A Dissertation on

FACE RECOGNITION AND 3D-MESH CONSTRUCTION

Submitted to

Department of Computer Science
School of Computer Sciences

By

NITEN CHANDRA SETHY
Reg No: 2021MCA12
MCA IV Semester

Under the Guidance of

Internal Guide
DR. GURURAJ MUKARAMBI
Assistant Professor
Department of Computer Science

Central University of Karnataka
Kadaganchi, Kalaburagi - 585367, INDIA
July 12, 2023

ಕರ್ನಾಟಕ ಕೇಂದ್ರೀಯ ವಿಶ್ವವಿದ್ಯಾಲಯ
कर्नाटक केंद्रीय विश्वविद्यालय
CENTRAL UNIVERSITY OF KARNATAKA
(Established by an Act of the Parliament in 2009)
Kadaganchi, Kalaburagi – 585367, INDIA, Website: www.cuk.ac.in
Block No.17, CUK Campus

Dept. of Computer Science

Email: hodcs@cuk.ac.in

School of Computer Sciences

CERTIFICATE

This is to certify that the dissertation entitled with the **FACE RECOGNITION AND 3D-MESH CONSTRUCTION** submitted by **NITEN CHANDRA SETHY** bearing Registration Number: **2021MCA12**, studying in MCA IV Semester, to the Department of Computer Science, Central University of Karnataka, Kadaganchi, Kalaburagi, is a record of the original project work carried out by him under my guidance and supervision during the IV semester of MCA program. The work embodied in this project dissertation has not been submitted fully or in part any where else.

Guide
DR. GURURAJ MUKARAMBI
Assistant Professor
Dept. of Computer Science

Head
Dept. of Computer Science
School of Computer Science

Name and Signature of Examiner

1. ....

2. ....

DECLARATION

I hereby declare that the work embodied in this dissertation entitled with **FACE RECOGNITION AND 3D-MESH CONSTRUCTION** , submitted to Department of Computer Science Central University of Karnataka, Kadaganchi, Kalaburagi is a record of my original project work carried out by me in the Department of Computer Science, Central University of Karnataka, Kadaganchi, Kalaburagi, under the guidance of **DR. GURURAJ MUKARAMBI** , and that the full of part of this dissertation has not been submitted to this or any other University or Institute.

Place: Kadaganchi
Date: July 12, 2023

NITEN CHANDRA SETHY
MCA IV Semester
Central University of Karnataka
Kadaganchi, Kalaburagi

Acknowledgements

I would like to express my sincere gratitude to the Central University of Karnataka, for their support and guidance in the completion of this project. I am particularly grateful to the Honorable Vice-Chancellor, Professor Battu Satyanarayana, for his vision and leadership.

I would also like to thank the Dean of School of Computer Science, Professor R S Hegadi and the Head of the Department of Computer Science, Professor Srikantaiah K C, for their encouragement and support.

I would like to extend my heartfelt appreciation to my guide **DR. GURURAJ MUKARAMBI** for their valuable insights, suggestions and guidance. Their expertise and insights have greatly contributed to the successful completion of this work.

I am grateful to the faculty members of the Department of Computer Science, Dr.P.M. Kamble, Dr.Kagale Madhuri R, Ms. Shilpa Sthavarikmath, and Dr. Sukanya S. Gaikwad for their valuable guidance, support, and encouragement throughout the course of this project.

Furthermore, I would like to acknowledge my fellow classmates and friends for their camaraderie, motivation, and intellectual discussions, which have enriched my learning experience and provided a supportive environment for academic growth.

Last but not least, I would like to express my deepest gratitude to my family for their unwavering love, understanding, and encouragement throughout my academic journey. Their constant support has been the driving force behind my achievements.

Place: Kadaganchi
Date: July 12, 2023

NITEN CHANDRA SETHY
MCA IV Semester
Central University of Karnataka
Kadaganchi, Kalaburagi

Abstract

Facial recognition is a method/way of recognizing or verifying a person's identification by looking at their facial traits. People can be identified in pictures, films, or in real-time using facial recognition technology. Face recognition is the task of identifying an already detected/identified object as a known or unknown face.

Face-recognition/identification is based on machine learning, and a cascade function is trained using both a large number of positive and negative images. It is generally used to detect objects/faces in other images. all face recognition algorithm consists of face detection and face face identification. In this paper, I will be preparing my own data as well as taking global dataset ie ORL dataset and by using different algorithms like GLCM ,SIFT,CNN and mixing features and classifying using different classifiers i will be finding accuracy and will be giving a comparative analysis about dataset. And in 3D I will be explaining about how to generate mesh from 2d-images.

Contents

List of Figures

Chapter 1

INTRODUCTION

1.1 BACKGROUND

One of the simplest methods to distinguish between each individual identities is to look at each other's faces. Face recognition is a type of personal identification system that uses a person's personal traits/characters to determine their identity. The human face recognition mechanism is divided into two phases: face detection, which occurs relatively quickly in humans unless in situations where the item is positioned at a short distance away, and identification, which recognizes a face as an individual from the facial features/traits. Face recognition is broadly classified into 2 types (i) 2D-face recognition system (ii)3d-face recognition system . A 2D-face recognition system is particularly is designed to capture images of face using standard cameras like smartphone. In this process , it extracts the facial features like eye, mouth , nose, texture etc. as features, then the extracted data is stored in database for comparison or matching against newly captured images to verify an individuals identity.The main core of the face recognition lies in its algorithm which consists of various types of image processing techniques to make the images clean,feature extraction and recognition .But it have some flaws and to remove those i.e., while capturing the images .Due to these reasons , need of 3D face-recognition.
3D face-recognition system was developed as a more accurate and reliable method of identifying and verifying individuals than traditional 2D face recognition systems. Unlike 2D systems, which rely on images captured from a single camera angle, 3D systems use depth information to create a three-dimensional model

of a face, which can then be compared to a database of known faces. 3D face recognition has ability to accurately identify individuals under varying lighting conditions and facial expressions and also its ability to provide higher security and accuracy in identification, making it particularly useful in applications such as security and surveillance.

1.2 LITERATURE SURVEY

*** Face Recognition: A Literature Review:** A.S.Tolba et al.[1] explained in his research paper about that description and limitation of face database . they have considered FRTV (face recgnition vendor test)2002 for the task. in this they have considered the methods like eigenfaces (eigenfeatures),dynamic link architecture, neural networks, Hidden Markov Model, geometrical feature matching, and template matching. (a)On Eigenfaces they got an accuracy of 96,85 and 64 percentage averaged over lighting, orientation, over size variations, for this they have considered the FERET database having 7562 images . (b) Neural Networks they got an accuracy of 96.2 percentage on ORL database of 400 images of 40 individuals. (c) Graph matching is also one of the method they have used in which object recognition is done by finding the closest stored graph. this database consist of 111 faces of 15 degree rotation and 110 faces of 30degree rotation and 112 front view and got an accuracy of 86.5 percentage. (d)Using HMM approach they got an accuracy of 87 percentage on ORL dataset. (e) Geometrical feature matching techniques, these are generally based on geometrical features from the picture of a face such as eyes, nose, mouth etc and got an accuracy upto 95 percentage.[1]

***A Review of Face Recognition Technology**:LIXIANG LI et al.[2] described in their paper about the development stages of face recognition and related technologies and the accuracy they got . Here they have used the databases like GeorgiaTech , AR, Extended Yale B,Yale A, LFW, FERET and got a accuracy range of 84.72 percentage to 99.70 percentage . In this they have done manual design features , non negative matrix factorization and used new data resource.[2]

*** Human Face recognition Based on convolutional neural networks(CNN) and augmented dataset** Peng Lu et al.[3] explained about how to deal small original dataset. They did data augmentation of face dataset to achieve higher accuracy than others using CNN for this they have considered the ORL face dataset. when data is augmented it produced data set having rotation , shift , scaling and flip which helped in producing more unique features, they have carried several and achieved an accuracy of 99.5 percentage which is the highest accuracy they have received during there experiments , with ANN they received an accuracy of 80.3 and when with combination of PCA and ANN it's accuracy is 91.0 , with PCA and SVM it's accuracy is 97.40 with wavelets and SVM they got accuracy up-to 98.1 and while merging wavelets , PCA and SVM they accuracy is 98.0. they mentioned that the proposed is , an economic method to augment the small dataset.[3]

***Automatic Face Recognition Based on Sparse Representation and Extended Transfer Learning**: ZHI LIU el at.[4] described in his paper about the efficient strategy to enhance sparse representation to solve problems with a small-size dataset. They developed a weighted fusion scheme in ORL dataset and using their algorithm they got an accuracy of 95 percentage and in FERET dataset the accuracy rate is 95 percentage too. and with LFW dataset they got an accuracy of 83.33 percentage . This paper will be very much helpful for the identification based on IMT (Internet-of-Medical-Things.)[4]

***Face Detection and Recognition System using Digital Image Processing** :Gurlove Singh and Amit Kumar Goel el at.[5] described about the face recognition system development by using Digital image processing . They have classified recognition system into 2 main approach (i) Geometric ie., nose , eye, mouth .Based on these categories the face is first separated and then estimation is done.(ii) Photometric stereo , It is a cv technology which recuperates the structure of an underlying objects from the image that were taken by different light conditions. While doing classification they combined MATLAB and NN toolbox . They have considered two basic approach (i)Feature Based Approach , in this (a) they started with Active shape model (ASM) whose work is to automatically locate the benchmark point points which will draw the appearance of statistically craved entity in picture. (b) Low Level Analysis consists of color processing as it is faster .there are basically 3 types of algo HIS ,YCbCr and RBG. there are 3 steps that are needed to be followed in the implementation of these algo first categorizing the sink area in the color then putting the threshold to disguise the skin are in the picture and at last for detecting a face image drawing a bounding box. (c) Feature Analysis the main focus of these algo is that they remain into the existence even if the lighting condition and viewing angle (ii)Image Based approach. here also they have mentioned 3 section (a) Neural Networks (b)Linear Sub-Space Method (LSSM) and (c)Statistical Approach here thwy have used PCA for Eigen faces and after the processing they have got an accuracy of 90 percentage. [5]

***3D Face Recognition** Berk G¨okberk, Albert Ali Salah, Ne¸se Aly¨uz, Lale Akarun et al.[6] explained in there paper that, how 3d face recognition has evolved in last decade they discussed about what real-world scenarios and acquisition technologies are needed.For Acquisition 3d camera with IR and a proper lighting condition while capturing the images. He also mentioned about the golbal datasets available for the task like BU-3DFE, ASU PRISM,Bosphorus , BJUT-3D Texsas 3D etc.1) pre-processing of raw 3D facial datas, 2) preprocessing of faces, 3) feature extraction, and 4) matching. Here they have used the Bosphorus database and their experiment they got an accuracy upto 95 percentage.and they have concluded that 3D face recognition has matured to match the performance of 2D face recognition and the accuracy of algorithms have met requirements in controlled tests.[6]

***3D face recognition based on machine learning**: S. Qatawneh, S. Ipson et al.[7] described that how they have made 3D facial system based on the machine learning with the use of landmarks for feature extraction and Cascade Correlation neural network to make the decision. For this they have taken 3D face images from the Face Recognition Grand Challenge database with CNN using Jack-knife they got an accuracy of 100percentage . First they have extracted the facial area then after that they have inserted the Z axis and removed the spikes [7]

***MobileFace: 3D Face Reconstruction with Efficient CNN Regression**:in this paper Nikolai Chinaev et al.[8] explain about the 3D face reconstruction ie estimation of facial shape using CNN. They have created a Morphable Model allow to generate variability in both face identity and expressions. they have tried both geometric method as well as projection method. For the image formation they have created a texture model , lighting model , calculation of energy function, regularization and optimization. They have taken BU4DFE dataset.And they have concluded that their neural network perform faster speed and smaller model size.[8]

1.3 OBJECTIVES

1. To create own dataset through camera capturing and comparing with standard dataset.
2. To use of existing feature extraction algorithms for own dataset as well as standard dataset.
3. To use of existing classifier for face classification.
4. Comparative analysis on proposed work with existing work found in the literature.
5. To create a 3D mesh based on 2D images.

1.4 DATA COLLECTION

Data collection is done in two methods , (i)Creation of own dataset ,(ii) Globally available dataset.
(i)creation of own dataset can be done by using the web-cam , in a proper lighting and need a proper environment. (ii) Globally available dataset can be downloaded from the internet. Globally available dataset from internet (https://cam-orl.co.uk/facedatabase.html) which is developed by AT&T Laboratories Cambridge.
The main purpose of designing the dataset is that they have carry out the face recognition technology with carried out in collaboration with the Speech, Vision and Robotics Group of Cambridge University.

1.5 APPLICATION

1)Access-Control-Systems: Face recognition/identification technology plays a essential role in access-control-systems(ACS), replacing traditional methods i.e, keycards or PIN codes. By integrating face-recognition algorithms into the access-control-system(ACS), organizations can ensure that only authorized individuals gain entry to secure areas. Face recognition/identification technology not only enhances security but also offers convenience by eliminating the need for physical tokens/passwords etc.

2)Secure-Financial-Transactions: Face recognition/identification technology can be utilized in financial institutions to provide secure transactions. By linking an individual's face to their financial account, this technology ensures that, only the authorized account holder can access funds or perform transactions no other can do any kind of transactions. Additionally, facial recognition/identification can be employed as an extra layer of security during online banking, preventing un-authorized access and protecting against identity theft and many more.

3)Surveillance-Systems: Face recognition/identification technology has become an necessary as well as very much essential tool in surveillance systems, enabling real-time identification of individuals in public spaces. By comparing live video feeds with a database of known faces, authorities can quickly identify and track individuals . This technology has proven extremely useful in enhancing public safety, aiding law enforcement agencies in identifying suspects, and preventing criminal activities.

4)Attendance and Time Tracking: Face recognition/identification technology can automate attendance tracking in educational institutions and workplaces. By using facial recognition/identification algorithms, it accurately records the presence of individuals, eliminating the need for manual attendance. This ensures accuracy and reduces the chances of identity fraud, ultimately saving time and resources.

Chapter 2

FEATURE EXTRACTION

2.1 GRAY-LEVEL CO-OCCURRENCE MATRIX

- Gray-Level Co-occurrence Matrix(GLCM) [9], is a method used in image processing and computer vision to perform texture analysis.It is utilized to extract statistical information from an image regarding the spatial relationships of pixel intensity values. It is used in the tasks such as tasks such as image classification, segmentation, and feature extraction.Basic idea of GLCM is to analyze how often different pairs of pixel intensities occurs relative to each other at different spatial displacements or directions in an image. The matrix is constructed based on these co-occurring pixel pairs and their frequencies, which provides useful information about the texture present in the image.

Working:First the image is converted to a grayscale format as GLCM operates on single-channel images.Then GLCM is computed by analyzing the neighboring pixel pairs at different displacement vectors (offsets) in various directions, such as horizontally, vertically, and diagonally and the displacement vector determines how far apart the pixels are considered. Once the GLCM matrix is calculated and normalized, various statistical measures can be computed from it to describe the texture properties of the image. Texture features extracted from GLCM are:

Contrast: Measures the local intensity variations between neighboring pixels.

Energy: Represents the sum of squared elements in the GLCM, indicating the homogeneity of the texture.

Homogeneity: Captures the closeness of the diagonal elements in the GLCM,

indicating the similarity of neighboring pixel pairs.
Entropy: Reflects the randomness of the texture.
Correlation: Indicates the linear dependency between pixel pairs.

2.2 SIFT(Scale-Invariant Feature Transform)

-SIFT (Scale-Invariant Feature Transform)[9] is a widely used feature extraction technique in computer vision and image processing. It was introduced by David Lowe in 1999.
Working SIFT starts by constructing a scale-space representation of the image using Gaussian blurring at multiple scales. The image is convolved with a series of Gaussian filters of increasing standard deviation. This process creates a pyramid of images, each representing a different scale level.The difference between adjacent scales in the scale-space pyramid is computed to highlight regions of significant intensity changes, which correspond to potential feature points.this process is known as Difference of Gaussians (DoG) pyramid.In the DoG pyramid, potential keypoint locations are identified as local extrema compared to their surrounding pixels in both scale and space. This step involves comparing each pixel with its eight neighbors in the current scale level, as well as nine pixels in each of the adjacent scale levels.o enhance the stability and accuracy of the detected keypoints, SIFT applies a series of refinement steps. These include rejecting low-contrast keypoints and removing keypoints along edges or with poorly determined orientations.It calculates gradient magnitudes and orientations in the local neighborhood around each keypoint and builds a histogram of orientations. The highest peak in the histogram is chosen as the keypoint's dominant orientation.this process is known as Orientation Assignment. SIFT constructs a local feature descriptor for each keypoint based on the gradients in its local neighborhood. This descriptor captures information about the distribution of gradients and their orientations. The descriptor is normalized to be invariant to changes in illumination, scale, and rotation. and by this Feature Descriptor Calculation is done. The final step involves comparing the descriptors of keypoints from different images to establish matches. It is known as Keypoint Matching.

2.3 CONVOLUTIONAL NEURAL NETWORK (CNN)

-CNN (Convolutional Neural Network) is a deep learning architecture widely used for tasks such as image recognition, object detection, and image classification.

Working CNN starts with convolutional layers, where filters (also called kernels) are applied to the input data. These filters slide over the input image to detect local patterns or features, such as edges, corners, or textures. Convolution captures spatial relationships, and multiple filters can be used to detect different patterns. After the convolution step, an activation function (usually ReLU - Rectified Linear Unit) is applied element-wise to introduce non-linearity into the network, allowing it to learn more complex patterns.Pooling layers follow the activation functions and are used to down sample the spatial dimensions of the data. Common pooling techniques include max pooling, which retains the maximum value in each pooling region, and average pooling, which takes the average value. Pooling reduces computation and helps make the model more robust to slight variations in input data. After several convolutional and pooling layers, the feature maps are flattened and passed through fully connected layers. These layers generally perform high-level reasoning and decision-making based on the features extracted from previous layers.he final layer of the CNN is the output layer, which typically consists of one or more neurons depending on the task. For image classification, the number of neurons is equal to the number of classes, and the Softmax activation function is often used to convert the output scores into probabilities, indicating the likelihood of each class.CNNs are trained using labeled data and an optimization algorithm like gradient descent to minimize the difference between predicted outputs and ground-truth labels.

2.4 MIX-FEATURES (GLCM ,LBP , GABOR)

-Mix Features1: In this section i have , mixed the 3 algorithm like GLCM ,LBP , GABOR . I have taken the features like 'LBP Energy', 'LBP Entropy', 'Contrast', 'Dissimilarity', 'Homogeneity', 'Energy', 'Correlation', 'Gabor Energy', 'Gabor Entropy'.

2.5 MIX-FEATURES (GLCM and GABOR)

-Mix feature2: This section contains the combination of the 2 algorithm like GLCM and GABOR

Chapter 3

EXPERIMENTAL RESULTS AND DISCUSSIONS OF 2D FACE RECOGNITION SYSTEM

3.1 WORKING

The working of face recognition technology involves several steps:

***Data acquisition**: The first step in face recognition is to capture a the face. This is typically done using a camera .

***Preprocessing**: The captured data is then pre-processed to remove any noise or distortion, normalization , and prepare it for feature extraction.

***Feature extraction**: Feature extraction algorithms are used to identify key features of the face, such as the shape of the eyes, nose, and mouth, and create a digital representation of the face.

*** Matching**: The digital representation of the face is then compared to a database of known faces to identify the person. This is done using matching algorithms that compare the features extracted from the face data to those in the database.

***Decision-making**: Based on the results of the matching process, a decision is made about the identity of the person. This can be a binary decision (matched or not matched) or a ranking of potential matches.

***Authentication or identification**: Finally, the system either authenticates the identity of the person (i.e., verifies that they are who they claim to be) or identifies them (i.e., determines their identity from a pool of potential matches available).

3.1.1 FLOW DIAGRAM

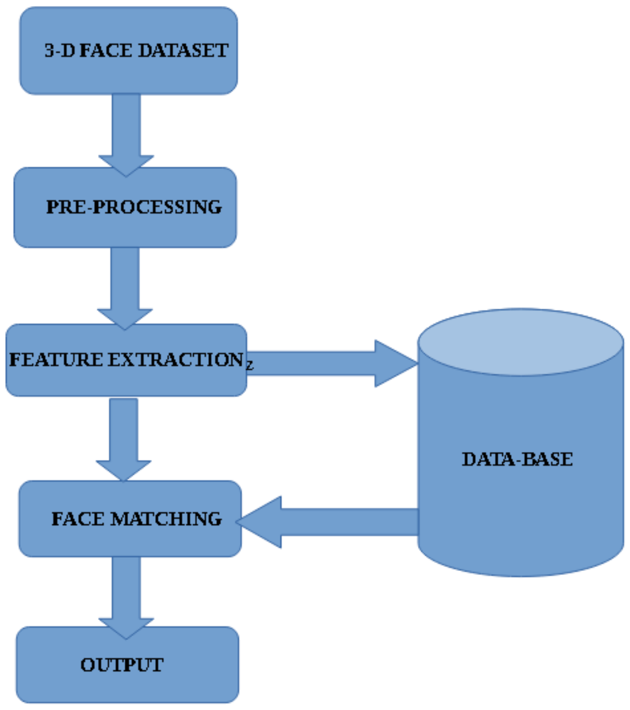


Figure 3.1: DATA FLOW DIAGRAM

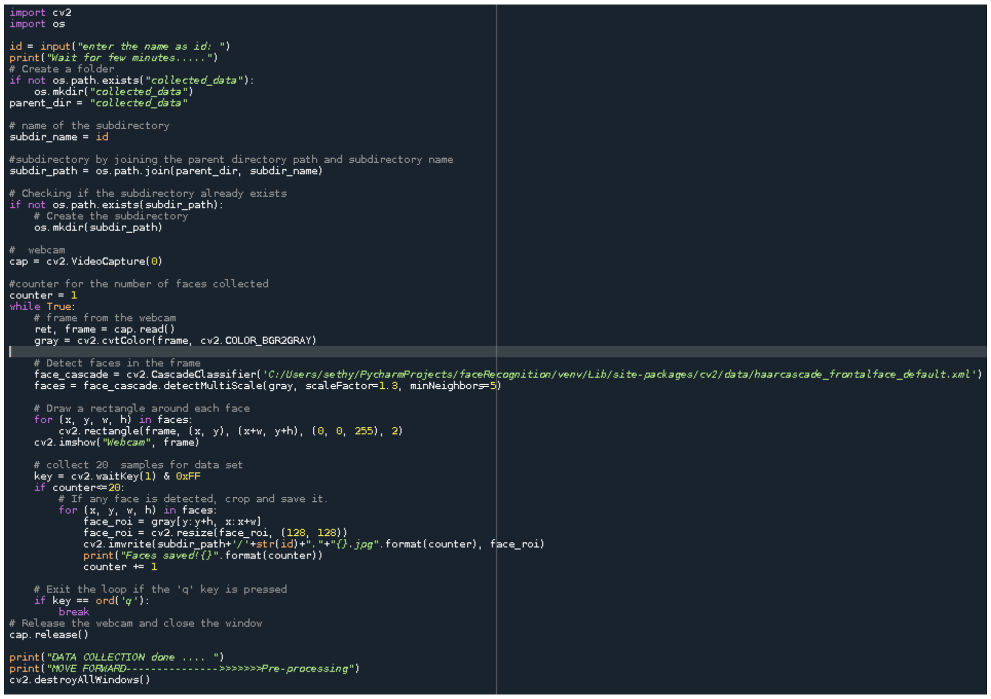
3.2 METHODOLOGY

In this section i will be mentioning the methodology i have used to implement the 2d-face recognition successfully.
The project was coded in python using Spyder IDE. The first step is to create face detection system with the use of Haar-cascade . OpenCV has a robust set of Haar-cascade that i have used in the project.
The basic requirement needed for this project is a computational device , python IDE , knowledge of machine learning algorithms and digital image processing.
The following are the steps/process I have implemented :
-Dataset collection The first step is to have a data-set upon which we will be working. We can either make own dataset or we can use globally accepted and available dataset.

I have done on both ,first i have created my own dataset using my web camera available in the laptop . i have collected 20 images of each person and stored in the folder. Each new person face is assigned a new id and the sub folders will be created with the name/id of the person.

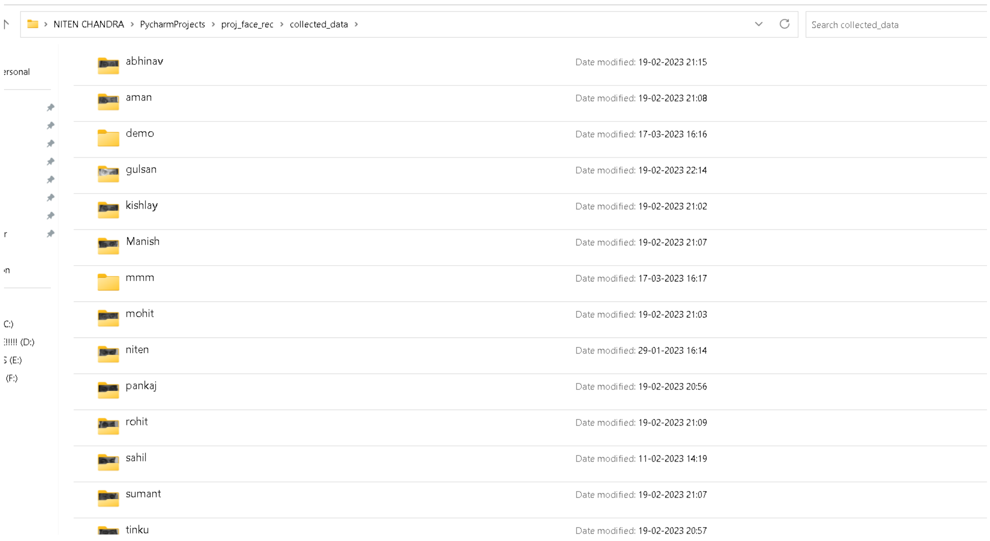
The main objective is to detect face in real time . To make this we need to use computer vision library . Here i have used the function VideoCapture() to gain access to web cam and mentioned the argument 0 as im using my in-built camera . for external camera we can mention argument as 1 while capturing the images we will be converting the images into gray scale and resizing the images.

And also considered ORL a golbal dataset having total images of 400, 10 samples of each person for the recognition.
sample code:



```python
import cv2
import os

id = input("enter the name as id: ")
print("Wait for few minutes.....")
# Create a folder
if not os.path.exists("collected_data"):
    os.mkdir("collected_data")
parent_dir = "collected_data"

# name of the subdirectory
subdir_name = id

#subdirectory by joining the parent directory path and subdirectory name
subdir_path = os.path.join(parent_dir, subdir_name)

# Checking if the subdirectory already exists
if not os.path.exists(subdir_path):
    # Create the subdirectory
    os.mkdir(subdir_path)

#  webcam
cap = cv2.VideoCapture(0)

#counter for the number of faces collected
counter = 1
while True:
    # frame from the webcam
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detect faces in the frame
    face_cascade = cv2.CascadeClassifier('C:/Users/sethy/PycharmProjects/faceRecognition/venv/Lib/site-packages/cv2/data/haarcascade_frontalface_default.xml')
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)

    # Draw a rectangle around each face
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 255), 2)
    cv2.imshow("Webcam", frame)

    # collect 20  samples for data set
    key = cv2.waitKey(1) & 0xFF
    if counter<=20:
        # If any face is detected, crop and save it.
        for (x, y, w, h) in faces:
            face_roi = gray[y:y+h, x:x+w]
            face_roi = cv2.resize(face_roi, (128, 128))
            cv2.imwrite(subdir_path+'/'+str(id)+"."+"{}.jpg".format(counter), face_roi)
            print("Faces saved!{}".format(counter))
            counter += 1

    # Exit the loop if the 'q' key is pressed
    if key == ord('q'):
        break
# Release the webcam and close the window
cap.release()

print("DATA COLLECTION done .... ")
print("MOVE FORWARD--------------->>>>>>>Pre-processing")
cv2.destroyAllWindows()
```

Figure 3.2: SAMPLE CODE FOR DATA COLLECTION

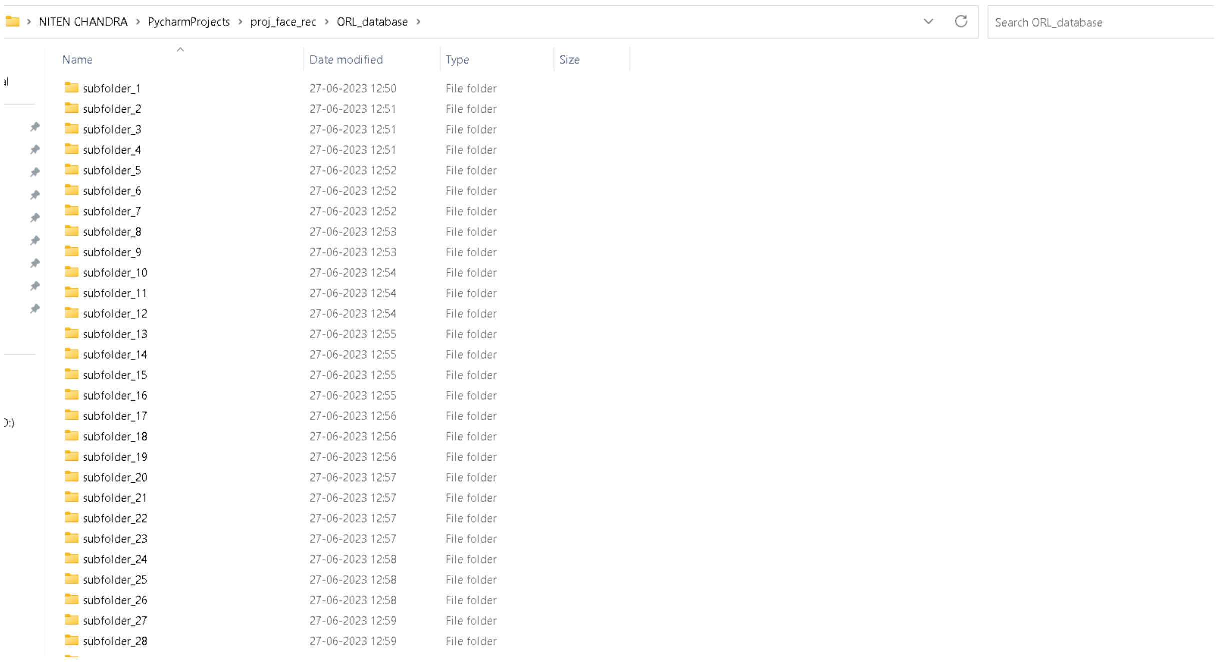


NITEN CHANDRA › PycharmProjects › proj_face_rec › collected_data ›

Search collected_data

```
abhinav     Date modified: 19-02-2023 21:15
aman        Date modified: 19-02-2023 21:08
demo        Date modified: 17-03-2023 16:16
gulsan      Date modified: 19-02-2023 22:14
kishlay     Date modified: 19-02-2023 21:02
Manish      Date modified: 19-02-2023 21:07
mmm         Date modified: 17-03-2023 16:17
mohit       Date modified: 19-02-2023 21:03
niten       Date modified: 29-01-2023 16:14
pankaj      Date modified: 19-02-2023 20:56
rohit       Date modified: 19-02-2023 21:09
sahil       Date modified: 11-02-2023 14:19
sumant      Date modified: 19-02-2023 21:07
tinku       Date modified: 19-02-2023 20:57
```

Figure 3.3: MY OWN DATABASE



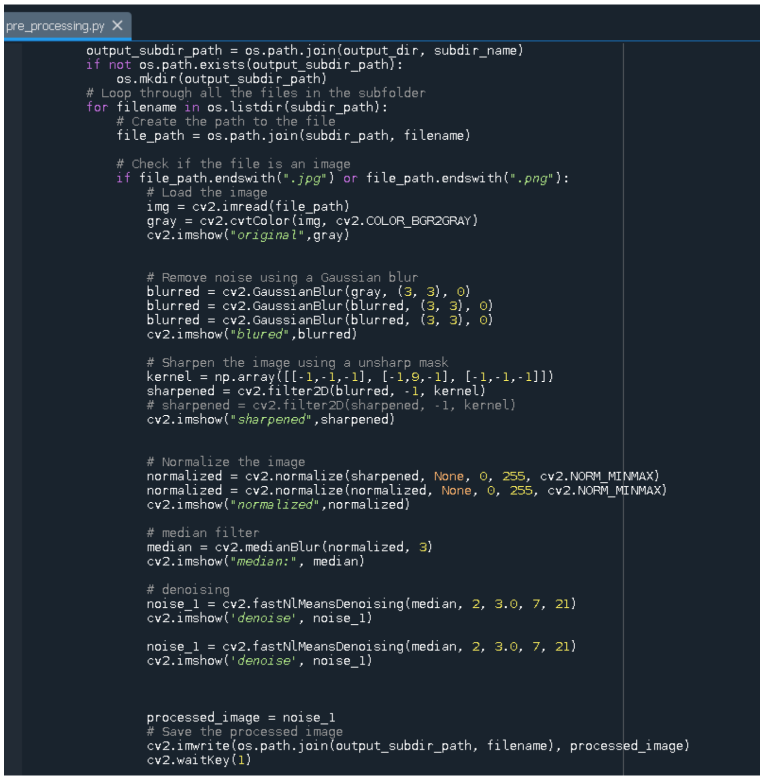
NITEN CHANDRA › PycharmProjects › proj_face_rec › ORL_database ›

Search ORL_database

Name	Date modified	Type	Size
subfolder_1	27-06-2023 12:50	File folder	
subfolder_2	27-06-2023 12:51	File folder	
subfolder_3	27-06-2023 12:51	File folder	
subfolder_4	27-06-2023 12:51	File folder	
subfolder_5	27-06-2023 12:52	File folder	
subfolder_6	27-06-2023 12:52	File folder	
subfolder_7	27-06-2023 12:52	File folder	
subfolder_8	27-06-2023 12:53	File folder	
subfolder_9	27-06-2023 12:53	File folder	
subfolder_10	27-06-2023 12:54	File folder	
subfolder_11	27-06-2023 12:54	File folder	
subfolder_12	27-06-2023 12:54	File folder	
subfolder_13	27-06-2023 12:55	File folder	
subfolder_14	27-06-2023 12:55	File folder	
subfolder_15	27-06-2023 12:55	File folder	
subfolder_16	27-06-2023 12:55	File folder	
subfolder_17	27-06-2023 12:56	File folder	
subfolder_18	27-06-2023 12:56	File folder	
subfolder_19	27-06-2023 12:56	File folder	
subfolder_20	27-06-2023 12:57	File folder	
subfolder_21	27-06-2023 12:57	File folder	
subfolder_22	27-06-2023 12:57	File folder	
subfolder_23	27-06-2023 12:57	File folder	
subfolder_24	27-06-2023 12:58	File folder	
subfolder_25	27-06-2023 12:58	File folder	
subfolder_26	27-06-2023 12:58	File folder	
subfolder_27	27-06-2023 12:59	File folder	
subfolder_28	27-06-2023 12:59	File folder	

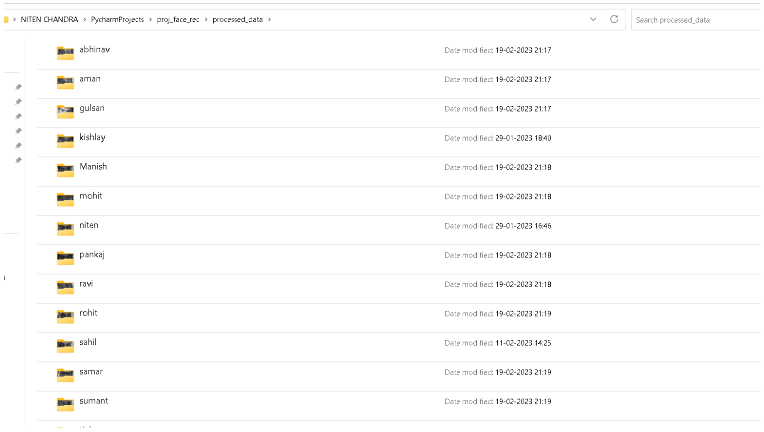
Figure 3.4: ORL-DATABASE

-Preprocessing:: Pre-processing is a method of cleaning the dataset in such a way that it will be well suited for the use . Here, i have taken a parent directory where all my images are there and another folder to store the processed data . Here i have applied Gaussian blur,done sharpening, then normalized on image then again used median filter and done denoising . This will be done to all images save those in another folder .


pre_processing.py

```python
        output_subdir_path = os.path.join(output_dir, subdir_name)
        if not os.path.exists(output_subdir_path):
            os.mkdir(output_subdir_path)
        # Loop through all the files in the subfolder
        for filename in os.listdir(subdir_path):
            # Create the path to the file
            file_path = os.path.join(subdir_path, filename)

            # Check if the file is an image
            if file_path.endswith(".jpg") or file_path.endswith(".png"):
                # Load the image
                img = cv2.imread(file_path)
                gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
                cv2.imshow("original",gray)


                # Remove noise using a Gaussian blur
                blurred = cv2.GaussianBlur(gray, (3, 3), 0)
                blurred = cv2.GaussianBlur(blurred, (3, 3), 0)
                blurred = cv2.GaussianBlur(blurred, (3, 3), 0)
                cv2.imshow("blured",blurred)

                # Sharpen the image using a unsharp mask
                kernel = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-1]])
                sharpened = cv2.filter2D(blurred, -1, kernel)
                # sharpened = cv2.filter2D(sharpened, -1, kernel)
                cv2.imshow("sharpened",sharpened)


                # Normalize the image
                normalized = cv2.normalize(sharpened, None, 0, 255, cv2.NORM_MINMAX)
                normalized = cv2.normalize(normalized, None, 0, 255, cv2.NORM_MINMAX)
                cv2.imshow("normalized",normalized)

                # median filter
                median = cv2.medianBlur(normalized, 3)
                cv2.imshow("median:", median)

                # denoising
                noise_1 = cv2.fastNlMeansDenoising(median, 2, 3.0, 7, 21)
                cv2.imshow('denoise', noise_1)

                noise_1 = cv2.fastNlMeansDenoising(median, 2, 3.0, 7, 21)
                cv2.imshow('denoise', noise_1)


                processed_image = noise_1
                # Save the processed image
                cv2.imwrite(os.path.join(output_subdir_path, filename), processed_image)
                cv2.waitKey(1)
```

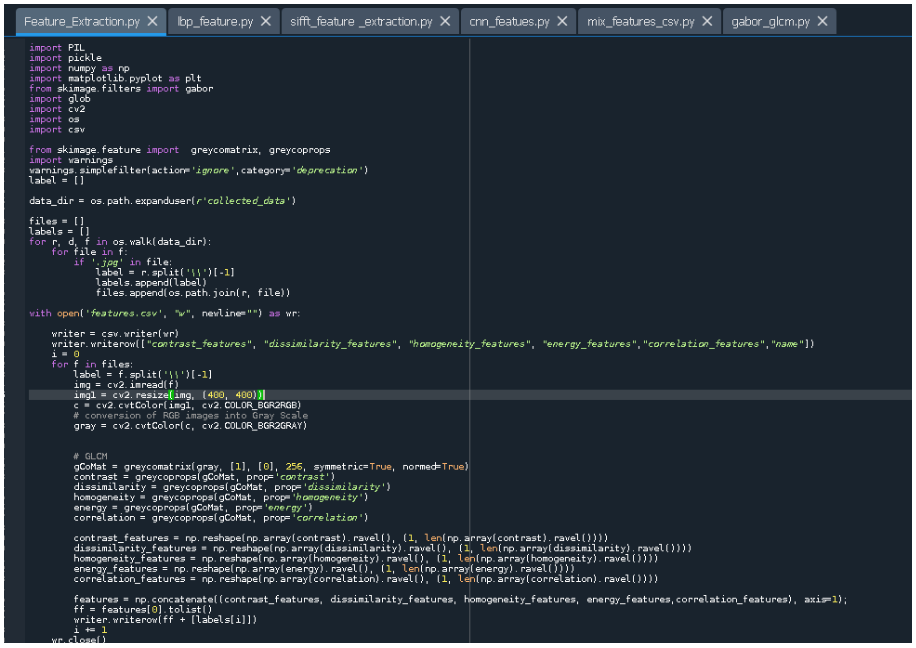
Figure 3.5: SAMPLE CODE FOR PRE-PROCESSING


NITEN CHANDRA › PycharmProjects › proj_face_rec › processed_data ›
Search processed_data

```
abhinav     Date modified: 19-02-2023 21:17
aman        Date modified: 19-02-2023 21:17
gulsan      Date modified: 19-02-2023 21:17
kishlay     Date modified: 29-01-2023 18:40
Manish      Date modified: 19-02-2023 21:18
mohit       Date modified: 19-02-2023 21:18
niten       Date modified: 29-01-2023 16:46
pankaj      Date modified: 19-02-2023 21:18
ravi        Date modified: 19-02-2023 21:18
rohit       Date modified: 19-02-2023 21:19
sahil       Date modified: 11-02-2023 14:25
samar       Date modified: 19-02-2023 21:19
sumant      Date modified: 19-02-2023 21:19
```

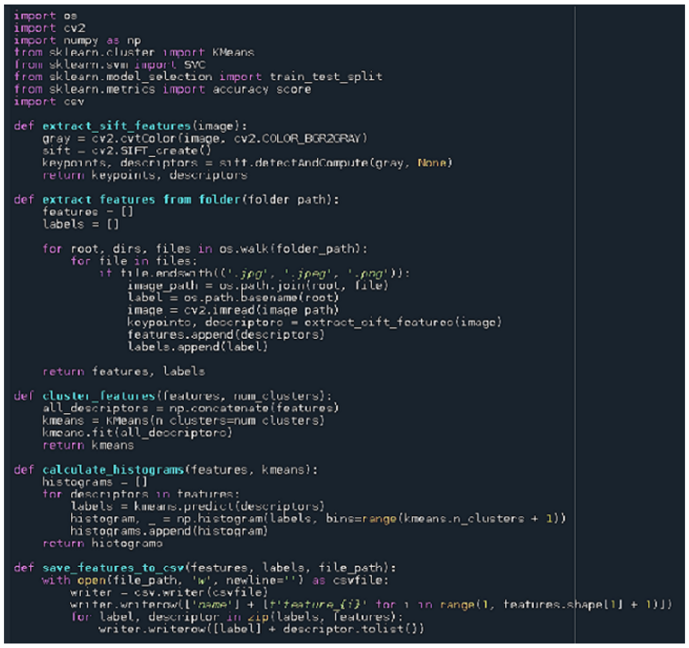
Figure 3.6: PRE-PROCESSED DATASET

-Feature Extraction:Feature extraction is a fundamental as well as very crucial process where meaningful and informative characteristics or patterns (features) are extracted from raw datas. These features serve as a condensed representation of the data that can be used for various tasks like classifications, clustering, object recognition, and many more. Feature extraction plays a crucial role in transforming raw data into a suitable format that machine learning algorithms can work with effectively. Here i have used the algorithm for feature extraction are GLCM ,SIFT ,CNN , GLCM+LBP+GABOR ,GLCM+GABOR . I have extracted the feature and saved in the csv file format.
Sample Codes:



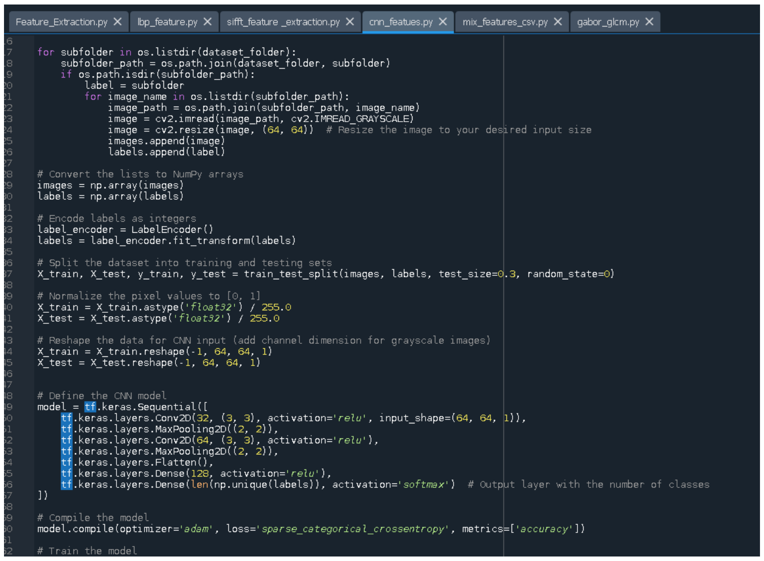
```
import PIL
import pickle
import numpy as np
import matplotlib.pyplot as plt
from skimage.filters import gabor
import glob
import cv2
import os
import csv

from skimage.feature import  greycomatrix, greycoprops
import warnings
warnings.simplefilter(action='ignore',category='deprecation')
label = []

data_dir = os.path.expanduser(r'collected_data')

files = []
labels = []
for r, d, f in os.walk(data_dir):
    for file in f:
        if '.jpg' in file:
            label = r.split('\\')[-1]
            labels.append(label)
            files.append(os.path.join(r, file))

with open('features.csv', "w", newline="") as wr:

    writer = csv.writer(wr)
    writer.writerow(["contrast_features", "dissimilarity_features", "homogeneity_features", "energy_features","correlation_features","name"])
    i = 0
    for f in files:
        label = f.split('\\')[-1]
        img = cv2.imread(f)
        img1 = cv2.resize(img, (400, 400))
        c = cv2.cvtColor(img1, cv2.COLOR_BGR2RGB)
        # conversion of RGB images into Gray Scale
        gray = cv2.cvtColor(c, cv2.COLOR_BGR2GRAY)


        # GLCM
        gCoMat = greycomatrix(gray, [1], [0], 256, symmetric=True, normed=True)
        contrast = greycoprops(gCoMat, prop='contrast')
        dissimilarity = greycoprops(gCoMat, prop='dissimilarity')
        homogeneity = greycoprops(gCoMat, prop='homogeneity')
        energy = greycoprops(gCoMat, prop='energy')
        correlation = greycoprops(gCoMat, prop='correlation')

        contrast_features = np.reshape(np.array(contrast).ravel(), (1, len(np.array(contrast).ravel())))
        dissimilarity_features = np.reshape(np.array(dissimilarity).ravel(), (1, len(np.array(dissimilarity).ravel())))
        homogeneity_features = np.reshape(np.array(homogeneity).ravel(), (1, len(np.array(homogeneity).ravel())))
        energy_features = np.reshape(np.array(energy).ravel(), (1, len(np.array(energy).ravel())))
        correlation_features = np.reshape(np.array(correlation).ravel(), (1, len(np.array(correlation).ravel())))

        features = np.concatenate((contrast_features, dissimilarity_features, homogeneity_features, energy_features,correlation_features), axis=1);
        ff = features[0].tolist()
        writer.writerow(ff + [labels[i]])
        i += 1
    wr.close()
```

Figure 3.7: SAMPLE CODE FOR GLCM FEATURE EXTRACTION



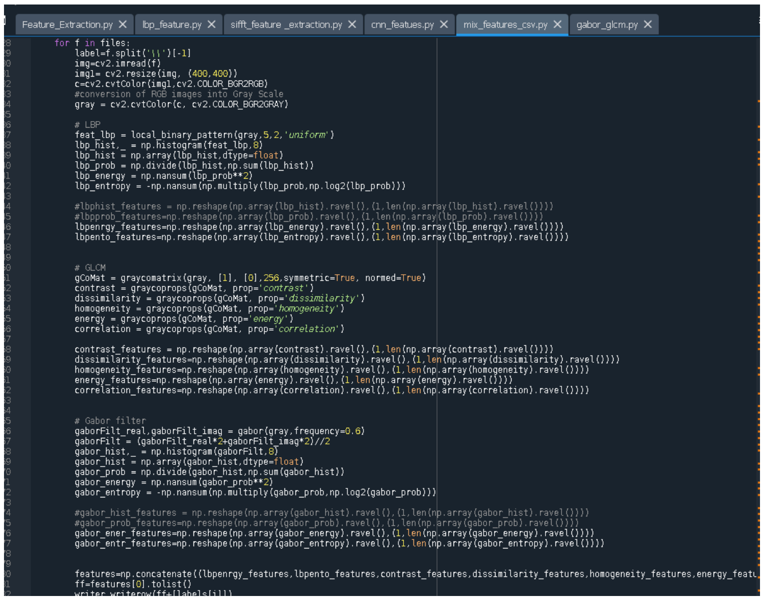
```python
import os
import cv2
import numpy as np
from sklearn.cluster import KMeans
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy score
import csv

def extract_sift_features(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    sift = cv2.SIFT_create()
    keypoints, descriptors = sift.detectAndCompute(gray, None)
    return keypoints, descriptors

def extract features from folder(folder path):
    features = []
    labels = []

    for root, dirs, files in os.walk(folder_path):
        for file in files:
            if file.endswith(('.jpg', '.jpeg', '.png')):
                image_path = os.path.join(root, file)
                label = os.path.basename(root)
                image = cv2.imread(image path)
                keypoints, descriptors = extract_sift_features(image)
                features.append(descriptors)
                labels.append(label)

    return features, labels

def cluster_features(features, num_clusters):
    all_descriptors = np.concatenate(features)
    kmeans = KMeans(n clusters=num clusters)
    kmeans.fit(all_descriptors)
    return kmeans

def calculate_histograms(features, kmeans):
    histograms = []
    for descriptors in features:
        labels = kmeans.predict(descriptors)
        histogram, _ = np.histogram(labels, bins=range(kmeans.n_clusters + 1))
        histograms.append(histogram)
    return histograms

def save_features_to_csv(features, labels, file_path):
    with open(file_path, 'w', newline='') as csvfile:
        writer = csv.writer(csvfile)
        writer.writerow(['name'] + [f'feature_{i}' for i in range(1, features.shape[1] + 1)])
        for label, descriptor in zip(labels, features):
            writer.writerow([label] + descriptor.tolist())
```

Figure 3.8: SAMPLE CODE FOR SIFT FEATURE EXTRACTION



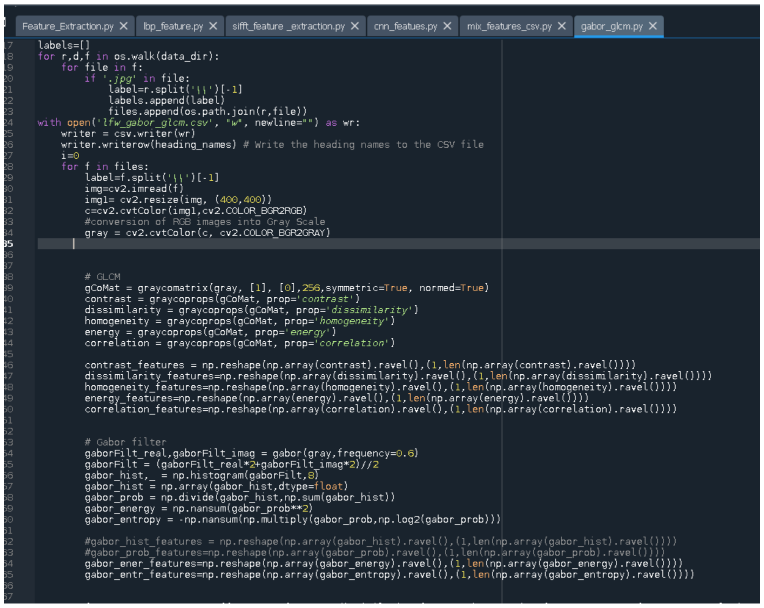
```python

for subfolder in os.listdir(dataset_folder):
    subfolder_path = os.path.join(dataset_folder, subfolder)
    if os.path.isdir(subfolder_path):
        label = subfolder
        for image_name in os.listdir(subfolder_path):
            image_path = os.path.join(subfolder_path, image_name)
            image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
            image = cv2.resize(image, (64, 64))  # Resize the image to your desired input size
            images.append(image)
            labels.append(label)

# Convert the lists to NumPy arrays
images = np.array(images)
labels = np.array(labels)

# Encode labels as integers
label_encoder = LabelEncoder()
labels = label_encoder.fit_transform(labels)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.3, random_state=0)

# Normalize the pixel values to [0, 1]
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0

# Reshape the data for CNN input (add channel dimension for grayscale images)
X_train = X_train.reshape(-1, 64, 64, 1)
X_test = X_test.reshape(-1, 64, 64, 1)


# Define the CNN model
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 1)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(len(np.unique(labels)), activation='softmax')  # Output layer with the number of classes
])

# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train the model
```

Figure 3.9: SAMPLE CODE FOR CNN FEATURE EXTRACTION



```
    for f in files:
        label=f.split('\\')[-1]
        img=cv2.imread(f)
        img1= cv2.resize(img, (400,400))
        c=cv2.cvtColor(img1,cv2.COLOR_BGR2RGB)
        #conversion of RGB images into Gray Scale
        gray = cv2.cvtColor(c, cv2.COLOR_BGR2GRAY)

        # LBP
        feat_lbp = local_binary_pattern(gray,5,2,'uniform')
        lbp_hist,_ = np.histogram(feat_lbp,8)
        lbp_hist = np.array(lbp_hist,dtype=float)
        lbp_prob = np.divide(lbp_hist,np.sum(lbp_hist))
        lbp_energy = np.nansum(lbp_prob**2)
        lbp_entropy = -np.nansum(np.multiply(lbp_prob,np.log2(lbp_prob)))

        #lbphist_features = np.reshape(np.array(lbp_hist).ravel(),(1,len(np.array(lbp_hist).ravel())))
        #lbpprob_features=np.reshape(np.array(lbp_prob).ravel(),(1,len(np.array(lbp_prob).ravel())))
        lbpenrgy_features=np.reshape(np.array(lbp_energy).ravel(),(1,len(np.array(lbp_energy).ravel())))
        lbpento_features=np.reshape(np.array(lbp_entropy).ravel(),(1,len(np.array(lbp_entropy).ravel())))


        # GLCM
        gCoMat = graycomatrix(gray, [1], [0],256,symmetric=True, normed=True)
        contrast = graycoprops(gCoMat, prop='contrast')
        dissimilarity = graycoprops(gCoMat, prop='dissimilarity')
        homogeneity = graycoprops(gCoMat, prop='homogeneity')
        energy = graycoprops(gCoMat, prop='energy')
        correlation = graycoprops(gCoMat, prop='correlation')

        contrast_features = np.reshape(np.array(contrast).ravel(),(1,len(np.array(contrast).ravel())))
        dissimilarity_features=np.reshape(np.array(dissimilarity).ravel(),(1,len(np.array(dissimilarity).ravel())))
        homogeneity_features=np.reshape(np.array(homogeneity).ravel(),(1,len(np.array(homogeneity).ravel())))
        energy_features=np.reshape(np.array(energy).ravel(),(1,len(np.array(energy).ravel())))
        correlation_features=np.reshape(np.array(correlation).ravel(),(1,len(np.array(correlation).ravel())))


        # Gabor filter
        gaborFilt_real,gaborFilt_imag = gabor(gray,frequency=0.6)
        gaborFilt = (gaborFilt_real*2+gaborFilt_imag*2)//2
        gabor_hist,_ = np.histogram(gaborFilt,8)
        gabor_hist = np.array(gabor_hist,dtype=float)
        gabor_prob = np.divide(gabor_hist,np.sum(gabor_hist))
        gabor_energy = np.nansum(gabor_prob**2)
        gabor_entropy = -np.nansum(np.multiply(gabor_prob,np.log2(gabor_prob)))

        #gabor_hist_features = np.reshape(np.array(gabor_hist).ravel(),(1,len(np.array(gabor_hist).ravel())))
        #gabor_prob_features=np.reshape(np.array(gabor_prob).ravel(),(1,len(np.array(gabor_prob).ravel())))
        gabor_ener_features=np.reshape(np.array(gabor_energy).ravel(),(1,len(np.array(gabor_energy).ravel())))
        gabor_entr_features=np.reshape(np.array(gabor_entropy).ravel(),(1,len(np.array(gabor_entropy).ravel())))


        features=np.concatenate((lbpenrgy_features,lbpento_features,contrast_features,dissimilarity_features,homogeneity_features,energy_featu
        ff=features[0].tolist()
        writer.writerow(ff+[labels[i]])
```

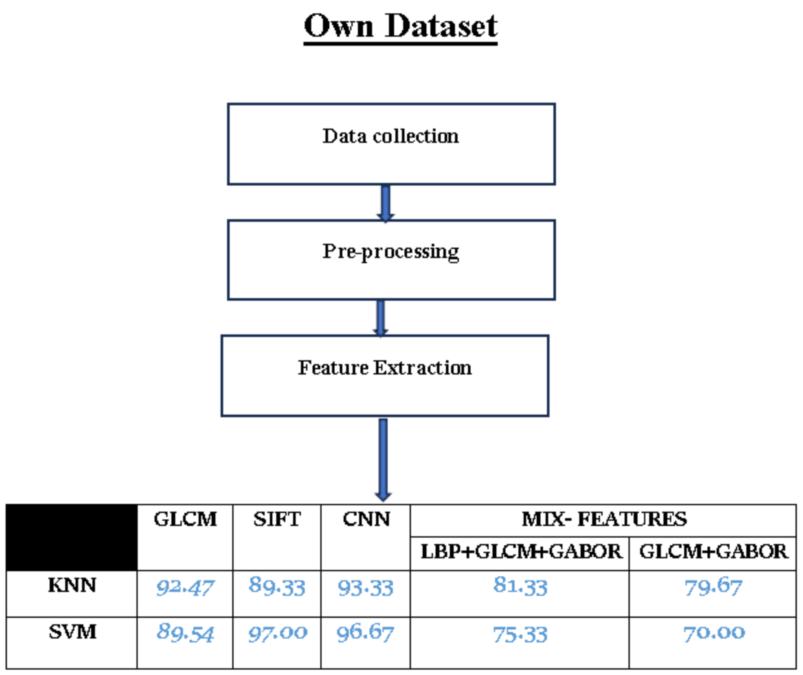
Figure 3.10: SAMPLE CODE FOR GLCM+LBP+GABOR FEATURE EXTRACTION



```
labels=[]
for r,d,f in os.walk(data_dir):
    for file in f:
        if '.jpg' in file:
            label=r.split('\\')[-1]
            labels.append(label)
            files.append(os.path.join(r,file))
with open('lfw_gabor_glcm.csv', "w", newline="") as wr:
    writer = csv.writer(wr)
    writer.writerow(heading_names) # Write the heading names to the CSV file
    i=0
    for f in files:
        label=f.split('\\')[-1]
        img=cv2.imread(f)
        img1= cv2.resize(img, (400,400))
        c=cv2.cvtColor(img1,cv2.COLOR_BGR2RGB)
        #conversion of RGB images into Gray Scale
        gray = cv2.cvtColor(c, cv2.COLOR_BGR2GRAY)



        # GLCM
        gCoMat = graycomatrix(gray, [1], [0],256,symmetric=True, normed=True)
        contrast = graycoprops(gCoMat, prop='contrast')
        dissimilarity = graycoprops(gCoMat, prop='dissimilarity')
        homogeneity = graycoprops(gCoMat, prop='homogeneity')
        energy = graycoprops(gCoMat, prop='energy')
        correlation = graycoprops(gCoMat, prop='correlation')

        contrast_features = np.reshape(np.array(contrast).ravel(),(1,len(np.array(contrast).ravel())))
        dissimilarity_features=np.reshape(np.array(dissimilarity).ravel(),(1,len(np.array(dissimilarity).ravel())))
        homogeneity_features=np.reshape(np.array(homogeneity).ravel(),(1,len(np.array(homogeneity).ravel())))
        energy_features=np.reshape(np.array(energy).ravel(),(1,len(np.array(energy).ravel())))
        correlation_features=np.reshape(np.array(correlation).ravel(),(1,len(np.array(correlation).ravel())))


        # Gabor filter
        gaborFilt_real,gaborFilt_imag = gabor(gray,frequency=0.6)
        gaborFilt = (gaborFilt_real*2+gaborFilt_imag*2)//2
        gabor_hist,_ = np.histogram(gaborFilt,8)
        gabor_hist = np.array(gabor_hist,dtype=float)
        gabor_prob = np.divide(gabor_hist,np.sum(gabor_hist))
        gabor_energy = np.nansum(gabor_prob**2)
        gabor_entropy = -np.nansum(np.multiply(gabor_prob,np.log2(gabor_prob)))

        #gabor_hist_features = np.reshape(np.array(gabor_hist).ravel(),(1,len(np.array(gabor_hist).ravel())))
        #gabor_prob_features=np.reshape(np.array(gabor_prob).ravel(),(1,len(np.array(gabor_prob).ravel())))
        gabor_ener_features=np.reshape(np.array(gabor_energy).ravel(),(1,len(np.array(gabor_energy).ravel())))
        gabor_entr_features=np.reshape(np.array(gabor_entropy).ravel(),(1,len(np.array(gabor_entropy).ravel())))
```

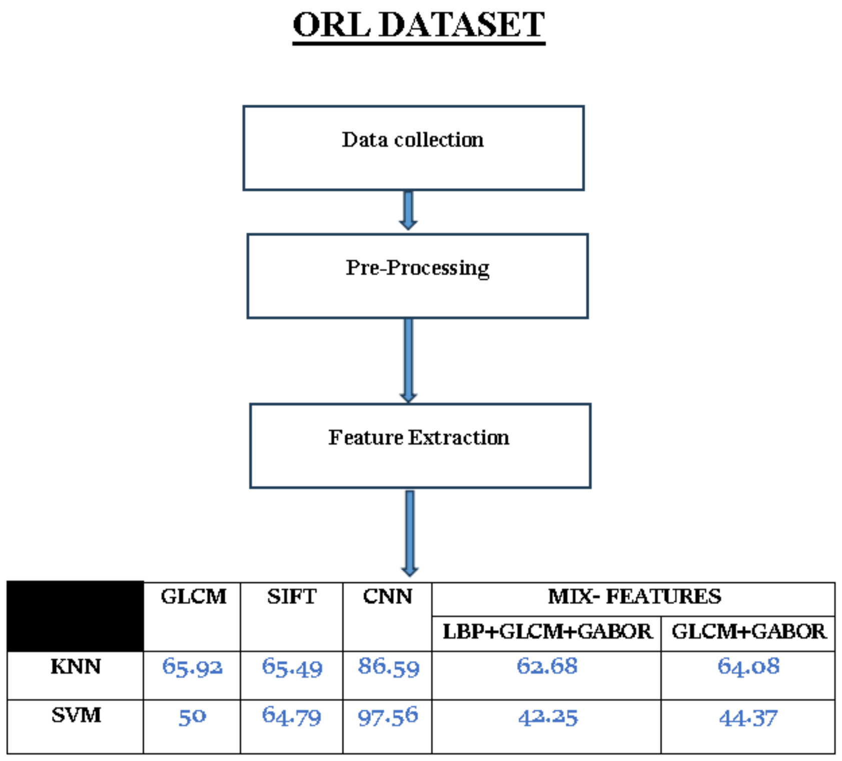
Figure 3.11: SAMPLE CODE FOR GABOR+GLCM FEATURE EXTRACTION

-Matching/Recognition:This section consist of recognising/identification of an image. In this section we haave to read the csv file then have to check if there is any missing value is there or null value is there if it is there then we need to remove it after doing that we need to check if the csv file have any string value and is not require then we need to drop that column and if its require we need to do lable encoding to change the string to numeric value.Next step is to split the dataset into trainig and testing, after this i have used classifiers like KNN ,Random Forest ,SVM to predict the accuracy.



	GLCM	SIFT	CNN	MIX- FEATURES	
				LBP+GLCM+GABOR	GLCM+GABOR
KNN	92.47	89.33	93.33	81.33	79.67
SVM	89.54	97.00	96.67	75.33	70.00

Figure 3.12: OUTPUT of OWN DATASET



	GLCM	SIFT	CNN	MIX- FEATURES	
				LBP+GLCM+GABOR	GLCM+GABOR
KNN	65.92	65.49	86.59	62.68	64.08
SVM	50	64.79	97.56	42.25	44.37

Figure 3.13: OUTPUT of ORL DATASET

Chapter 4

COMPARITIVE ANALYSIS

4.1 OWN DATASET VS STANDARD DATASET(ORL)

This section consists of the comparative analysis of my dataset and standard dataset i.e, ORL dataset.
According to my study and analysis of the outputs given by own dataset and ORL dataset . I received a better result in my dataset. When with GLCM I applied classifiers like KNN and SVM in own dataset I received the accuracy 92.47 and 89.54 percentage respectively but when i applied classifiers like KNN and SVM in ORL dataset, I received 65.92 and 50.00 respectively , with adjusting the random state as 8 .
When with SIFT I applied classifiers like KNN, SVM in own dataset I received the accuracy 89.33 and 97.00 percentage respectively but when I applied classifiers like KNN and SVM in ORL dataset, I received 65.49 and 64.79 percentage with adjusting the random state as 8.
Similarly, when I applied CNN to both I received a very good accuracy. For my dataset, when i applied classifiers like KNN and SVM with adjusting the random state as 20 , I received 93.33 and 96.67 percentage and with ORL 86.59 and 97.56 percentage receptively. While performing on mixing features like LBP,GLCM and GABOR and using the same classifiers KNN and SVM for my dataset I received 81.33and 75.33 percentage and with ORL I got 62.68 and 42.25 percentage. And in list I mixed two algorithms features like GLCM and GABOR

and using the same classifiers KNN and SVM for my dataset I received 79.67 and 70 percentage and with ORL I got 64.08 and 44.37 percentage. In overall , I can conclude that my data-set is performed very well with my combinations but ORL dataset failed to achieve the accuracy they claimed.

4.2 GRAPH REPRESENTATION

Here is a comparative bar graph presentation of the above comparison.

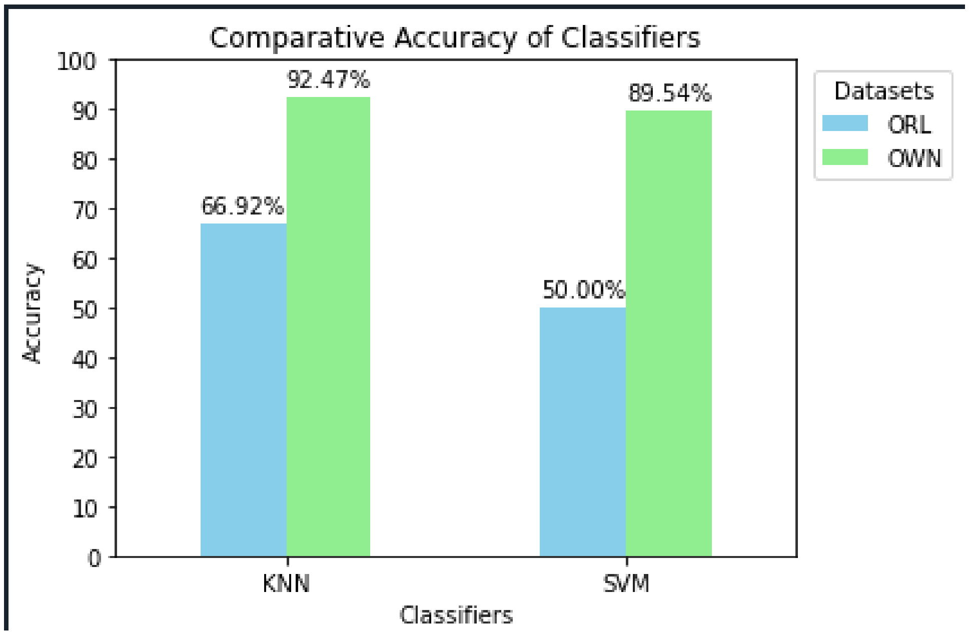


Figure 4.1: GLCM WITH KNN AND SVM

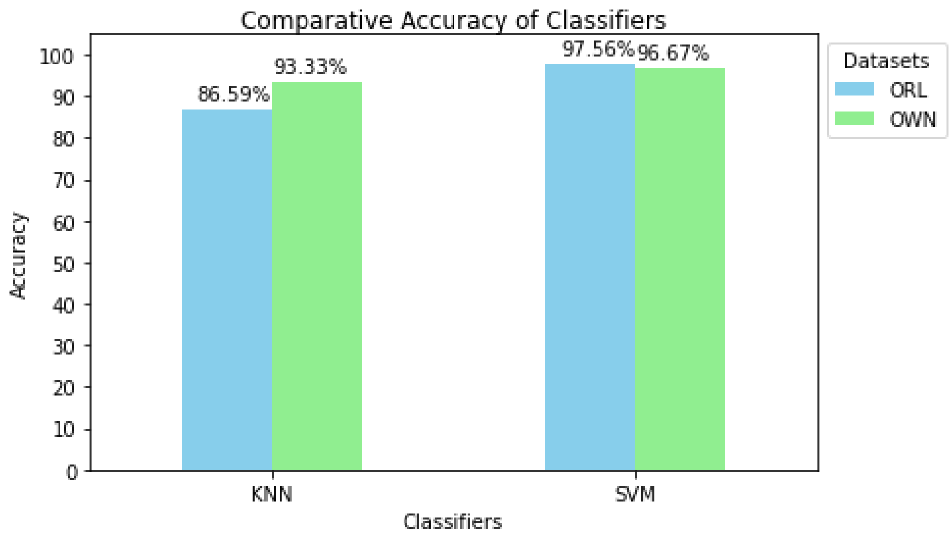


Figure 4.2: CNN WITH KNN AND SVM

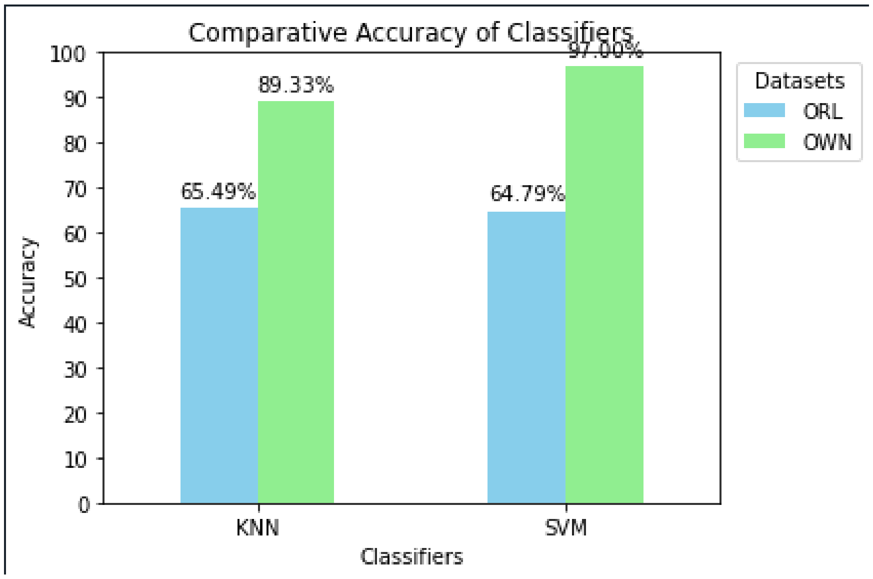


Figure 4.3: SIFT WITH KNN AND SVM

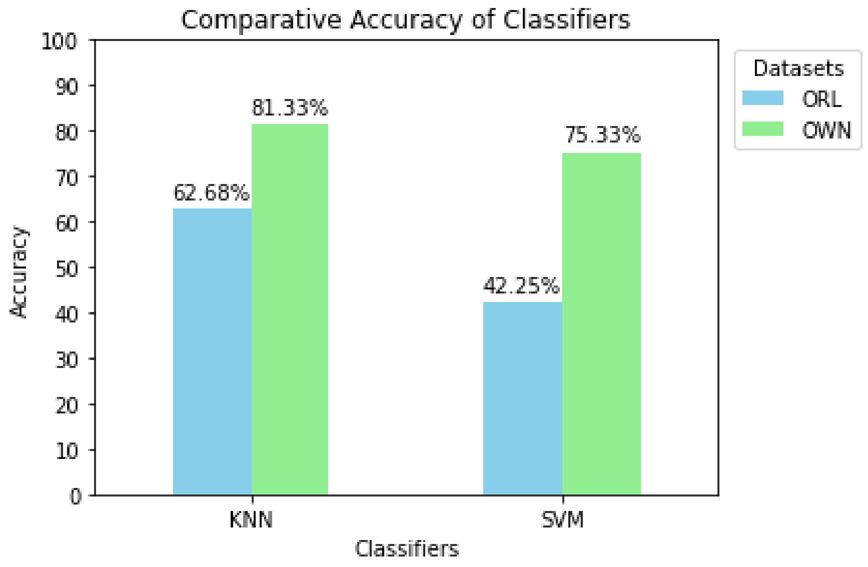


Figure 4.4: LBP+GLCM+GABOR WITH KNN AND SVM

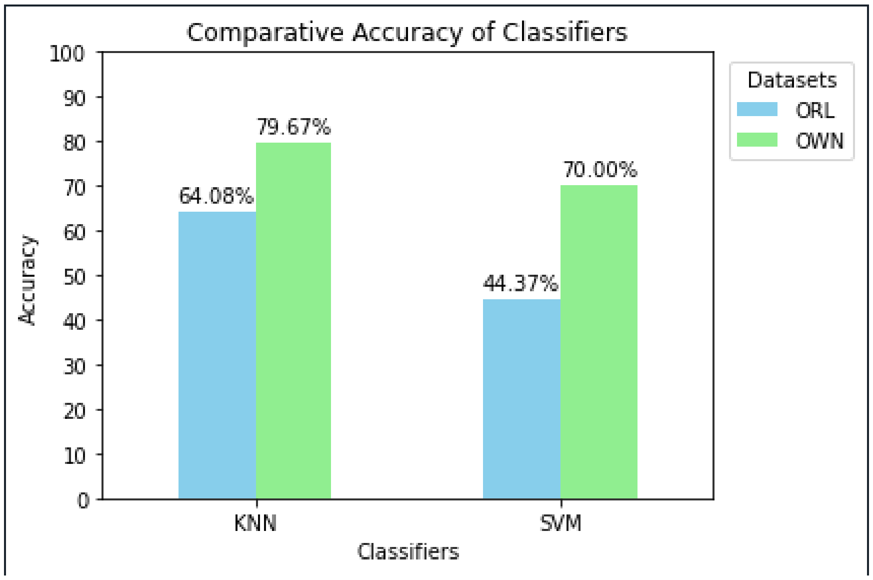


Figure 4.5: GLCM+GABOR WITH KNN AND SVM

Chapter 5

3D MESH CREATION

A 2D face Recognition System have few flaws that 2D can not overcome like ageing,lighting condition,rotations, depth information etc, to over come this a 3D face image recognition comes into play as the three-dimensional face information conforms to the three-dimensional characteristics of human vision, and the face depth information. In this technique 3-D sensors are used to capture information about the shape and size of a face. This information is then used to identify distinct features on the surface of a faces, such as the contour of the eye sockets, nose, and chin etc.[10]

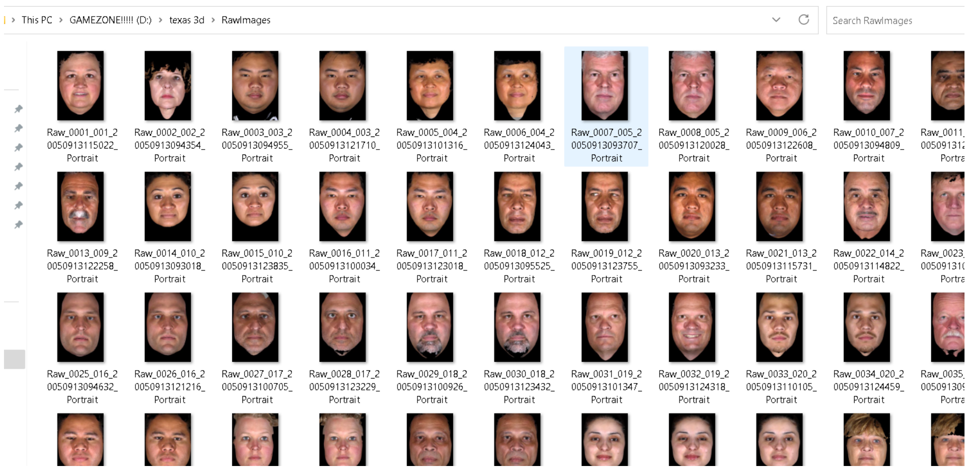
My aim is to construct a 3d Mesh from a 2d images .I have collected the Texas 3d Dataset on which i will be performing my experiment.

5.1 METHODOLOGY:

In this section i will be mentioning the way I constructed the face mesh.

***Loading dataset:** It is the first step , in this step im fetching all the images from the Texas-3d dataset. There are total 1150 face images .

Sample image is shown as Figure 5.1

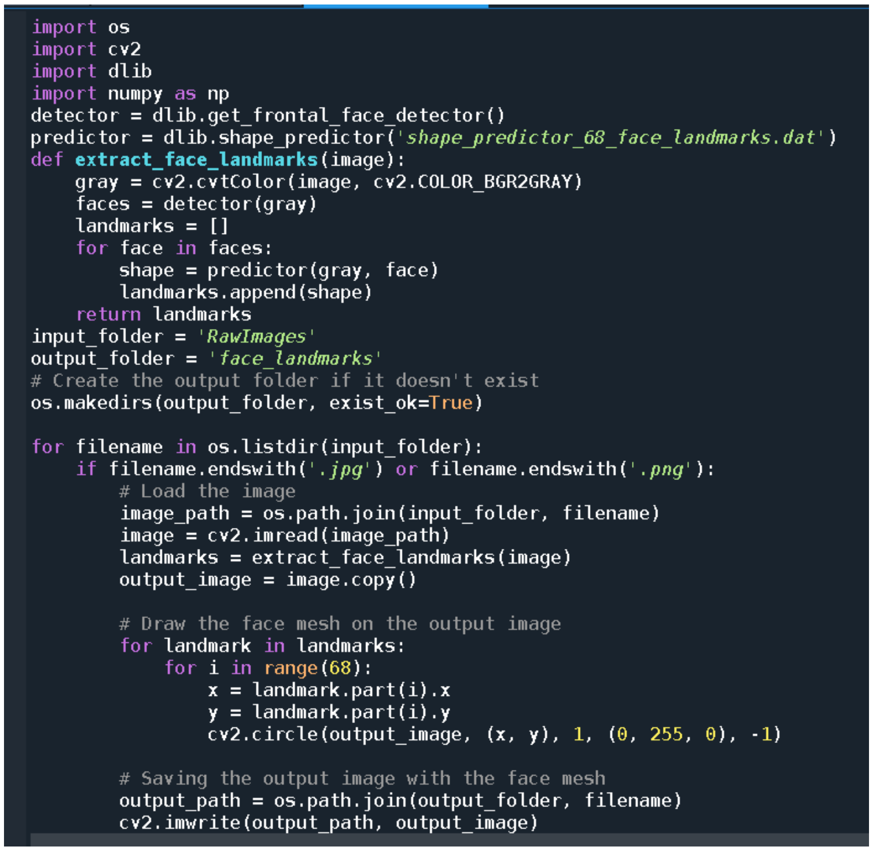


This PC › GAMEZONE!!!!! (D:) › texas 3d › RawImages
Search RawImages
Raw_0001_001_2 0050913115022_ Portrait
Raw_0002_002_2 0050913094354_ Portrait
Raw_0003_003_2 0050913094955_ Portrait
Raw_0004_003_2 0050913121710_ Portrait
Raw_0005_004_2 0050913101316_ Portrait
Raw_0006_004_2 0050913124043_ Portrait
Raw_0007_005_2 0050913093707_ Portrait
Raw_0008_005_2 0050913120028_ Portrait
Raw_0009_006_2 0050913122608_ Portrait
Raw_0010_007_2 0050913094809_ Portrait
Raw_0013_009_2 0050913122258_ Portrait
Raw_0014_010_2 0050913093018_ Portrait
Raw_0015_010_2 0050913123835_ Portrait
Raw_0016_011_2 0050913100034_ Portrait
Raw_0017_011_2 0050913123018_ Portrait
Raw_0018_012_2 0050913095525_ Portrait
Raw_0019_012_2 0050913123755_ Portrait
Raw_0020_013_2 0050913093233_ Portrait
Raw_0021_013_2 0050913115731_ Portrait
Raw_0022_014_2 0050913114822_ Portrait
Raw_0025_016_2 0050913094632_ Portrait
Raw_0026_016_2 0050913121216_ Portrait
Raw_0027_017_2 0050913100705_ Portrait
Raw_0028_017_2 0050913123229_ Portrait
Raw_0029_018_2 0050913100926_ Portrait
Raw_0030_018_2 0050913123432_ Portrait
Raw_0031_019_2 0050913101347_ Portrait
Raw_0032_019_2 0050913124318_ Portrait
Raw_0033_020_2 0050913110105_ Portrait
Raw_0034_020_2 0050913124459_ Portrait

Figure 5.1: TEXAS-3D-DATASET

***Landmark detection:** In this section i have coded for to extract the land mark from the images . here i have used opencv and dlib lib for extracting the face area with shape_predictor_68_face_landmarks.dat file for a proper landmark detection. It will extract the landmarks like eyes , nose , mouth ,eyebrows and edges of the face.

Sample code is shown as Figure 5.2



```python
import os
import cv2
import dlib
import numpy as np
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
def extract_face_landmarks(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    faces = detector(gray)
    landmarks = []
    for face in faces:
        shape = predictor(gray, face)
        landmarks.append(shape)
    return landmarks
input_folder = 'RawImages'
output_folder = 'face_landmarks'
# Create the output folder if it doesn't exist
os.makedirs(output_folder, exist_ok=True)

for filename in os.listdir(input_folder):
    if filename.endswith('.jpg') or filename.endswith('.png'):
        # Load the image
        image_path = os.path.join(input_folder, filename)
        image = cv2.imread(image_path)
        landmarks = extract_face_landmarks(image)
        output_image = image.copy()

        # Draw the face mesh on the output image
        for landmark in landmarks:
            for i in range(68):
                x = landmark.part(i).x
                y = landmark.part(i).y
                cv2.circle(output_image, (x, y), 1, (0, 255, 0), -1)

        # Saving the output image with the face mesh
        output_path = os.path.join(output_folder, filename)
        cv2.imwrite(output_path, output_image)
```

Figure 5.2: SAMPLE SOURCE-CODE LANDMARK TO DETECT FEATURES

Sample image is shown as Figure 5.3

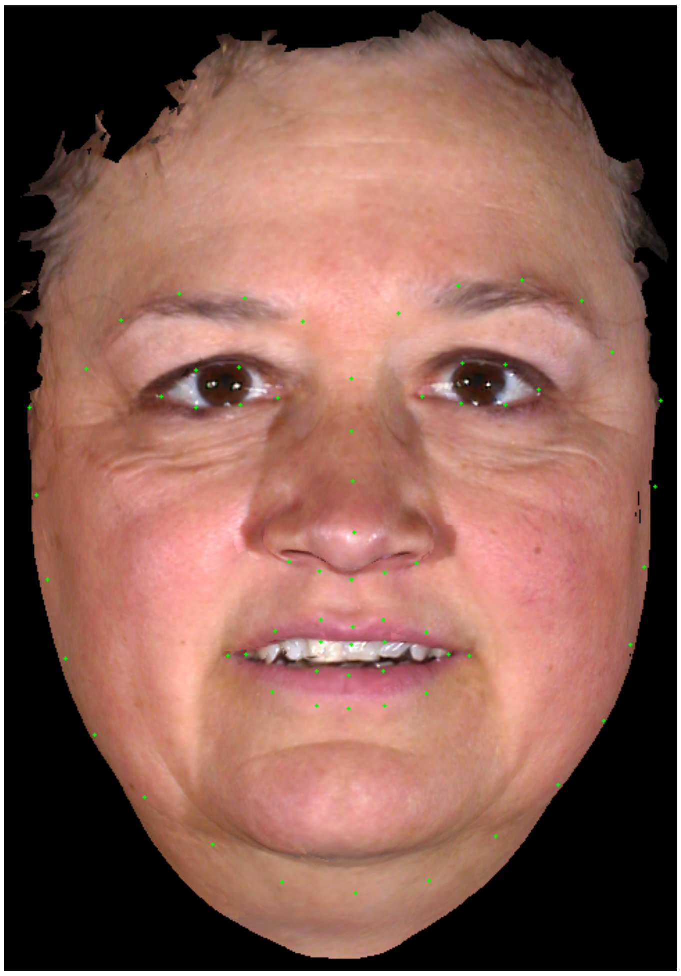
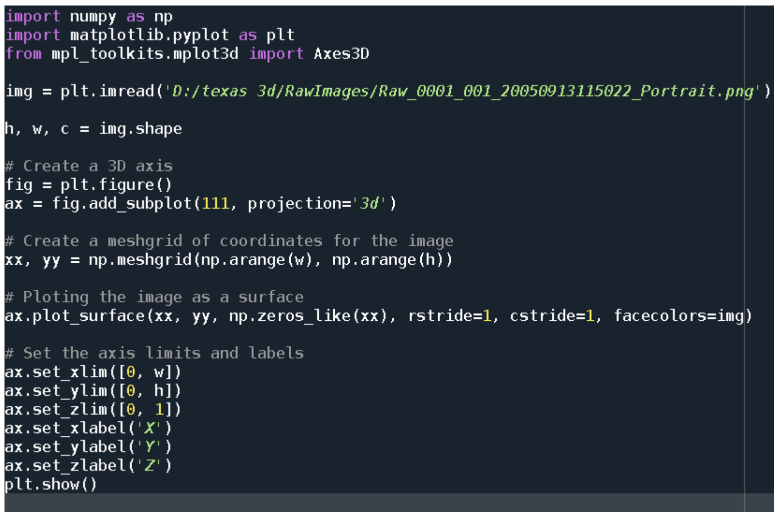


Figure 5.3: LANDMARK TO DETECT FEATURES

***3D plot:** Here this section makes a 3d visualization plot of each images. In this we place the 2D image in the 3D axis to visualize the image in 3D plane

Sample code is shown as Figure 5.4



```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

img = plt.imread('D:/texas 3d/RawImages/Raw_0001_001_20050913115022_Portrait.png')

h, w, c = img.shape

# Create a 3D axis
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Create a meshgrid of coordinates for the image
xx, yy = np.meshgrid(np.arange(w), np.arange(h))

# Ploting the image as a surface
ax.plot_surface(xx, yy, np.zeros_like(xx), rstride=1, cstride=1, facecolors=img)

# Set the axis limits and labels
ax.set_xlim([0, w])
ax.set_ylim([0, h])
ax.set_zlim([0, 1])
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.show()
```

Figure 5.4: SAMPLE SOURCE-CODE 3d PLOT

Sample image is shown as Figure 5.5

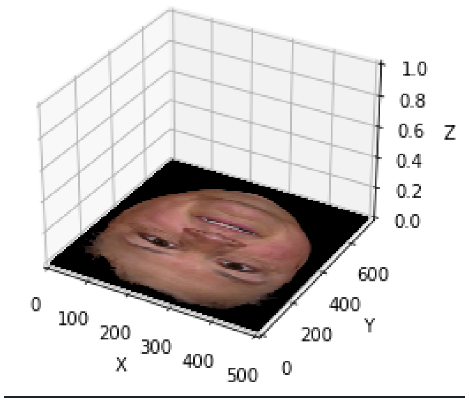
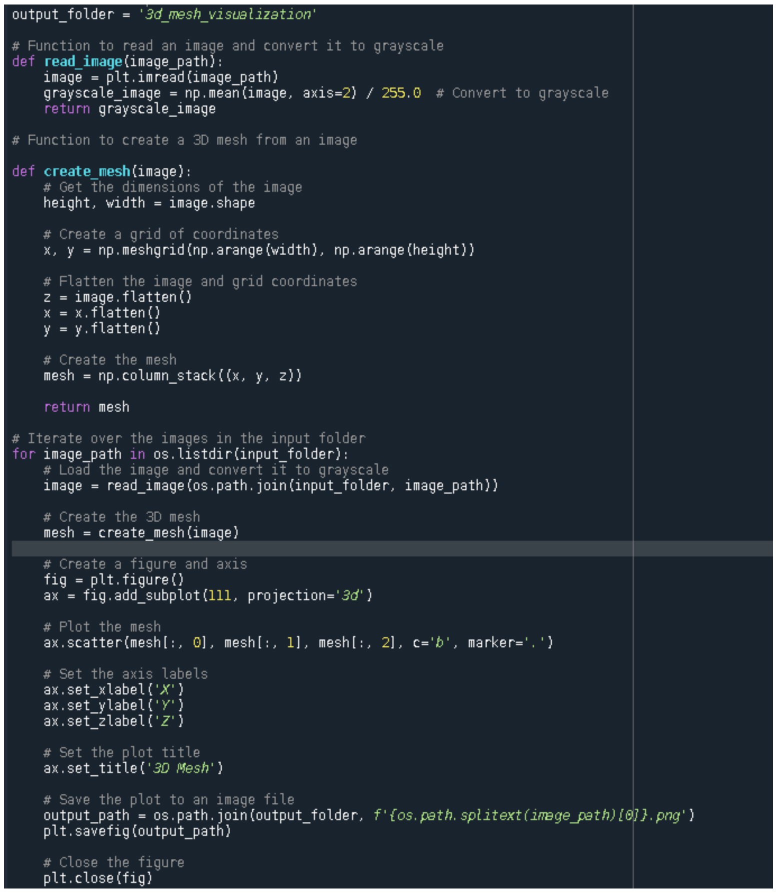


Figure 5.5: 3D-PLOT

***3D mesh visualization:** Through this section i have extracted the 3d mesh visualization. 3D mesh from an image.I have defined a create_mesh function which flattens the image and grid coordinates (x, y) into 1D arrays and stacks them together to form the mesh.The mesh is plotted using the scatter function in a 3D plot.

Sample code is shown as Figure 5.6



```
output_folder = '3d_mesh_visualization'

# Function to read an image and convert it to grayscale
def read_image(image_path):
    image = plt.imread(image_path)
    grayscale_image = np.mean(image, axis=2) / 255.0  # Convert to grayscale
    return grayscale_image

# Function to create a 3D mesh from an image

def create_mesh(image):
    # Get the dimensions of the image
    height, width = image.shape

    # Create a grid of coordinates
    x, y = np.meshgrid(np.arange(width), np.arange(height))

    # Flatten the image and grid coordinates
    z = image.flatten()
    x = x.flatten()
    y = y.flatten()

    # Create the mesh
    mesh = np.column_stack((x, y, z))

    return mesh

# Iterate over the images in the input folder
for image_path in os.listdir(input_folder):
    # Load the image and convert it to grayscale
    image = read_image(os.path.join(input_folder, image_path))

    # Create the 3D mesh
    mesh = create_mesh(image)

    # Create a figure and axis
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')

    # Plot the mesh
    ax.scatter(mesh[:, 0], mesh[:, 1], mesh[:, 2], c='b', marker='.')

    # Set the axis labels
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')

    # Set the plot title
    ax.set_title('3D Mesh')

    # Save the plot to an image file
    output_path = os.path.join(output_folder, f'{os.path.splitext(image_path)[0]}.png')
    plt.savefig(output_path)

    # Close the figure
    plt.close(fig)
```

Figure 5.6: SAMPLE SOURCE-CODE 3d MESH CONSTRUCTION VISUALIZATION

Sample image is shown as Figure 5.7

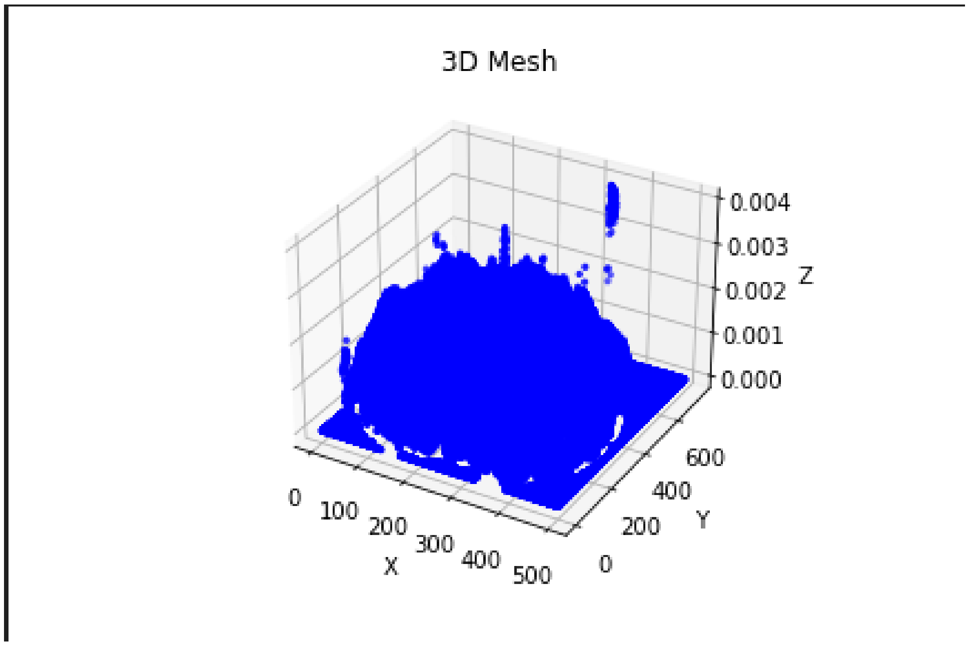
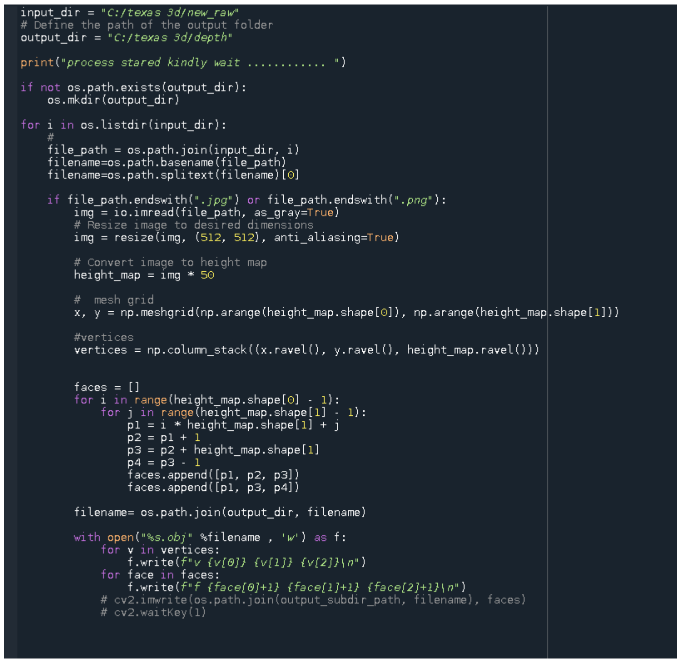


Figure 5.7: 3D-MESH VISUALIZATION ON 3D-PLOT

*** 3D mesh object creation:**This section comprises of how mesh object will be created and stored as obj file.First, the faces list is initialized to an empty list. The outer loop iterates over the height of the height_map array, excluding the last row (height_map.shape[0] - 1). The inner loop iterates over the width of the height_map array, excluding the last column (height_map.shape[1] - 1). The variable p1 is calculated as the current row (i) multiplied by the width of the height_map plus the current column (j). This gives the index of the top-left vertex of a face. p2 is calculated as p1 + 1, representing the top-right vertex of the face. p3 is calculated as p2 + height_map.shape[1], representing the bottom-right vertex of the face. p4 is calculated as p3 - 1, representing the bottom-left vertex of the face. Two faces are appended to the faces list: [p1, p2, p3] and [p1, p3, p4]. This process is repeated for all valid combinations of rows and columns in the height_map array, creating a set of faces for the 3D-object.[11]

Sample code is shown as Figure 5.8



```python
input_dir = "C:/texas 3d/new_raw"
# Define the path of the output folder
output_dir = "C:/texas 3d/depth"

print("process stared kindly wait ............ ")

if not os.path.exists(output_dir):
    os.mkdir(output_dir)

for i in os.listdir(input_dir):
    #
    file_path = os.path.join(input_dir, i)
    filename=os.path.basename(file_path)
    filename=os.path.splitext(filename)[0]

    if file_path.endswith(".jpg") or file_path.endswith(".png"):
        img = io.imread(file_path, as_gray=True)
        # Resize image to desired dimensions
        img = resize(img, (512, 512), anti_aliasing=True)

        # Convert image to height map
        height_map = img * 50

        #  mesh grid
        x, y = np.meshgrid(np.arange(height_map.shape[0]), np.arange(height_map.shape[1]))

        #vertices
        vertices = np.column_stack((x.ravel(), y.ravel(), height_map.ravel()))


        faces = []
        for i in range(height_map.shape[0] - 1):
            for j in range(height_map.shape[1] - 1):
                p1 = i * height_map.shape[1] + j
                p2 = p1 + 1
                p3 = p2 + height_map.shape[1]
                p4 = p3 - 1
                faces.append([p1, p2, p3])
                faces.append([p1, p3, p4])

        filename= os.path.join(output_dir, filename)

        with open("%s.obj" %filename , 'w') as f:
            for v in vertices:
                f.write(f"v {v[0]} {v[1]} {v[2]}\n")
            for face in faces:
                f.write(f"f {face[0]+1} {face[1]+1} {face[2]+1}\n")
            # cv2.imwrite(os.path.join(output_subdir_path, filename), faces)
            # cv2.waitKey(1)
```

Figure 5.8: SAMPLE SOURCE-CODE 3d MESH OBJECT CONSTRUCTION

Sample image is shown as Figure 5.9

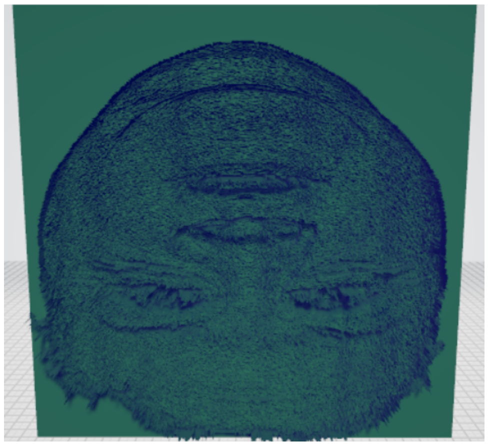


Figure 5.9: 3D MESH

Chapter 6

CONCLUSION

A fairly sophisticated technology that has received a lot of attention recently is a face-recognition system. In other applications, including as surveillance and identity-verification, it has also demonstrated considerable promise. To recognize and distinguish human faces in photos or videos, this system uses machine learning approaches and optimized algorithms.
In conclusion, facial recognition has the ability to completely change how we engage with technology and offers a wide range of advantages. They enable for seamless authentication and quick identification procedures, as well as improved security and convenience. These technologies can precisely match and authenticate people in real-time by analyzing facial patterns and its traits.Face recognition technologies are also widely used in law enforcement to identify suspects or missing people from surveillance films. They also offer practical applications in personal devices, such as unlocking smartphones or laptops with facial-recognition.

REFERENCES

[1] AS Tolba, AH El-Baz, and AA El-Harby. Face recognition: A literature review. *International Journal of Signal Processing*, 2(2):88–103, 2006.

[2] Lixiang Li, Xiaohui Mu, Siying Li, and Haipeng Peng. A review of face recognition technology. *IEEE access*, 8:139110–139120, 2020.

[3] Peng Lu, Baoye Song, and Lin Xu. Human face recognition based on convolutional neural network and augmented dataset. *Systems Science & Control Engineering*, 9(sup2):29–37, 2021.

[4] Zhi Liu, Dongmei Jiang, Yujun Li, Yankun Cao, Mingyu Wang, and Yong Xu. Automatic face recognition based on sparse representation and extended transfer learning. *Ieee Access*, 7:2387–2395, 2018.

[5] Gurlove Singh and Amit Kumar Goel. Face detection and recognition system using digital image processing. *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, pages 348–352, 2020.

[6] Berk Gokberk, Albert Salah, Lale Akarun, Remy Etheve, Daniel Riccio, and Jean-Luc Dugelay. *3D Face Recognition*, pages 263–295. 04 2009.

[7] Sokyna Qatawneh, Stanley Ipson, Rami Qahwaji, and Hassan Ugail. 3d face recognition based on machine learning. *Proceedings of the 8th IASTED International Conference on Visualization, Imaging, and Image Processing, VIIP 2008*, 01 2008.

[8] Nikolai Chinaev, Alexander Chigorin, and Ivan Laptev. Mobileface: 3d face reconstruction with efficient cnn regression. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[9] Wamidh K Mutlag, Shaker K Ali, Zahoor M Aydam, and Bahaa H Taher. Feature extraction methods: a review. In *Journal of Physics: Conference Series*, volume 1591, page 012028. IOP Publishing, 2020.

[10] Andrea F Abate, Michele Nappi, Daniel Riccio, and Gabriele Sabatino. 2d and 3d face recognition: A survey. *Pattern recognition letters*, 28(14):1885–1906, 2007.

[11] Feng Liu, Dan Zeng, Qijun Zhao, and Xiaoming Liu. Joint face alignment and 3d face reconstruction. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14*, pages 545–560. Springer, 2016.