2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO) | 978-1-6654-7433-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICRITO56286.2022.9964836

Face Recognition using Open Source Computer Vision Library (OpenCV) with Python

Gurpreet Singh
Department of CSE
Chandigarh University
Mohali, India
aiet.cse.gurpreet@gmail.com

Ishika Gupta
Research scholar
Chandigarh University
Mohali, India
ishika01301@gmail.com

Jaspreet Singh
Department of CSE
Chandigarh University
Mohali, India
cec.jaspreet@gmail.com

Navneet Kaur
Department of CSE
Chandigarh University
Mohali, India
navneet.e2000@cumail.in

***Abstract*—Within a human's lifetime, faces are the visually embellished images that appear most frequently. Facial Recognition is the ability to recognize and discover someone primarily based totally on their facial features. Because the face is multidimensional, it necessitates several mathematical calculations. Thus, this work is aimed at developing an improved face recognition system that will serve in numerous application areas. Many approaches have already been developed, but they have low recognition capabilities and a high false alarm rate. As a result, there is a need for face recognition system that is more accurate and has a shorter recognition time. Due to rising security concerns and the quick development of mobile devices, face detection has recently been a hot research area. This paper focuses on the implementation of a face recognition system for human identification based on the open-source computer vision library (OpenCV) with python.At last the current application areas related to face recognition system is represented.**

***Index Terms*—Face recognition using Open CV, OpenCV with Python, Computer Vision Library with Python.**

I. Introduction

Face identification has been very simple and reliable since 2002, thanks to Intel for providing an open-source framework OpenCV [1]. This includes face detection, which works in 91-96 percent of lucid images of some person looking directly towards the camera. Facial recognition of a person from an angle, on the other hand, is often more challenging, necessitating 3D Head Pose Estimation in some circumstances. A lack of proper image brightness, or higher contrast in shadows on the face, if the image is grainy, or if an individual wears glasses, can all have a significant impact and enhance the difficulty of distinguishing a face. Face recognition, on the other hand, is much less genuine than face detection, with a popular accuracy of 35-75%. Detection of faces and calculation of directions of faces are crucial for facial recognition. When we use surveillance cameras for non-public identification, for example, it's miles essential to apprehend a face whose size, position, and mindset are not known. An estimate of face direction is useful for correct facial recognition after the detection of a face because it allows us to choose the face picture with the most desirable direction among the images of the face captured through several cameras. Facial recognition has been a popular study topic since 1990, however, it's far nonetheless an extended manner from being a dependable technique of consumer authentication. Each year, a greater number of procedures are developed [2].Facial recognition is one of the most commonly used biometrics authentication methods. It is the most enthralling and successful use of Pattern Recognition and Image Analysis. For the face detection approach, the input image is usually rescaled. The main tasks of a face recognition system are verification and identification. The term verification refers to a 1 : 1 match between photographs of a person's face and images of a template face whose identification is being claimed.Identification is a 1 : N problem in which the image of the query face is compared to all template images in a face database [3] [4].
The procedure involved in face recognition and verification is depicted through a flowchart below:

II. OpenCV Library

Gary Bradski founded OpenCV at Intel in 1999 to hurry up studies and business programs of computer vision around the world whilst additionally driving call for extra effective computer systems for Intel.Vadim Pisarevsky has joined Gary to steer Intel's Russian OpenCV software program crew. Since then, the OpenCV crew has moved directly to different groups and educational initiatives. Several contributors of the preliminary group went directly to work in robots and subsequently ended up at Willow Garage [5].Willow Garage realized the requirement to quickly enhance robotic perception skills in a manner that benefits the whole academic and business community in 2008 promoting OpenCV, Gary and Vadim took the charge once more. Intel's open-source library for computer vision can make programming of computer vision a lot easier. It comes with ready-to-use advanced features including facial detection, tracking of face, facial recognition, Kalman filtering, and a range of algorithms of AI (artificial intelligence). It also includes several fundamental computer-vision techniques in its low-level APIs [6].

It is a sizable open-source library for computer vision, machine learning, and image processing. Python, C++, and

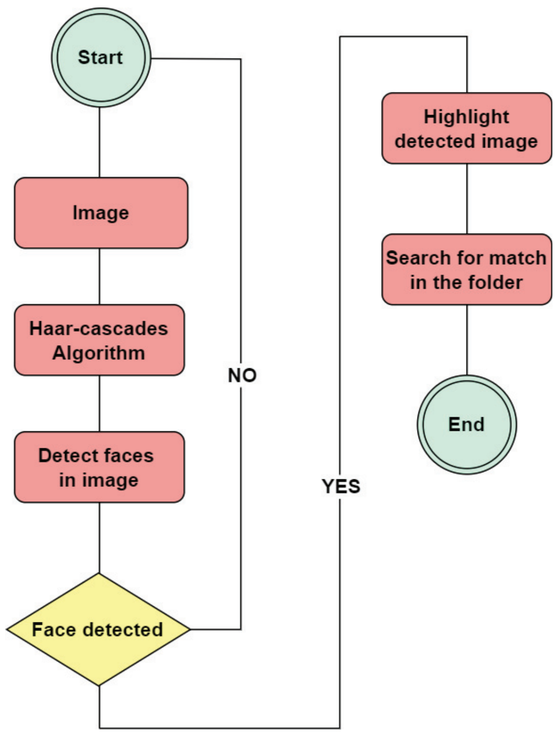


Fig. 1. Flow of the Face Recognition Mechanism

Java are just a few of the many programming languages that OpenCV is compatible with. To identify objects, faces, and even human handwriting, it will analyse images and movies. OpenCV performs well when combined with a variety of other libraries, such as the high-performance library for turning machines Numpy; all operations that can be carried out in Numpy can therefore be merged with OpenCV .It is written based on C++ and has a C++ interface as its main interface, but it also has a less robust but still detailed older Language training [7].

Benefits of OpenCV [5]-

1) It is an open-source library that is free to use.
2) Because OpenCV is written in C/C++, it is faster than others.
3) OpenCV performs better with less system RAM.
4) It is compatible with the most of the operating systems, including Linux, Windows and macOS.

III. PHASES IN FACE RECOGNITION-

1) Face Detection:
   It is critical to detect facial features in an image to recognize a face. We must additionally add a few random photographs to the collection during this step.
2) Preparing and training the dataset for analysis:
   In the second stage, we'll use OpenCV Recognizer to train the dataset. The OpenCV function can be used to fulfill this. As a result, the training Data.yml file gets saved in the trainer directory.
3) Face Recognition:
   Finally, we'll capture a completely new face by the camera, and if the face of the person has been captured and trained previously, the recognizer makes a prediction, returning its index and id, indicating the confidence of the recognizer with this match.cv2recognizer.
   Predict() chooses a captured portion of the face that is to be studied as a parameter and returns its matched Image. It displays the Image Id as well as the recognizer's level of confidence in this match. It's worth noting that if it's an ideal match, the arrogance index will return zero. It forecasts a Face in the last step. We'll also show the name over the picture, along with its likely id and the percentage of the time that it's a correct match (probability = 100 confidence index). If this is not the case, an unknown label is applied to the face [8]–[10]
   The different phases of face recognition can also be depicted through the image below:

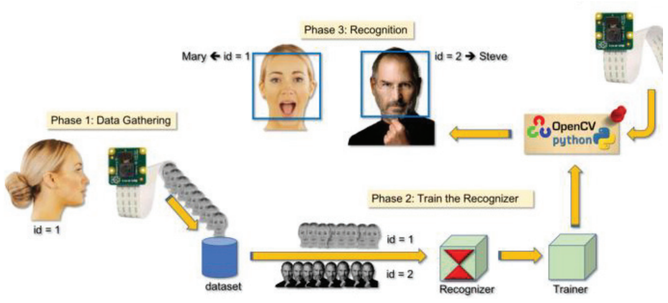


Fig. 2. Different phases under the Face Recognition Mechanism[12]

IV. IMPLEMENTATION OF FACE RECOGNITION SYSTEM USING OPENCV AND PYTHON

1) **Face Detection:**
   There are many APIs built to execute the process of face detection. One such API is ML Kit's face detection API. Face detection gives the records required for duties consisting of improving selfies and snapshots and developing avatars from a user's photo. We can make use of ML Kit in apps like video chat or video games that respond to the player's expressions due to the fact it could come across faces in real-time. The key features of this API are as follows:

   - Locate and identify the features of the face. Get the coordinates of every face detected's eyes, nose, cheeks, mouth and ears.
   - Recognize the contours of your face. Obtain the outlines of the faces you've detected, including their lips, brows, eyes, and nose.
   - Identify different types of facial expressions. Determine whether or not a person is smiling by looking at their eyes.
   - Faces are tracked throughout video frames. For each unique face spotted, obtain an identification.

Because the recognizer is persistent between acknowledgments, we can manipulate a picture of a specific individual in a video stream.

- Real-time video frame processing. Detection of the face is done on the system, and it's fast enough for real-time applications like video processing.

We'll be operating on facial detection here the approach calls for a massive range of tremendous photographs (photographs of faces) and poor photographs (photographs without faces)[1][5].

Code:

For detecting a face using OpenCV and Python,there

```
import numpy as np
import cv2faceCascade =
cv2.CascadeClassifier(&#39;
Cascades/haarcascade_frontalface_default.
xml&#39;)
cap = cv2.VideoCapture(0)
cap.set(3,640) # set Width
cap.set(4,480) # set Heightwhile True:
ret, img = cap.read()
img = cv2.flip(img, -1)
gray = cv2.cvtColor(img,
    cv2.COLOR_BGR2GRAY)
faces = faceCascade.detectMultiScale(
gray,
scaleFactor=1.2,
minNeighbors=5,
minSize=(20, 20)
)
for (x,y,w,h) in faces:
cv2.rectangle(img,(x,y),(x+w,y+h),
(255,0,0),2)
roi_gray = gray[y:y+h, x:x+w]
roi_color = img[y:y+h, x:x+w]
cv2.imshow(&#39;video&#39;,img)
k = cv2.waitKey(30) &amp; 0xff
if k == 27: # press &#39;ESC&#39; to
    quit
breakcap.release()
cv2.destroyAllWindows()
```

is a need a few lines of code. The code written above:

- The image in graysacle input is called gray.
- The scaleFactor term controls how much the size of the image shrinks at each image scale. It is employed in the construction of the scale pyramid.
- minNeighbors is a parameter that determines the minimal number of neighbours that each candidate rectangle must have in order to be kept. When the number is larger, the number of false positives is lowered.
- minSize is the smallest rectangle that can be considered a face.

The result obtained is as shown in Figure 3:

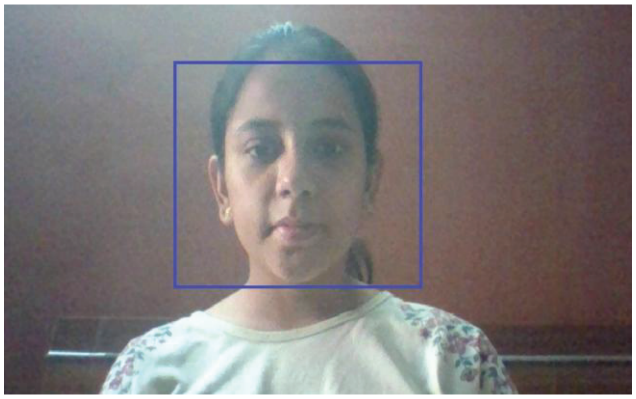


Fig. 3. The result obtained for detecting a face using OpenCV and Python

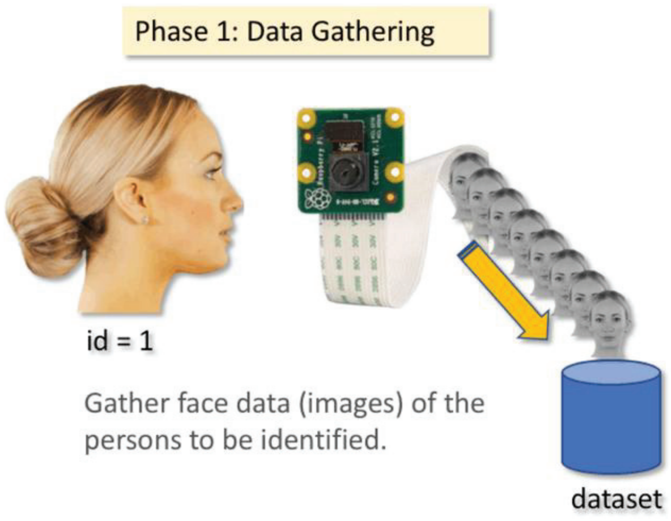


Fig. 4. Data Gathering Phase of the Face Recognition Mechanism[12].

2) **Gathering Data and Training the Dataset:** Starting with the last phase (Face Detection), we'll simply establish a dataset in which we'll keep a collection of grayscale photographs for each id, along with the portion of each photo that was used for face detection.

First we create a directory where we will work on our project, such as Facial Recognition Project:

mkdir Facial Recognition Project

We must have stored the Facial Classifier in this directory, in addition to the three Python scripts that we will be writing for our project.

Next, we create a subdirectory called dataset where all the facial samples will be kept:

mkdir dataset

Code: This code is pretty similar to the facial detection algorithm we examined earlier. An input command was introduced in order to collect a user id, which must be an integer number.

face_id = input('\n enter user id end press ==> ')

We should save each one of the captured frames as a single file in the dataset directory:

cv2.imwrite("dataset/User." + str (face_id) + '.' + str (count) + ".jpg", gray[y: y+h, x: x+w])

You must have imported the library os to save the above file. The following structure will be used to name each file:

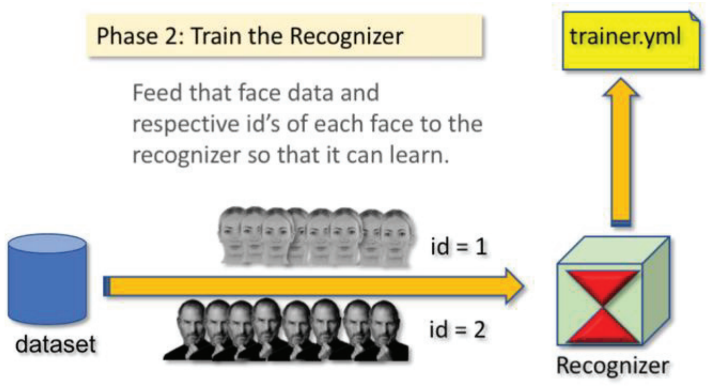
User.face_id.count.jpg

The number of samples is used to break the loop where face samples are taken. Run the Python script to get a

```
import cv2
import oscam = cv2.VideoCapture(0)
cam.set(3, 640) # set video width
cam.set(4, 480) # set video height
face_detector = cv2.CascadeClassifier
('haarcascade_frontalface_default.xml')
# For each person, enter one numeric
      face id
face_id = input('\n enter user id end
      press <return> ==> ')
print("\n [INFO] Initializing face
      capture. Look the camera and wait
      ...")# Initialize individual
      sampling face count count = 0
for (x,y,w,h) in faces:
cv2.rectangle(img, (x,y), (x+w,y+h),
      (255,0,0), 2)
count += 1
# Save the captured image into the
      datasets folder
cv2.imwrite("dataset/User." +
      str(face_id) + '.' +
str(count) + ".jpg", gray[y:y+h,x:x+w])
cv2.imshow('image', img)
k = cv2.waitKey(100) & 0xff # Press
      'ESC' for exiting video
if k == 27:
break
elif count >= 30: # Take 30 face
      sample and stop video
break# Do a bit of cleanup
print("\n [INFO] Exiting Program and
      cleanup stuff")
cam.release()
cv2.destroyAllWindows()
```

few unique identifiers. Every time we wish to include a new user, we must run the script (or change photos for one that exists already).

Now, we'll train the OpenCV Recognizer using all of the data of the user from our dataset. A special OpenCV function handles this directly. We will create a .yml file and save it in the trainer/ directory as an output this process.



Phase 2: Train the Recognizer
trainer.yml
Feed that face data and respective id's of each face to the recognizer so that it can learn.
id = 1
id = 2
dataset
Recognizer

Fig. 5. The Training Phase of the Face Recognition Mechanism[12].

Code:

"The LOCAL BINARY PATTERNS HISTOGRAMS (LBPH) Face Recognizer from the OpenCV package will be used as a recognizer"[5].

The function getImagesAndLabels (path) will return two arrays: Ids and faces, which will contain all photographs in the directory dataset/. We'll train our recognizer with those arrays as input:

recognizer.train(faces, ids)

Then, in the trainer directory that we previously created, a file trainer.yml will be saved[1][5].

```
import cv2
import oscam = cv2.VideoCapture(0)
cam.set(3, 640) # set video width
cam.set(4, 480) # set video height
face_detector = cv2.CascadeClassifier
('haarcascade_frontalface_default.xml')
# For each person, enter one numeric
    face id
face_id = input('\n enter user id end
    press <return> ==> ')
print("\n [INFO] Initializing face
    capture. Look the camera and wait
    ...")# Initialize individual
    sampling face count
count = 0
while(True):
ret, img = cam.read()
img = cv2.flip(img, -1) # flip video
    image vertically
gray = cv2.cvtColor(img,
    cv2.COLOR_BGR2GRAY)
faces =
    face_detector.detectMultiScale(gray,
    1.3, 5)
for (x,y,w,h) in faces:
cv2.rectangle(img, (x,y), (x+w,y+h),
    (255,0,0), 2)
count += 1
# Save the captured image into the
    datasets folder
cv2.imwrite("dataset/User." +
    str(face_id) + '.' +
str(count) + ".jpg", gray[y:y+h,x:x+w])
cv2.imshow('image', img)
k = cv2.waitKey(100) & 0xff # Press
    'ESC' for exiting video
if k == 27:
break
elif count >= 30: # Take 30 face
    sample and stop video
break# Do a bit of cleanup
print("\n [INFO] Exiting Program and
    cleanup stuff")
cam.release()
cv2.destroyAllWindows()
```

3) **Face Recognition:**

Our improved facial recognition system now reached its conclusion. If this person's face has been taken and trained before, our recognizer will make a prediction and return its id and an index, indicating how confident the recognizer is with this match. **Code:** Following

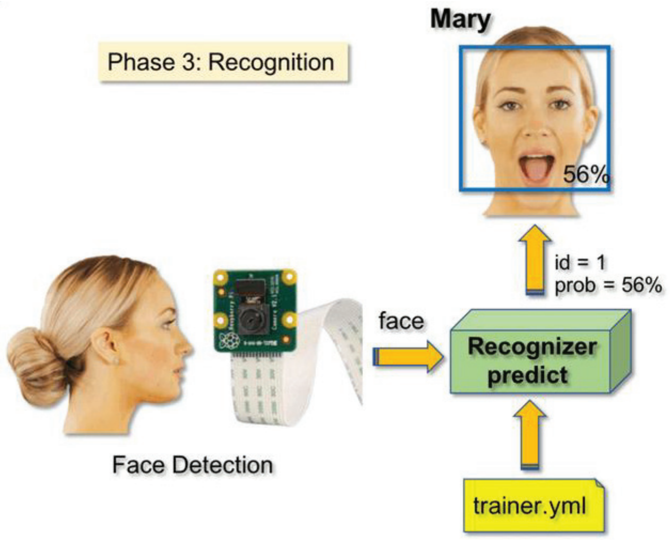


Fig. 6. The Recognition Phase of the Face Recognition Mechanism[12].

that, we'll detect a face, as we did previously with the HaarCascade classifier. The recognizer.predict () function takes a captured piece of the face to be studied as a parameter and returns its likely owner, including its id and the level of confidence the recognizer has in this match.

It's worth noting that if it's a perfect match, the confidence index will return zero. Finally, if the recognizer correctly predicted a face, we overlayed a text with the likely id and the probability in the percent of the match being correct (probability = 100 confidence index). If

```
import cv2
import numpy as np
import os recognizer =
    cv2.face.LBPHFaceRecognizer_create()
recognizer.read('trainer/trainer.yml')
cascadePath =
    "haarcascade_frontalface_default.xml"
faceCascade =
    cv2.CascadeClassifier(cascadePath);
font =
    cv2.FONT_HERSHEY_SIMPLEX#iniciate
    id counter
id = 0# names related to ids: example
    ==> Marcelo: id=1, etc
names = ['None', 'Marcelo', 'Paula',
    'Ilza', 'Z', 'W'] # Initialize and
    start realtime video capture
for(x,y,w,h) in faces:
cv2.rectangle(img, (x,y), (x+w,y+h),
    (0,255,0), 2)
id, confidence =
    recognizer.predict(gray[y:y+h,x:x+w])
# If confidence is less them 100 ==>
    "0" : perfect match
if (confidence < 100):
id = names[id]
confidence = " {0}%".format(round(100 -
    confidence))
else:
id = "unknown"
confidence = " {0}%".format(round(100 -
    confidence))
cv2.imshow('camera',img)
k = cv2.waitKey(10) & 0xff # Press
    'ESC' for exiting video
if k == 27:
break# Do a bit of cleanup
print("\n [INFO] Exiting Program and
    cleanup stuff")
cam.release()
cv2.destroyAllWindows()
```

this is not the case, the face is labelled unknown[1][5].
Result:

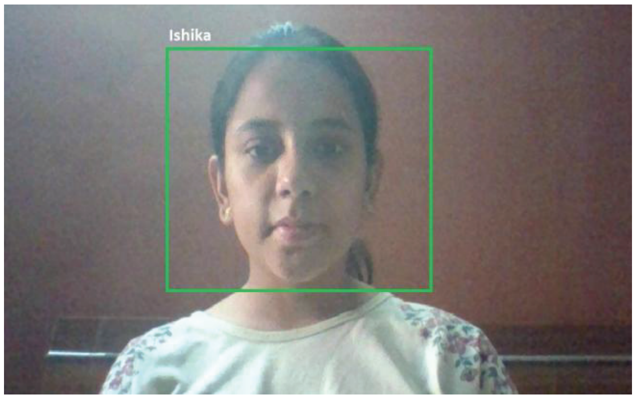


Fig. 7. The result obtained for Face Recognition as labeled unknown

V. APPLICATIONS OF FACE RECOGNITION-

Face recognition may appear futuristic and created for the future rather than for today, yet it is being employed in a variety of businesses around the world. There are numerous application fields and use cases. Furthermore, the technology is highly adaptable and can be tailored to meet the aims and demands of individual clients with just a little ingenuity and innovation.Major applications are listed below:

1) **Authentication and Payments:**
Face Recognition can be used to speed up payments and make them more convenient for both the client and the business. Consumers in some countries, such as China, can pay with their faces instead of cash or credit cards. Furthermore, throughout the payment process, Face Recognition can be utilized to verify identity. Face recognition may be used by various industries to verify that a consumer is who he claims to be [11].This capability can be used in a variety of scenarios, including mobile banking and government-related institutions.

2) **Security and Access Control:**
Face recognition can be incorporated with a variety of physical equipment and items in addition to being used as a verification measure. Face Recognition can be used to gain access to many fields of interest, such as a phone. FaceTech is a technology developed by Apple that allows users to unlock their phones with their faces. Furthermore, facial recognition can be used to guarantee that security measures are fully applied and that no room for fraud or error exists within a specific system or corporate location. Furthermore, various Facial Recognition software parts include a liveness check that helps prevent hackers from impersonating a client by using a picture of the consumer. To ensure that they are a real person, users are asked to conduct a random sequence of movements [12].The person is certified as alive only after the software validates that the sequence was completed appropriately.

3) **Monitoring of Attendance:**
Schools, universities, and other institutions may effortlessly track the attendance of their students, employees, and other visitors using face recognition software, as well as avoid any questionable conduct [13].

4) **Controlling your age:**
Face recognition is used by some businesses to check people's ages. A grocery store, for example, may use face recognition to verify the age of consumers who want to buy age-restricted items like alcohol. Face recognition, in conjunction with the camera and human monitoring, can be used in age-restricted venues such as bars and adult shops to ensure that visitors are of legal age [13].

5) **Face recognition is often used for cloud security:**Through cloud computing, customers and companies can retrieve their personal files either from internet-connected computers without having to install any application software [14]. Technologies have the ability to make enormous analysis understanding storage and analysis more efficient, less time-consuming, and quite certainly less expensive [15] [16]. The least invasive method is face recognition, and facial parameters are definitely the most often deployed biometrics parameters in personal authentication. Face recognition authentication for cloud computing is based on security concerns with cloud data analytics and data access. It can begin offering, service providers, cloud users, and organizations around the world an adequate level of security measures. A seamless system is made

possible first by facial recognition processor and facial recognition database cloud server.

6) **Facial Recognition throughout the Security from Big Data:**A software package that is constructed to analyse, organize, and extract information from a commonly used types of enormous datasets is what will be referred to those as big data [17]. In essence, it is implemented to maximise the multiple processors of organised and unstructured knowledge while utilising a minimum number of computers. Big Data cannot be characterised in terms of size because that is not a tiny thing. Big data platform's main ingredient is volume [18].To manage this same potential including the use of data, approximately every organisation today is investigating embracing big data. highlighted risks of accessing massive data even without security solutions, that might also result in the data being discarded, destroyed, or transformed. Therefore, someone could increase access assurance and stop any unapproved data access by employing face recognition software to allow access [19].
7) **Machine Learning with Facial Recognition:**Using a single machine learning algorithm, input of material, as well as output, machine learning provides provided solutions for a wide range of issues [20]. It is unnecessary for us to create our particular neural network. We have a trained classifier at our disposal that can be applied. It accomplishes what we really need it to (outputs a number of face encodings when we pass in a facial image; compares face representations of faces from various photos).

VI. Conclusion

Face recognition systems are now linked to several leading technology companies and industries, making facial recognition work simpler. When it comes to human face recognition, it is believed that the brain retains significant information such as the sizes and hues of key features such as the eyes, nose, forehead, cheeks, and lips. The use of OpenCV and Python makes it a more useful and adaptable system or tool that anyone can create according to their needs. Because it is a user-friendly and cost-effective system, the method described during this project will be beneficial to a variety of people. Today's core technologies have progressed, and the cost of equipment has decreased considerably as a result of increased integration and computing capacity. The improved system presented in this paper may appear futuristic and created for the future rather than for today, yet it is being employed in many applications around the world. Applying scheduling algorithms for multiple face recognition simultaneously can be used in future research to improve the service performance of presented facial recognition systems

References

[1] N. Pandey, P. K. Yadav, and K. Arjun, "Face recognition system using computational algorithms," in *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. IEEE, 2022, pp. 1739–1744.

[2] H. Nigam, M. N. Abbas, M. Tiwari, H. M. Shalaj, and M. N. Hasib, "Review of facial recognition techniques."

[3] E. M. Sharma, J. Singh, and E. B. Duhan, "Image enhancement based upon gwo-ga optimized algorithm using kuwahara filter," *Solid State Technology*, vol. 63, no. 5, pp. 192–206, 2020.

[4] P. Singh, M. Kaur, and J. Singh, "The review of various methods and color models in face recogntion," in *2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC)*. IEEE, 2018, pp. 233–239.

[5] M. Khan, S. Chakraborty, R. Astya, and S. Khepra, "Face detection and recognition using opencv," in *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*. IEEE, 2019, pp. 116–119.

[6] J. Singh, S. Aggarwal *et al.*, "A performed optimized load balancing genetic approach technique in cloud environment," in *Recent Trends in Communication and Intelligent Systems*. Springer, 2022, pp. 269–279.

[7] B. M. S. Hasan and A. M. Abdulazeez, "A review of principal component analysis algorithm for dimensionality reduction," *Journal of Soft Computing and Data Mining*, vol. 2, no. 1, pp. 20–30, 2021.

[8] A. Sharma, K. Shah, and S. Verma, "Face recognition using haar cascade and local binary pattern histogram in opencv," in *2021 Sixth International Conference on Image Information Processing (ICIIP)*, vol. 6. IEEE, 2021, pp. 298–303.

[9] B. Gomathy, K. S. Sathya *et al.*, "Face recognition based student detail collection using opencv," in *2022 8th International Conference on Smart Structures and Systems (ICSSS)*. IEEE, 2022, pp. 1–4.

[10] P. Shukla, R. Shrestha, and A. Karmacharya, "Face recognition attendance system opencv," EasyChair, Tech. Rep., 2022.

[11] W. Ali, W. Tian, S. U. Din, D. Iradukunda, and A. A. Khan, "Classical and modern face recognition approaches: a complete review," *Multimedia Tools and Applications*, vol. 80, no. 3, pp. 4825–4880, 2021.

[12] M. Rovai, "Real-time face recognition," https://towardsdatascience.com/real-time-face-recognition-an-end-to-end-project-b738bb0f7348, Mar 12, 2018.

[13] K. Mridha and N. T. Yousef, "Study and analysis of implementing a smart attendance management system based on face recognition tecqnique using opencv and machine learning," in *2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT)*. IEEE, 2021, pp. 654–659.

[14] J. Singh and D. Gupta, "An smarter multi queue job scheduling policy for cloud computing," *International Journal of Applied Engineering Research*, vol. 12, no. 9, pp. 1929–1934, 2017.

[15] J. Singh, B. Duhan, D. Gupta, and N. Sharma, "Cloud resource management optimization: Taxonomy and research challenges," in *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*. IEEE, 2020, pp. 1133–1138.

[16] J. Singh, R. Bajaj, and A. Kumar, "Scaling down power utilization with optimal virtual machine placement scheme for cloud data center resources: A performance evaluation," in *2021 2nd Global Conference for Advancement in Technology (GCAT)*. IEEE, 2021, pp. 1–6.

[17] J. Singh, G. Singh, and A. Verma, "The anatomy of big data: Concepts, principles and challenges," in *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1. IEEE, 2022, pp. 986–990.

[18] J. Singh, G. Singh, and B. S. Bhati, "The implication of data lake in enterprises: A deeper analytics," in *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1. IEEE, 2022, pp. 530–534.

[19] Muskan, G. Singh, J. Singh, and C. Prabha, "Data visualization and its key fundamentals: A comprehensive survey," in *2022 7th International Conference on Communication and Electronics Systems (ICCES)*, 2022, pp. 1710–1714.

[20] S. Aggarwal, A. Verma, and J. Singh, "Application based categorization of datasets for implementing data mining techniques," in *2021 2nd Global Conference for Advancement in Technology (GCAT)*. IEEE, 2021, pp. 1–7.