    rostro = imageAux[y:y+h,x:x+w]
    rostro = cv2.resize(rostro,(150,150), interpolation=cv2.INTER_CUBIC)
    cv2.imwrite('rostro_{}.jpg'.format(count),rostro)
    count = count + 1

    cv2.imshow('rostro',rostro)
    cv2.imshow('image',image)
    cv2.waitKey(0)

cv2.destroyAllWindows()
```

Fig. 11: Code to store faces from an image.

Line 1: We import OpenCV.
Line 3: Load the face classifier.
Line 5: We proceed to read the input image that would correspond to the image in Figure 12.
Line 6: Create a copy of the input image, this will remain intact, so it will serve to govern the faces found.
Line 7: Transform the input image from BGR to grayscale.
Line 9: The detection of faces present in the image is performed, according to the input parameters of the detectMultiScale function.
Line 11: Start a counter count = 0, which helps us to count each of the detected faces.
Line 13: Extract the coordinates, width, and height of each of the detected faces.
Line 15: Display the detected faces in a pink rectangle.
Line 16 and 17: In line 16 we proceed to crop the faces of the imageAux image with the help of the x, y width, and height coordinates that we obtained from line 13. In line 17 the faces are resized to 150 pixels wide and high.
Line 18: Each of the images will have a different number thanks to the count counter.
Lines 21 to 25: These lines are for displaying the face and detections.

Running the code returns the following as presented in Figure 12.

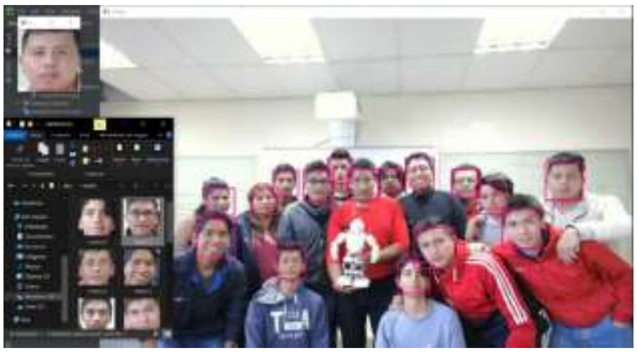
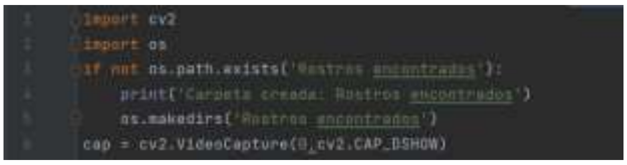


Fig. 12: Storing faces from an image.

As any key is pressed, a face is surrounded and stored in the folder where the script is located. It should be taken into consideration that all the faces have the same width and height since we resized them in the code.

3.8 Store Faces Present in a Video

For the extraction and storage of faces through the use of videos, it can be done in a streaming video or read a video. For example, we will do it in a streaming video as presented in Figure 13.



```
1   import cv2
2   import os
3   if not os.path.exists('Rostros encontrados'):
4       print('Carpeta creada: Rostros encontrados')
5       os.makedirs('Rostros encontrados')
6   cap = cv2.VideoCapture(0,cv2.CAP_DSHOW)
```

Fig. 13: Code to store faces from a video

Line 6: Specify that a video will be streamed, however, a video can be read. Cv2.CAP_DSHOW is used so that at the time of visualization they completely occupy the window.

Running the code returns the following. They are presented in Figure 14, and Figure 15.

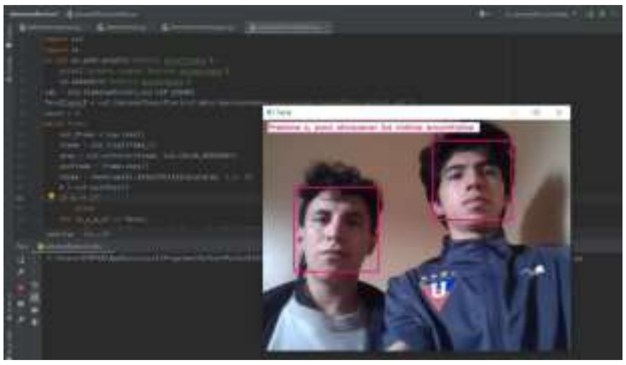


Fig. 14: Storing faces.

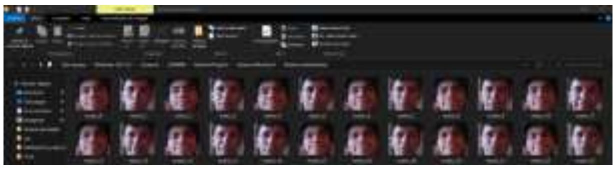


Fig. 15: Stored Faces

This procedure will be used later for facial recognition since we need a large number of images of people to be recognized by the algorithm.

To carry out facial recognition, in the first place it is necessary to collect data, that is, the faces of the people that you want to recognize, then proceed to train the classifier, to finally test it. For all this process it will be necessary to use face detection with haarcascades.

3.9 Creation of the Database

To perform facial recognition, the faces of the people who want to recognize themselves are necessary. These faces must denote a variety of expressions such as: happiness, sadness, boredom, and surprise, among others. Another aspect that these images must have is the variation in light conditions, whether people wear glasses or not, even whether they close their eyes or wink.

It is recommended that these images be collected in the setting or environment where facial

recognition is to be applied. All this variety of images obtained from the faces will contribute to the performance of the algorithms used in this work.
In the code used for the demo, 130 faces from a streaming video will be stored automatically. It is presented in Figure 16.

Fig. 16: Code to store faces in the database.

Line 1 to 3: Import OpenCV, os and imutils.
Lines 5 to 7: In these lines, a folder is created with the name of the person you want to recognize, this will be created inside the Data folder that you had previously created manually. So in line 5 in personName, the name of the person is assigned, and in line 6 dataPath is assigned the location of the directory where each folder will be created with the name of each person to be recognized. Finally, personPath will be the full path.
Lines 9 to 11: With the information from the last lines, the directory will be created with the name of the person to be recognized inside the Data folder. In line 5, for example, you can see that a folder called 'Tutillo_9no_Sis1' will be created as presented in Figure 17.

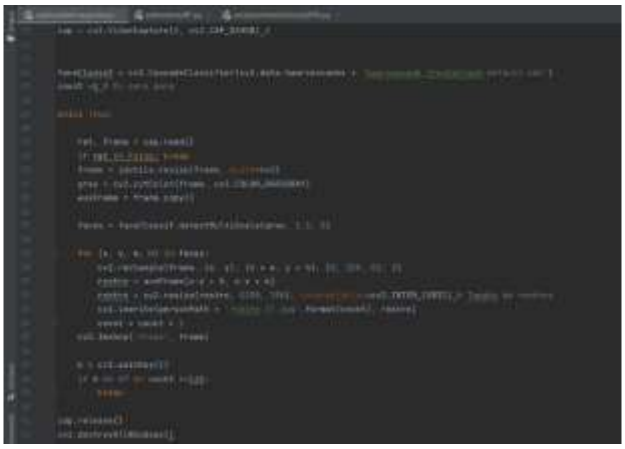
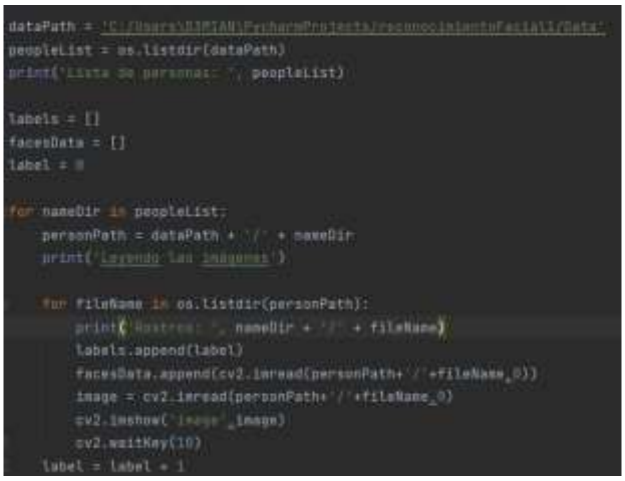


Fig. 17: Code to establish the number of images.

Line 23: Resize with imutils.resize, this is done to resize the input video frames.
Line 32: The images corresponding to the faces are resized so that they all have the same size. 150 pixels have been set.
Line 38: In this line, I have added the condition of count >= 130 so that the storage process ends at 130 stored faces. Running the code returns the following. It is presented in Figure 18.

Fig. 18: Database created from the established code.

3.10 Preparing Data for Training
To proceed with the training, it is necessary to have each of the images with their respective label associated with the person to whom the faces belong. For example, when we read the folder 'Abata, all those images will be assigned a label 0, then all the images of the faces of 'Caizaluisa' will be assigned 1, 'Canizares' will be assigned 2, and so on. With each of these labels, the computer will know that the images correspond to different people. In another class that we call 'RFtrainer.py' this process is prepared, that is, the images and labels to train them used: Eigenfaces, fisherfaces, and Local Binary Patterns Histograms. It is presented in Figure 19.



```
dataPath = 'C:/Users/DJMIAN/PycharmProjects/reconocimientoFacial1/Data'
peopleList = os.listdir(dataPath)
print('Lista de personas: ', peopleList)

labels = []
facesData = []
label = 0

for nameDir in peopleList:
    personPath = dataPath + '/' + nameDir
    print('Leyendo las imagenes')

    for fileName in os.listdir(personPath):
        print('Rostros: ', nameDir + '/' + fileName)
        labels.append(label)
        facesData.append(cv2.imread(personPath+'/'+fileName,0))
        image = cv2.imread(personPath+'/'+fileName,0)
        cv2.imshow('image',image)
        cv2.waitKey(10)
    label = label + 1
```

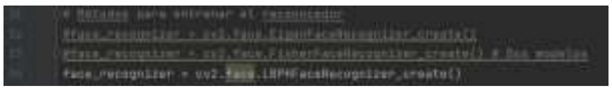
Fig. 19: Code to prepare the images for training.

Line 5 to 7: First, specify the path of the 'Data' folder that was created manually before and within it will be each folder with the name of the people you want to identify. Line 6 lists all the names of the folders stored in 'Data'. Line 7 prints the list obtained.

Line 9: Labels are declared in this the labels corresponding to each image will be stored according to the person.
Line 10: faces Data is declared where each of the face images will be stored.
Line 11: A label counter is set to 0, so as it finishes reading the images of a person, it changes to another value. This will help the classifier understand what it takes from different people.
Line 13 to 15: Read each folder inside 'Data'. The person Path sets the path to each person's folder. While in line 15 we print the message 'Reading the images...' so that when it is executed it can know in which part of the process it is.
Lines 17 and 18: All the images corresponding to each face are read. On line 18 the name of the folder and the image are printed.
Line 19: In labels, the labels of each image are added.
Line 20: Each image (face) is added to the faces Data array.
Line 24: Every time the faces and labels of a folder are finished storing, the label will be increased by 1.

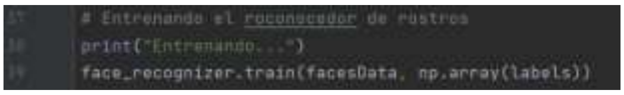
3.11 Training

Starting from facial recognition using OpenCV, we move on to code. They are presented in Figure 20, and Figure 21.



```
30  # Métodos para entrenar el reconocedor
31  #face_recognizer = cv2.face.EigenFaceRecognizer_create()
32  #face_recognizer = cv2.face.FisherFaceRecognizer_create() # Dos modelos
33  face_recognizer = cv2.face.LBPHFaceRecognizer_create()
```

Fig. 20: Methods for training the knower.



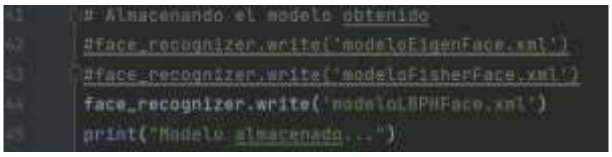
```
37  # Entrenando el reconocedor de rostros
38  print("Entrenando...")
39  face_recognizer.train(facesData, np.array(labels))
```

Fig. 21: Training the face recognizer.

Line 39: To train the face recognizer, face_recognizer.train is needed, where face_recognizer will be the variable where the method was assigned (any of lines 31, 32, or 33). Within the parentheses, you have to specify the array where the faces or training images are contained, while the second parameter corresponds to the labels. It is necessary that this be a numpy array so I have put np.array(labels).

3.12 Save the Obtained Model

Once the model that helps us to recognize faces has been trained, we proceed to save it and read it in another class. To store the model, write is used, within the parentheses the name to be assigned is specified, together with the XML or YAML extension. It is presented in Figure 22.



```
# Almacenando el modelo obtenido
#face_recognizer.write('modeloEigenFace.xml')
#face_recognizer.write('modeloFisherFace.xml')
face_recognizer.write('modeloLBPHFace.xml')
print("Modelo almacenado...")
```

Fig. 22: Save obtained model

Running the code returns the following. They are presented in Figure 23, and Figure 24.

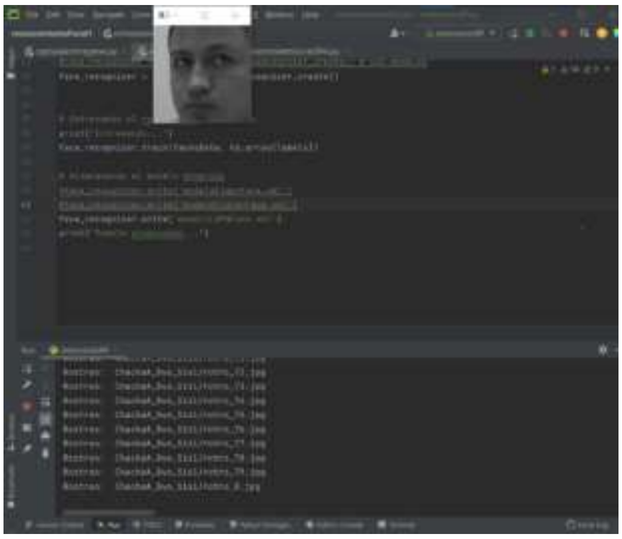


Fig. 23: Training the face recognizer.

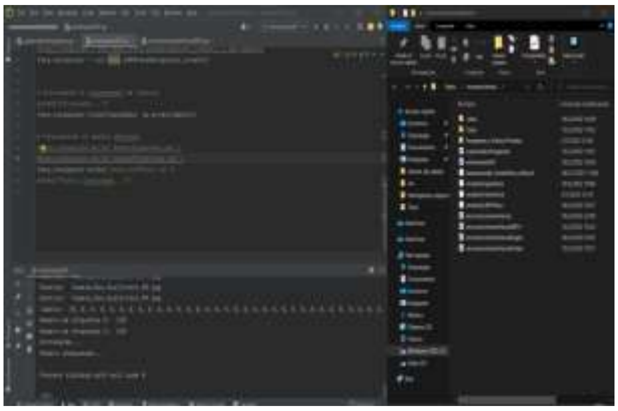


Fig. 24: Saving the obtained model.

4 Results

4.1 Efficiency Comparison between Facial Recognition Techniques

The success of the EIGENFACES technique corresponds to 90% effectiveness and 10% inefficiency, identifying 28 hits and 3 refusals out of a total of 31 students with 4030 photographs of which 3100 were used for training and 930 for the same tests. They have features like: gestures, light variations, and props

The success of the FISHERFACES technique corresponds to 100% effectiveness, identifying 31 correct answers out of a total of 31 students with 4030 photographs of which 3100 were used for training and 930 for tests, which have characteristics such as: gestures, variations of light and accessories. The success of the LBPH technique corresponds to 100% effectiveness, identifying 31 successes out of a total of 31 students with 4030 photographs of which 3100 were used for training and 930 for the tests, which have characteristics such as: gestures, variations of light, and accessories. It is presented in Table 1.

Table 1. Times and hits obtained in the test phase

TECHNIQUES						
	EIGENFACES		FISHERFACES		LBPH	
Student No.	Time (Seconds)	Recognition	Time (Seconds)	Recognition	Time (Seconds)	Recognition
1	4,16	Si	0,77	Si	2,27	Si
2	0	No	1,17	Si	1,96	Si
3	15,73	Si	1,37	Si	0,6	Si
4	10,75	Si	0,34	Si	1,52	Si
5	15,1	Si	1,94	Si	0,79	Si
6	0	No	3,34	Si	1,17	Si
7	3,77	Si	0,8	Si	1,95	Si
8	7,12	Si	1,32	Si	2,07	Si
9	6,57	Si	2,75	Si	1,22	Si
10	4,85	Si	2,72	Si	1,43	Si
11	8,16	Si	2,66	Si	0,27	Si
12	5,78	Si	1,28	Si	1,11	Si
13	6,44	Si	1,94	Si	0,27	Si
14	4,46	Si	1,32	Si	1,15	Si
15	0,87	Si	2,11	Si	1,56	Si
16	5,92	Si	3,31	Si	1,23	Si
17	7,42	Si	4,15	Si	0,44	Si
18	0	No	1,65	Si	1,96	Si
19	7,3	Si	8,12	Si	1,9	Si
20	12,78	Si	1,38	Si	3,68	Si
21	19,86	Si	1,71	Si	4,02	Si
22	18,66	Si	3,26	Si	1,07	Si
23	23,38	Si	1,82	Si	3,28	Si
24	5,53	Si	0,68	Si	1,05	Si
25	18,91	Si	1,64	Si	3,45	Si
26	4,67	Si	3,4	Si	2,9	Si
27	16,49	Si	3,11	Si	0,27	Si
28	1,79	Si	0,95	Si	0,84	Si
29	5,41	Si	0,58	Si	0,27	Si
30	15,95	Si	1,62	Si	3,88	Si
31	6,61	Si	0,87	Si	1,41	Si

In summary, according to Table 1, we worked with 4030 student photographs, 130 photographs per student. Of which 100 photographs were taken for training and 30 for tests. That is 3100 images for training and 930 for tests.

As a result, the FISHERFACES and LBPH techniques are more precise in terms of success in recognition. Based on the time spent and success, it is concluded that the LBPH is the most efficient technique.

4.2 Technique Recognition Average Time

Once the success time of each student for each technique was obtained (see Table 2), a comparison was made between the times of the techniques, in order to identify the efficiency in the response time. The LBPH technique with an identification average of 1.64 seconds.

Table 2. Statistics on the time spent in the facial recognition process.

AVERAGE TIME IN THE RECOGNITION PROCESS	
TECHNIQUES	AVERAGE TIME (seconds)
EIGENFACES	8,53
FISHERFACES	2,06
LBPH	1,64

5 Conclusion

The bibliographic review of several books, scientific articles, and web pages related to the 2D image database, feature extraction, and pattern recognition algorithms was of vital importance for the development of the theoretical foundation and the determination of technological tools. With the design of the prototype for global facial recognition using Python and OpenCV, he allowed the comparison of the techniques used to determine which technique is the most efficient. Through the endorsement of expert judgment, it was determined that the prototype for facial recognition complies with the correct operation for face detection and in an analysis of the techniques used.

The facial recognition process still has a lot to be analyzed. Since several factors affect the recognition of a face (age, lighting, gestures, scars, distance, and angle to the camera, etc.).

Currently, we have programming languages such as Python, which are incorporated into their libraries some extraction and classification techniques such as those studied in this work. However, in the future, it would be interesting to carry out facial recognition studies in 3D images. At least the problem of lighting and angle to the camera would be solved.

References:

[1] (*text in Spanish*) C. E. Franco, C. T. Ospina, E. S. Cuevas, and D. V. Capacho, “Reconocimiento Facial Basado En Eigenfaces, Lbhp Y Fisherfaces En La Beagleboard-Xm,” *Rev. Colomb. Tecnol. Av.*, vol. 2, no. 26, 2017, doi: 10.24054/16927257.v26.n26.2015.2387.

[2] (*text in Spanish*) J. González Astudillo and M. G. Zhindón Mora, “Plataforma de servicios de reconocimiento facial para detección de prófugos de la justicia en Ecuador,” *Rev. Cienc. e Investig.*, vol. 5, pp. 31–41, 2020.

[3] (*text in Spanish*) M. Kuliah and M. Kuliah, “Herramienta De Reconocimiento Facial Con

Técnica De Visión Computacional 2d," no. April, pp. 33–35, 2019.

[4] F. Serratosa, "Biometria_ES_(Modulo_1)," p. 50, 2008.

[5] (*text in Spanish*) J. J. I. Casanova, "Reconocimiento Facial En Ambientes No Autor : Asesor : Línea de Investigación : Infraestructura , Tecnología y Medio Ambiente," 2020.

[6] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," *Proceedings. 1991 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 586–591, doi: 10.1109/CVPR.1991.139758.

[7] (*text in Spanish*) M. Centeno, "Tecnicas De Biometria Basadas En Patrones Faciales Del Ser Humano," *Экономика Региона*, p. 32, 2012.

[8] A. Maisueche Cuadrado, "Utilización Del Machine Learning En La Industria 4.0," pp. 1–118, 2019, [Online]. Available: https://core.ac.uk/download/pdf/228074134.pdf#page=37&zoom=100,90,94.

[9] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*, vol. 9781107057. 2013.

[10] (*text in Spanish*) A. Moreno *et al.*, *Aprendizaje automático*. 1994.

[11] D. Angluin, "Computational learning theory: Survey and selected bibliography," *Proc. Annu. ACM Symp. Theory Comput.*, vol. Part F1297, pp. 351–369, 1992, doi: 10.1145/129712.129746.

[12] (*text in Spanish*) P. F. De Máster, "Utilización de métodos de visión artificial para PC como apoyo en la automoción.," 2015.

[13] "OpenCV: Clasificador en cascada." https://docs.OpenCV.org/3.4/db/d28/tutorial_cascade_classifier.html (accessed Jan. 15, 2022).

[14] (*text in Spanish*) J. Ibarra-Estévez and K. Paredes, "Redes neuronales artificiales para el control de acceso basado en reconocimiento facial," *Revistapuce*, 2018, doi: 10.26807/revpuce.v0i106.140.

[15] S. Mullainathan and J. Spiess, "Machine learning: An applied econometric approach," *J. Econ. Perspect.*, vol. 31, no. 2, pp. 87–106, 2017, doi: 10.1257/jep.31.2.87.

[16] (*text in Spanish*) A. M. Llapapasca Montes and M. A. Ochoa Zevallos, "Creación de una librería de software de reconocimiento facial enfocado a la identificación de trabajadores de una empresa Ingeniería de Software," *Univ. Tecnológica del Perú*, 2019, [Online]. Available: http://repositorio.utp.edu.pe/handle/UTP/2278.

[17] (*text in Spanish*) T. D. F. De Grado, "Vida En El Reconocimiento Facial Jaime Varela de la Escalera de Miguel," 2019.

[18] R. Gottumukkal and V. K. Asari, "System level design of real time face recognition architecture based on composite PCA," *Proc. IEEE Gt. Lakes Symp. VLSI*, pp. 157–160, 2003, doi: 10.1145/764808.764849.

[19] (*text in Spanish*) J. D. Alvarado and J. Fernández, "Análisis de textura en imágenes a escala de grises, utilizando patrones locales binarios (LBP)," 2012.

[20] (*text in Spanish*) P. Álvarez, "Prototipo de sistema piloto para control de acceso basado en reconocimiento de rostros," *Fac. Ing. Univ. Mil. Nueva Granada*, p. 72, 2013.

[21] A. M. Martinez and A. C. Kak, "PCA versus LDA," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 2, pp. 228–233, Feb. 2001, doi: 10.1109/34.908974.

[22] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 1064, no. 7, pp. 45–58, 1996, doi: 10.1007/bfb0015522.

[23] I. Challenger, Y. Díaz, and R. B. García, "El lenguaje de programación Python/The programming language Python," *Redalyc*, vol. XX, pp. 1–13, 2014, [Online]. Available: https://www.redalyc.org/pdf/1815/181531232001.pdf.

[24] (*text in Spanish*) La Biblioteca Estándar de Python — documentación de Python - 3.10.2. https://docs.python.org/es/3/library/index.html (accessed Jan. 15, 2022).

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

We confirm that all Authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflicts of Interest

Please declare anything related to this study.