RECOGNITION AND COMPRESSION OF DIGITAL IMAGE USING MACHINE LEARNING

A major project report submitted for the partial fulfillment of the requirements for the Degree

of

Bachelor of Technology in Computer Science & Engineering

By

Ankita Pradhan	**1801105071**
Balakram Tudu	**1801105113**
Jyotirmaya Pati	**1801105208**
Kajalrekha Sethi	**1801105213**
Sachikanta Raul	**1801105372**

Under the Guidance of

Dr. Biswanath Sethi

and

Mr. Binaya Kumar Patra

Dept of CSEA, IGIT Sarang

DEPRTMENT OF COMPUTER SCIENCE ENGINEERING &APPLICATIONS

INDIRA GANDHI INSTITUTE OF TECHNOLOGY, SARANG

DHENKANAL, ODISHA-759146

DECLARATION BY THE SCHOLARS

We hereby declare that the project report entitled ***"Recognition and Compression of Digital Image using Machine Learning"*** *under BPUT, ROURKELA, Odisha at* ***Indira Gandhi Institute Technology, Sarang, Dhenkanal, Odisha*** *is an authentic record of our work carried out under the supervision* ***Dr. Biswanath Sethi and Mr. Binaya Kumar Patra, Dept. of CSEA, IGIT, Sarang.*** *We have not submitted this project report elsewhere for any other degree or diploma.*

Ankita Pradhan 1801105071
Balakram Tudu 1801105113
Jyotirmaya Pati 1801105208
Kajalrekha Sethi 1801105213
Sachikanta Raul 1801105372

Department of CSEA,
IGIT, Sarang

INDIRA GANDHI INSTITUTE OF TECHNOLOGY
SARANG

Certificate

*This is to certify that this project entitled "***Recognition and Compression of Digital Image using Machine Learning***" submitted by* ***Ankita Pradhan, Regd. No. 1801105071, Balakram Tudu, Regd. No. 1801105113, Jyotirmaya Pati, Regd. No. 1801105208, Kajalrekha Sethi, Regd. No. 1801105213 and Sachikanta Raul Regd No. 1801105372*** *of Computer Science Engineering & Applications Department, Indira Gandhi Institute of Technology, Sarang for the partial fulfillment of the requirements for the award of Bachelor of Technology (Computer Science & Engineering) Degree of BPUT, Odisha, is a record of students own study carried under our supervision & guidance.*

This report has not been submitted to any other university or institution for the award of any degree.

Date:

Dr. Biswanath Sethi ,
Asst. professor

(Dr. S. Mishra)
Professor & Head,
Dept. of CSEA, IGIT, Sarang
Dhenkanal, Odisha

Mr. Binaya Kumar Patra,
Asst. professor

Signature of
External Examiner

ACKNOWLEDGEMENTS

It is a great pleasure and privilege to express our profound sense of gratitude to **Dr. Biswanath Sethi and Mr. Binaya Kumar Patra**, Dept. of CSEA, IGIT, Sarang, for their suggestions, motivation and support during the major project preparation and keen personal interest throughout the progress of our course work.

We would like to express my sincere thanks to **Prof. S. Mishra**, HEAD, Dept. of CSEA, IGIT, Sarang, for her suggestions, motivation and support during the major project preparation.

We have been fortunate enough to have all support motivation from our parents. We sincerely thanks to the other faculties of the department of CSEA, IGIT, Sarang for their positive and constructive suggestions and kind cooperation during major project preparation and presentation.

.

Ankita Pradhan

Balakram Tudu

Jyotirmaya Pati

Kajalrekha Sethi

Sachikanta Raul

LIST OF TABLES

CONTENTS **page**

ABSTRACT

The face is one of the easiest ways to distinguish the individual identity of each other. Face recognition is a personal identification system that uses personal characteristics of a person to identify the person's identity. Human face recognition procedure basically consists of two phases, namely face detection, where this process takes place very rapidly in humans, except under conditions where the object is located at a short distance away, the next is the introduction, which recognize a face as individuals. Stage is then replicated and developed as a model for facial image recognition (face recognition) is one of the much-studied biometrics technology and developed by experts. There are two kinds of methods that are currently popular in developed face recognition pattern namely, Eigenface method and Fisher face method. Facial image recognition Eigenface method is based on the reduction of face-dimensional space using Principal Component Analysis (PCA) for facial features. The main purpose of the use of PCA on face recognition using Eigen faces was formed (face space) by finding the eigenvector corresponding to the largest eigenvalue of the face image. The area of this project face detection system with face recognition is Image processing.

Image compression is considered as application performed for compression of data in digital format images. Digital images are comprised with large amount of information that requires bigger bandwidth. The techniques of image compression can be generally categorized into two types: lossless & lossy technique. DCT (discrete cosine transform) can also be used for compressing an image and also approaches like Huffman encoding, quantization & such steps are required for compression of images with JPEG format. The format of JPEG can be used for both of the RGB (colored) & YUV (gray scale) images. But here our main concentration is over decompression & compression of gray scale images. Here 2D-DCT can be used for transformation of a 8x8 matrix of images to an elementary frequency elements. The DCT is considered to be a mathematical function which will transform an image of digital format from spatial to frequency domain. It is very much easy to implement Huffman encoding & decoding for minimizing the complexity of memory. In this proposed technique, the analog image pixels are transformed to discrete image pixel, and therefore compression is performed. On the receiving side, the pixels are decompressed for obtaining the actual image. The PSNR is computed for analyzing the quality of image.

Keyword: face detection, Eigen face, PCA, JPEG, DCT, IDCT, Huffman Image Technique, PSNR

LIST OF FIGURES

LIST OF ABBREVIATIONS

AD	Average Difference
BPN	Back propagation Neural Network
CCITT	Consultative Committee for International Telephony and Telegraphy
CR	Compression Ratio.
DCT	Discrete cosine transforms.
DLib	Library for Machine Learning.
IDCT	Inverse Discrete Cosine Transform
JPEG	Joint Photographic Experts Group.
LDA	Local Delivery Agent.
MD	Maximum Difference
MPEG	Moving Picture Experts Group.
MSE	Mean Square Error
NAE	Normalized Absolute Error
NK	Normalized Cross Correlation
OpenCV	Open-Source Computer Vision Library.
PCA	Principal Component Analysis.
PSNR	Peak Signal to Noise Ratio
SC	Structural Content
TIFF	Tag Image File Format
XML	Extensible Markup Language.

CHAPTER-1

INTRODUCTION

Face recognition is the task of identifying an already detected object as a known or unknown face. Often the problem of face recognition is confused with the problem of face detection Face Recognition on the other hand is to decide if the "face" is someone known, or unknown, using for this purpose a database of faces in order to validate this input face.

1.1 FACE RECOGNIZATION:

DIFFERENT APPROACHES OF FACE RECOGNITION:

There are two predominant approaches to the face recognition problem: Geometric (feature based) and photometric (view based). As researcher interest in face recognition continued, many different algorithms were developed, three of which have been well studied in face recognition literature.

Recognition algorithms can be divided into two main approaches:

1.1.1 Geometric: Is based on geometrical relationship between facial landmarks, or in other words the spatial configuration of facial features. That means that the main geometrical features of the face such as the eyes, nose and mouth are first located and then faces are classified on the basis of various geometrical distances and angles between features. (Figure 3)

1.1.2 Photometric stereo: Used to recover the shape of an object from a number of images taken under different lighting conditions. The shape of the recovered object is defined by a gradient map, which is made up of an array of surface normal (Zhao and Chellappa, 2006) (Figure 2)

Popular recognition algorithms include:

1. Principal Component Analysis using Eigenfaces, (PCA)
2. Linear Discriminate Analysis,
3. Elastic Bunch Graph Matching using the Fisher face algorithm,

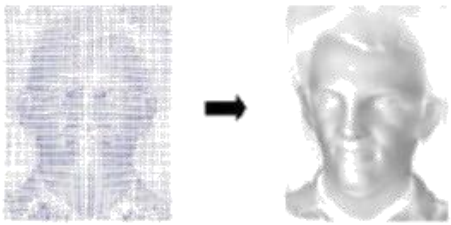


Figure 2 -Photometric stereo image.

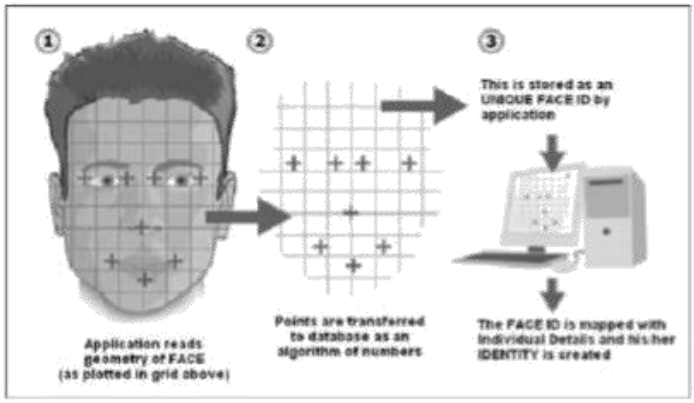


Figure 3 - Geometric facial recognition.

1.2 FACE DETECTION:

Face detection involves separating image windows into two classes; one containing faces (darning the background (clutter). It is difficult because although commonalities exist between faces, they can vary considerably in terms of age, skin colour and facial expression. The problem is further complicated by differing lighting conditions, image qualities and geometries, as well as the possibility of partial occlusion and disguise. An ideal face detector would therefore be able to detect the presence of any face under any set of lighting conditions, upon any background. The face detection task can be broken down into two steps. The first step is a classification task that takes some arbitrary image as input and outputs a binary value of yes or no, indicating whether there are any faces present in the image. The second step is the face localization task that aims to take an image as input and output the location of any face or faces within that image as some bounding box with (x, y, width, height).

The face detection system can be divided into the following steps: -

1. 2.1 Pre-Processing: To reduce the variability in the faces, the images are processed before they are fed into the network. All positive examples that is the face images are obtained by cropping

images with frontal faces to include only the front view. All the cropped images are then corrected for lighting through standard algorithms.

1.2.2 Classification: Neural networks are implemented to classify the images as faces or nonfaces by training on these examples. We use both our implementation of the neural network and the neural network toolbox for this task. Different network configurations are experimented with to optimize the results.

1.2.3 Localization: The trained neural network is then used to search for faces in an image and if present localize them in a bounding box. Various Feature of Face on which the work has done on:-Position Scale Orientation Illumination

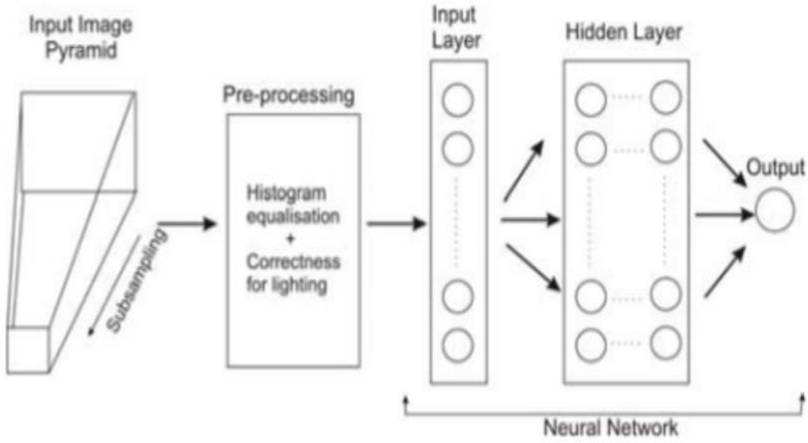


Fig: Face detection algorithm

1.2.4 Compression: The In the processing of digital images, Image compression is considered as a technique for compression of an image that is stored in a computer for upgrading its visual properties. As an illustration, the image is made either bright or dark or enhancing contrast of image through implementation of some transformation functions or any other approach [1].

The process of image enhancement works by making adjustments over digitized images so that outcomes are more appropriate for presentation or image analysis. As an illustration, several non-required elements are eliminated by using the approaches like Wiener filter, Gaussian filter etc. Also, the image can be brighten or sharpen & making it easy to recognize prime attributes. Though, it will improvise dynamic range of the selected features and they can be identified very easily. The most difficult part of this technique is that it quantifies enhancement criteria. Hence, several techniques for image enhancement are considered to be empirical & need interactive process to generate outcomes. A selection for suitable techniques is supported by imaging modality, task over hand & viewing scenario

CHAPTER-2

LITERATURE SURVEY

Face detection is a computer technology that determines the location and size of human face in arbitrary (digital) image. The facial features are detected and any other objects like trees, buildings and bodies etc are ignored from the digital image. It can be regarded as a _specific‘case of object-class detection, where the task is finding the location and sizes of all objects in an image that belong to a given class. Face detection, can be regarded as a more _general‘case of face localization. In face localization, the task is to find the locations and sizes of a known number of faces (usually one). Basically there are two types of approaches to detect facial part in the given image i.e. feature base and image base approach. Feature base approach tries to extract features of the image and match it against the knowledge of the face features. While image base approach tries to get best match between training and testing images.

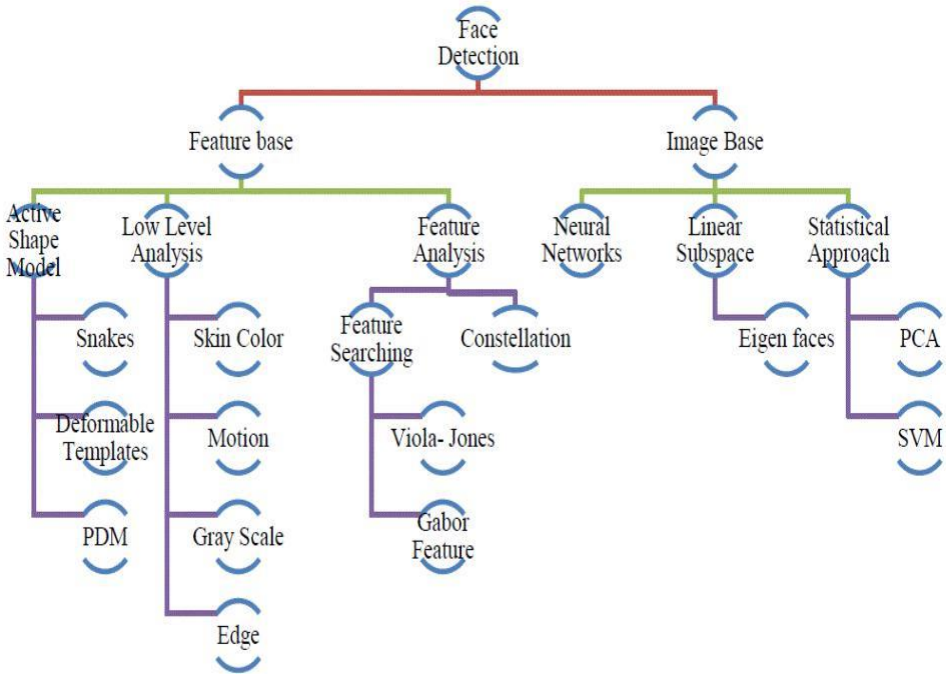


Fig 2.1 detection methods

2.1 1nstallation:

A Here we will be focusing on installing OpenCV for python only. We can install OpenCV using pip or conda(for anaconda environment).

Using pip:

Using pip, the installation process of openCV can be done by using the following command in the command prompt.

pip install opencv-python

2.2Face Recognition using Python

In this section, we shall implement face recognition using OpenCV and Python. First, let us see the libraries we will need and how to install them:

- OpenCV
- dlib
- Face recognition

OpenCV is an image and video processing library and is used for image and video analysis, like facial detection, license plate reading, photo editing, advanced robotic vision, optical character recognition, and a whole lot more.

The dlib library, maintained by Davis King, contains our implementation of "deep metric learning" which is used to construct our face embeddings used for the actual recognition process.

The face_recognition library, created by Adam Geitgey, wraps around dlib's facial recognition functionality, and this library is super easy to work with and we will be using this in our code. Remember to install dlib library first before you install face recognition.

To install OpenCV, type in command prompt

pip install opencv-python

I have tried various ways to install dlib on Windows but the easiest of all of them is via Anaconda. First, install Anaconda and then use this command in your command prompt:

conda install -c conda-forge dlib

Next to install face_recognition, type in command prompt

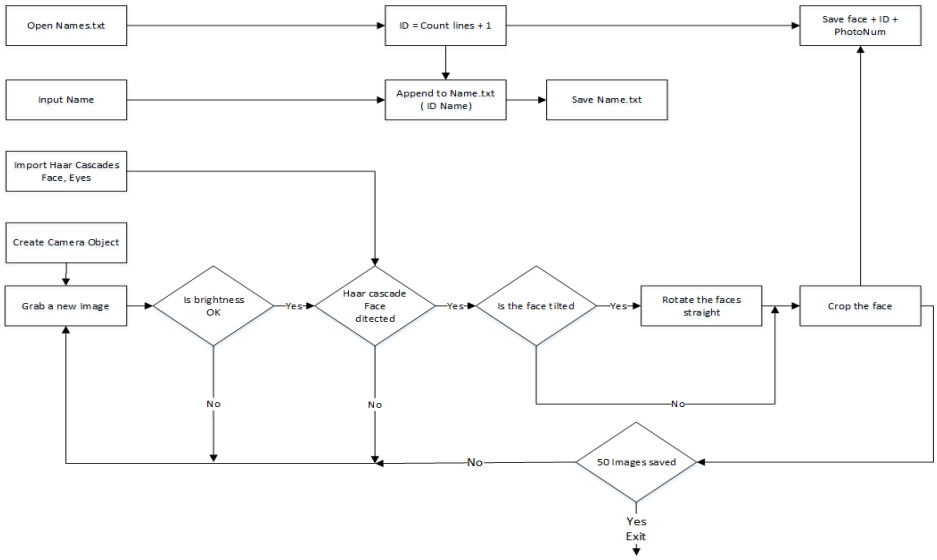
pip install face_recognition

Now that we have all the dependencies installed, let us start coding. We will have to create three files, one will take our dataset and extract face embedding for each face using dlib. Next, we will save these embedding in a file.

In the next file we will compare the faces with the existing the recognise faces in images and next we will do the same but recognise faces in live webcam feed.

2.2 Face Recognition Process:

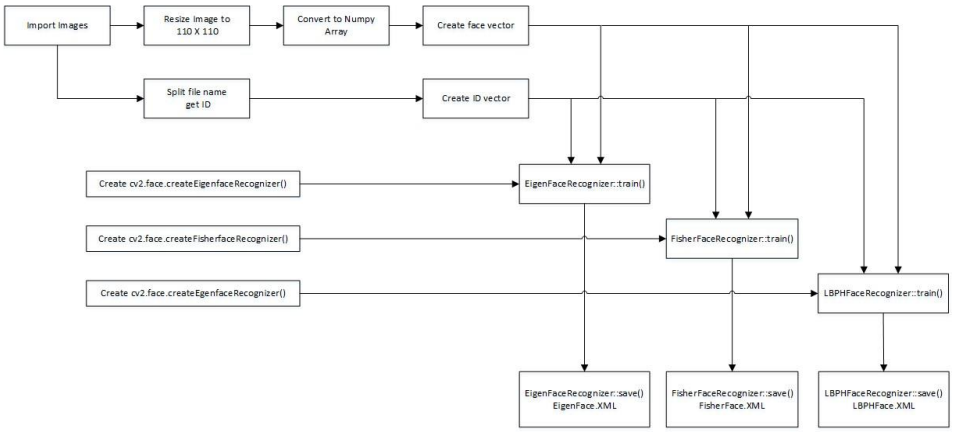
For this project three algorithms are implemented independently. These are Eigenface, Fisher face and Linear binary pattern histograms respectively. All three can be implemented using OpenCV libraries. There are three stages for the face recognition as follows: 1. Collecting images IDs 2. Extracting unique features, classifying them and storing in XML files 3. Matching features of an input image to the features in the saved XML files and predict identity. Collecting the image data Collecting classification images is usually done manually using a photo editing software to crop and resize photos. Furthermore, PCA and LDA requires the same number of pixels in all the images for the correct operation. This time consuming and a laborious task is automated through an application to collect 50 images with different expressions. The application detects suitable expressions between 300ms, straightens any existing tilt and save them. The Flow chart for the application is shown Application starts with a request for a name to be entered to be stored with the ID in a text file. The face detection system starts the first half. However, before the capturing begins, the application check for the brightness levels and will capture only if the face is well illuminated. Furthermore, after the face is detected, the position of the eyes are analysed. If the head is tilted, the application automatically corrects the orientation. These two additions were made considering the requirements for Eigenface algorithm. The Image is then cropped and saved using the ID as a filename to be identified later. A loop runs this program until 50 viable images are collected from the person. This application made data collection efficient.



The Flowchart for the image collection

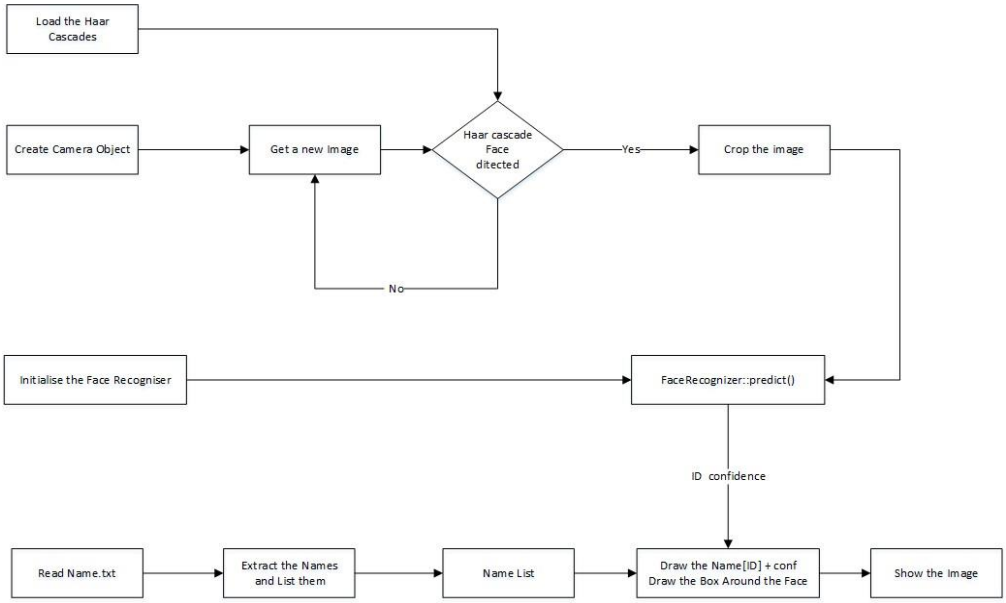
2.3 Training the Classifiers

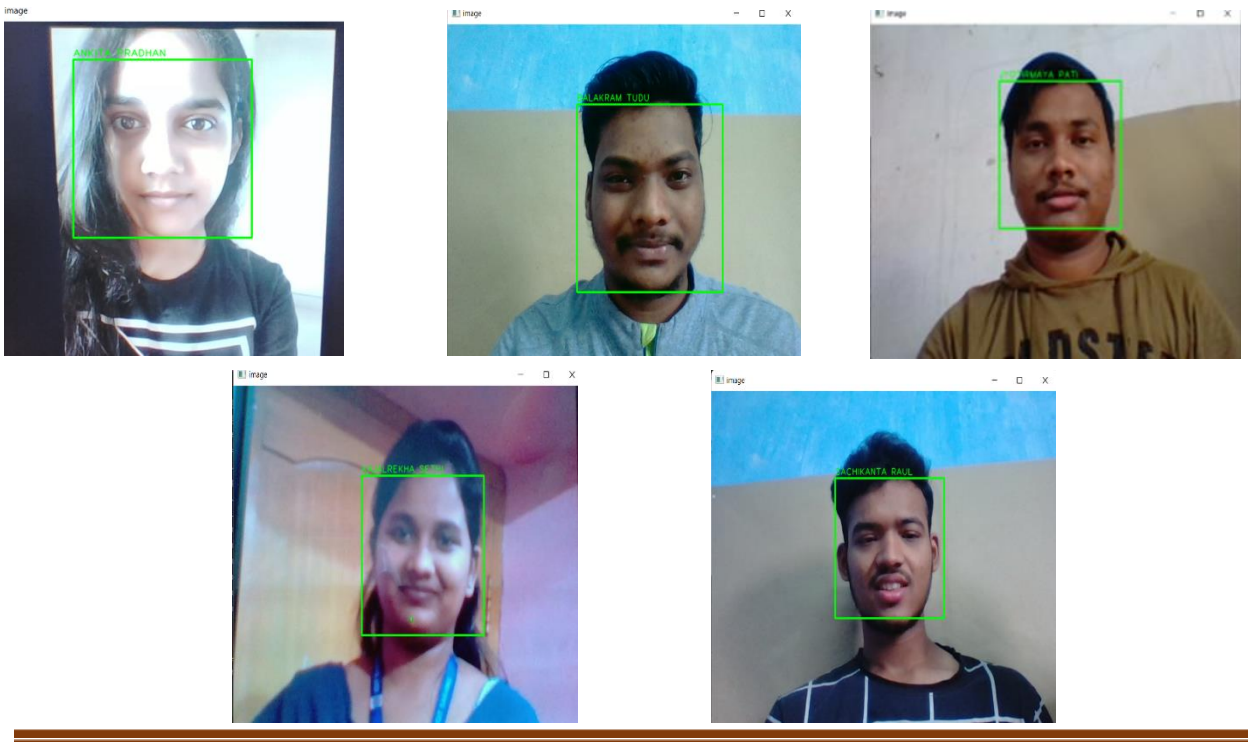
OpenCV enables the creation of XML files to store features extracted from datasets using the Face Recognizer class. The stored images are imported, converted to grayscale and saved with IDs in two lists with same indexes. Face Recognizer objects are created using face recogniser class. Each recogniser can take in parameters that are described below: cv2.face. createEigenFaceRecognizer() 1. Takes in the number of components for the PCA for crating Eigenfaces. OpenCV documentation mentions 80 can provide satisfactory reconstruction capabilities. 2. Takes in the threshold in recognising faces. If the distance to the likeliest Eigenface is above this threshold, the function will return a -1, that can be used state the face is unrecognisable 6 cv2.face. createFisherfaceRecognizer() 1. The first argument is the number of components for the LDA for the creation of Fisher faces. OpenCV mentions it to be kept 0 if uncertain. 2. Similar to Eigenface threshold. -1 if the threshold is passed. cv2.face.createLBPHFaceRecognizer() 1. The radius from the centre pixel to build the local binary pattern. 2. The Number of sample points to build the pattern. Having a considerable number will slow down the computer. 3. The Number of Cells to be created in X axis. 4. The number of cells to be created in Y axis. 5. A threshold value similar to Eigenface and Fisherface. if the threshold is passed the object will return -1 Recogniser objects are created and images are imported, resized, converted into numpy arrays and stored in a vector. The ID of the image is gathered from splitting the file name, and stored in another vector. By using FaceRecognizer.train(NumpyImage, ID) all three of the objects are trained. It must be noted that resizing the images were required only for Eigenface and Fisherface, not for LBPH. Next, the configuration model is saved as a XML file using FaceRecognizer.save(FileName). In this project, all three are trained and saved through one application for convenience. The flow chart for the trainer is shown



2.4 The Face Recognition

Face recogniser object is created using the desired parameters. Face detector is used to detect faces in the image, cropped and transferred to be recognised. This is done using the same technique used for the image capture application. For each face detected, a prediction is made using FaceRecognizer.predict() which return the ID of the class and confidence. The process is same for all algorithms and if the confidence his higher than the set threshold, ID is -1. Finally, names from the text file with IDs are used to display the name and confidence on the screen. If the ID is -1, the application will print unknown face without the confidence level. The flow chart for the application is shown





2.5 Compressing images using Python

Compressing images is a neat way to shrink the size of an image while maintaining the resolution. In this tutorial we're building an image compressor using Python, Numpy and Pillow. We'll be using machine learning, the unsupervised K-means algorithm to be precise.

If you don't have Numpy and Pillow installed, you can do so using the following command:

pip3 install pillow

pip3 install numpy

Start by importing the following libraries import os

import sys

from PIL import Image

import numpy as np

A. Performance Criteria In Image Compression

The performance can be estimated by implementation of below two necessary criteria the CR (compression ratio) & PSNR (quality measurement of reconstructed images).

4. *Compression ratio*

B. DCT Transformation

DCT (discrete cosine transforms) is mostly using for image compression from last many years. The selection as standard for JPEG is considerably onto the popular reason for popularity. DCT can be implemented in several non-analytical applications like signal processing & image processing DSP applications like video conferencing. The DCT can be used in the transformation for data compression. DCT is an actual transform that has a fixed set of basic functions. DCT is used for mapping an image space in a frequency

DCT aims various superiorities: (1) It is capable of confining the energy in lower frequency for image data. (2) It is capable of minimizing the blocking artifact & this impact will lead to visibility of boundaries in sub-images. Below we mention some advantages of the DCT:- real-valued.

2.6 FEATURE ANALYSIS

These algorithms aimto find structural features that exist even when the pose, viewpoint, or lighting conditions vary, andthen use these to locate faces. These methods are designed mainly for face localization

C. JPEG Compression

The JPEG standard is majorly known as ISO/ITU-T standard that is generated in late years of 1980s. The JPEG standard concentrates over full-colour still frame applications. A basic compression standard is the JPEG standard. Various modes are presented as JPEG comprising [1][2][3] baseline, lossless, progressive & hierarchal.

The general modes using DCT is JPEG baseline coding system, also it seems to be appropriate for majority of compression applications. In spite of the development in lower compression JPEG, it is highly helpful for DCT quantization & compression. JPEG compression will minimize the size of file through minimal image degradation by elimination of least required information. Though, it is referred as lossy image compression method as final image & actual image are not similar. The information of lossy compression will be lost while missed is affordable. Sequential steps will be performed for JPEG compression [11].

JPEG format will make use of lossy compression form constituted over DCT (discrete cosine transforms). Every field/frame of video source originated from spatial 2D-domain will be transformed to frequency domain (transom domain) with the help of some mathematical operations. A loosely operation constituted over perceptual model implemented over psychological system will eliminates the information having higher frequency i.e. sharp transitions of intensity & color hue. The process for minimizing the information in transform domain is termed as quantization. In a general wording, the process of quantization is explained as minimization of a larger number scale (having distinctive probabilities for every number) to the smaller values in optimal manner, and also the transform domain is an appropriate presentation of image as the frequency components re of higher value, that will contribute less to the whole picture than the other remaining coefficients, are having characteristically of lesser values with higher compressibility. The quantized coefficients will be further indulged to a sequence & loss loosely packed to output bit stream. Accordingly, all of the software applications of JPEG will allow control of user over compression ratio (also as per optimal parameters), permitting the user for trading off picture-quality for smaller size of file.
In the incorporated applications (like mini DV, that makes use of a similar type DCT-compression schema), the attributes are pre-selected & described for an application. The technique of compression will be generally lossy, which means that some original information about image will be lost, and it is not possible to restore them which will affect quality of image. The JPEG standard will define an additional lossless mode. Though, this mode will not be supported much by the products. Also there is an interlaced progressive JPEG format, where data get compressed in several passes with progressively higher details.

This works much fine for images of bigger size while it has to be downloaded from a slower connection. This will permit a sensible preview after obtaining a segment of the data. Though, progressive JPEG is not universally supported. As the programs receive progressive JPEGs which will not be supporting them (like versions of Internet Explorer even before Windows 7) [15] the software will present the image once it is downloaded completely.

There are several traffic & medical imaging systems which will generate & process 12-bit JPEG images, generally grayscale images. A 12-bit HPEG format has been considered as a segment of JPEG specification for some instances, but such format is not supported on wide scale.

- D. JPEG Process Steps For Color Images
- An RGB to YCbCr color space conversion (color specification).
- Actual image is segmented into blocks of 8 x 8 size.
- The pixel values within every block ranging from[-128 to 127] but pixel values of a black & white image range from [0-255] hence, every block is shifted from[0-255] to [-128 to 127].
- The DCT works from left side to right, top to bottom thereby it is implemented over every block.
- Quantization will compress every block. Quantized matrix is entropy encoded.
- Reverse process will help in reconstruction image. This process uses the IDCT (inverse Discrete Cosine Transform

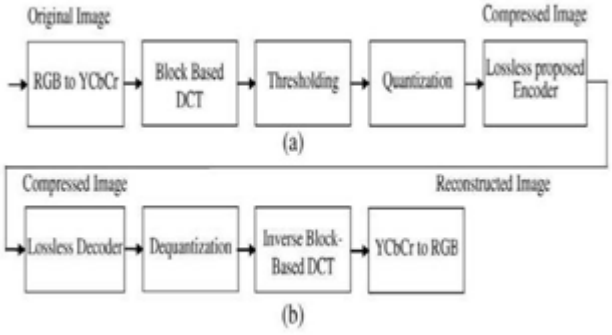


Figure . Compression algorithm scheme: (a) compression step and (b) decompression step

Below mention some advantages of Jpeg Image Compression

This file type has the lowest file sizes but compression artifacts may be visible as you lower the quality option.Since JPEG is a lossy format.

The quality loss is especially easy to see on sharp borders, on smooth borders and shading.
Similar as photos or realistic looking drawings.
The loss of quality is barely notices, if a JPEG image of high quality is selected.

E. Discrete Cosine Transform

After the transformation of colour coordinates, further step involves dividing the three colour elements of an image to several 8 x 8 blocks. Every component of original block in an 8-bit image will be considered in the range of [0,255]. A data range is produced that will centre around zero after subtraction of the mid pint (128 value) from every component of real block, so refined range will shift from [0,255] to [-128,127]. Images are segregated to several parts with different value of frequency

by DCT. The step of quantization will eliminates the less needed frequencies while decompression will make use of necessarily required frequencies for retrieval of image .

F. Quantization In JPEG

Generally, the data is eliminated through the Quantization. The quantization is performed through division of transformed image DCT matrix by making use of quantization matrix. The values for outcome matrix will be rounded off automatically. Quantization coefficient is presented in (6) & (7) will lead to reversal process.

Quantization focus over eliminating majority of the high frequency DCT coefficients that are not much important to 0, more will be the number of 0s better will be the compression of image. Lower frequencies are required for reconstruction of image as human eye possess more sensitiveness towards them & higher frequencies get discarded. Matrixes (8) & (9) stands for Q matrix defining luminance & chromatic elements [8][12].

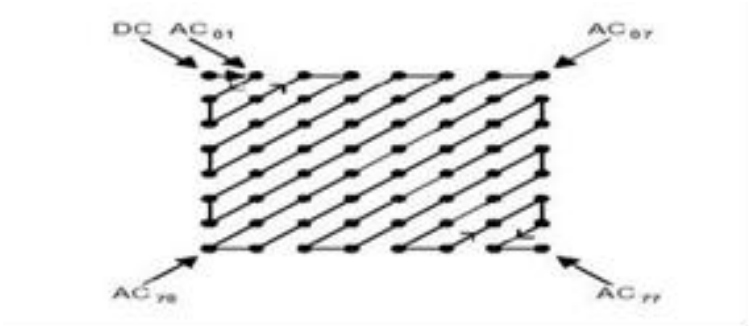


Figure . Zigzag Sequencing

G. Decompression

The phase of decompression works vice-versa to the compression. The initial step of this process is to restore Huffman tables from pictures & further decompression of Huffman tokens into the image. in the next step, DCT values for every block are firstly requires for decompressing a block. JPEG will help in decompression of remaining 63 values in every block by filling in required number of 0s. In the last step, combined zigzag order will be decoded and blocks of 8x8 sizes are formed. The IDCT (Inverse DCT) considers every value in the spatial domain & evaluates the contributions that are made by every 64 frequency values forming a pixel [7].

III. PROPOSED METHODOLOGY

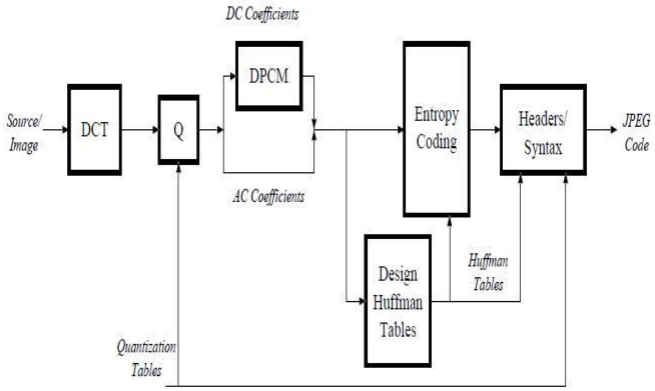
A novel technique for post-processing the JPEG encoded images is proposed for minimizing coding artifacts & improvises quality of visualization. Our proposed technique is re-applied over JPEG towards shifted versions of the images that are already been compressed & makes an average. In spite of the simplicity, this technique perform better that other methodologies while comprising non-linear filtration, redundant wavelets & POCS.

Block transform coding of pictures through DCT (Discrete Cosine Transform) has observed to be a basic though efficient technique for image compression. Various implementations of these techniques are observed to be accepted widely as per international standards of image & video compression, like MPEG & JPEG standards.

The prime approach for block transformation compression is very basic. The process of encoding is comprised of diving a image in blocks of standard 8x8 size. A block transform algorithm, generally DCT is implemented over these blocks & transform coefficients are quantized on individual basis (scalar quantization). Some lossless compression functions are executed for presenting the resultant data efficiently, while precisely comprising zigzag scanning of coefficients & entropy coding. A basic diagram of the whole process is demonstrated in figure 4.

The process of block encoding, while being simple & efficient, presents several non-desirable artifacts in an image; most prominently observant artifacts (discontinuities over block boundaries) & ringing artifacts (oscillations because of Gibbs phenomenon). The artifacts will sound more with an increase in compression ratio.

A significant working schema has been proposed for addressing the improvisation of DCT compressed images. The working in JPEG enhancement method is gaining much interest as the present number of JPEG encoded images is in millions and will keep on rising beyond impending introduction about JPEG 2000. A major illustration of this proliferation is over internet in which several web pages will makes use of JPEG encoded images. Another illustration is the images generated by digital cameras.



A novel post processing technique is presented for minimizing compression artifacts in JPEG encoded pictures. This technique is termed as significant departure from past few signs processing techniques, in this it don't particularly observe discontinuities in the boundaries of blocks, or neither it makes use of smoothness criterion. It follows JPEG process for minimizing compression artifacts of JPEG encoded images. This technique can be easily executed & apart from its simple nature, has a super competitive performance.

CHAPTER 3

3.1 Face Detection

Face detection is a computer technology that determines the location and size of human face in arbitrary (digital) image. The facial features are detected and any other objects like trees, buildings and bodies etc are ignored from the digital image. It can be regarded as a _specific ‘case of object-class detection, where the task is finding the location and sizes of all objects in an image that belong to a given class. Face detection, can be regarded as a more _general ‘case of face localization. In face localization, the task is to find the locations and sizes of a known number of faces (usually one). Basically, there are two types of approaches to detect facial part in the given image i.e., feature base and image base approach. Feature base approach tries to extract features of the image and match it against the knowledge of the face features. While image base approach tries to get best match between training and testing images.

3.2MOTION BASE:

When use of video sequence is available, motion information can be used to locate moving objects. Moving silhouettes like face and body parts can be extracted by simply thresholding accumulated frame differences. Besides face regions, facial features can be located by frame differences.

3.2.1Gray Scale Base:

Gray information within a face can also be treat as important features. Facial features such as eyebrows, pupils, and lips appear generally darker than their surrounding facial regions. Various recent feature extraction algorithms search for local gray minima within segmented facial regions. In these algorithms, the input images are first enhanced by contrast-stretching and gray-scale morphological routines to improvethe quality of local dark patches and thereby make detection easier. The extraction of darkpatches is achieved by low-level gray-scale thresholding. Based method and consist three levels. Yang and huang presented new approach i.e. faces gray scale behaviour in pyramid (mosaic) images. This system utilizes hierarchical Face location consist three levels. Higher two level based on mosaic images at different resolution. In the lower level, edge detection method is proposed. Moreover this algorithms gives fine response in complex background where size of the face is unknown

3.2.2 Edge Base:

Face detection based on edges was introduced by Sakai et al. This workwas based on analysing line drawings of the faces from photographs, aiming to locate facialfeatures. Than later Craw et al. proposed a hierarchical framework based on Sakai et al.‘swork to trace a human head outline.

Then after remarkable works were carried out by many researchers in this specific area. They proposed frame work which consist three steps i.e. initially the images are enhanced by applying median filter for noise removal and histogram equalization for contrast adjustment. In the second step the edge image is constructed from the enhanced image by applying sobel operator. Then a novel edge tracking algorithm is applied to extract the sub windows from the enhanced image based on edges.

3.3 Face Detection

The problem of face recognition is all about face detection. This is a fact that seems quite bizarre to new researchers in this area. However, before face recognition is possible, one must be able to reliably find a face and its landmarks. This is essentially a segmentation problem and in practical systems, most of the effort goes into solving this task. In fact the actual recognition based on features extracted from these facial landmarks is only a minor last step. There are two types of face detection problems:

1) Face detection in images and

2) Real-time face detection

1 FACE DETECTION IN IMAGES

Most face detection systems attempt to extract a fraction of the whole face, thereby eliminating most of the background and other areas of an individual's head such as hair that are not necessary for the face recognition task. With static images, this is often done by running a across the image. The face detection system then judges if a face is present inside the window (Brunelli and Poggio, 1993). Unfortunately, with static images there is a very large search space of possible locations of a face in image Most face detection systems use an example-based learning approach to decide whether or not a face is present in the window at that given instant (Sung and Poggio,1994 and sung,1995). A neural network or some other

classifier is trained using supervised learning with 'face' and 'nonface' examples, thereby enabling it to classify an image (window in face detection system) as a 'face' or 'non-face'.. Unfortunately, while it is relatively easy to find face examples, how would one find a representative sample of images which represent non-faces (Rowley et al., 1996)? Therefore, face detection systems using example-based learning need thousands of 'faces' and 'nonface' images for effective training. Rowley, Baluja, and Kanade (Rowley et al.,1996) used 1025 face images and 8000 non-face images (generated from 146,212,178 sub-images) for their training set!

There is another technique for determining whether there is a face inside the face detection system's window - using Template Matching. The difference between a fixed target pattern (face) and the window is computed and thresholder. If the window contains a pattern which is close to the target pattern(face) then the window is judged as containing a face. An implementation of template matching called Correlation Templates uses a whole bank of fixed sized templates to detect facial features in an image (Bichsel, 1991 & Brunelli and Poggio, 1993). By using several templates of different (fixed) sizes, faces of different scales (sizes) are detected. The other implementation of template matching is using a deformable template (Yuille, 1992). Instead of using several fixed size templates, we use a deformable template (which is non-rigid) and there by change the size of the template hoping to detect a face in an image. A face detection scheme that is related to template matching is image invariants. Here the fact that the local ordinal structure of brightness distribution of a face remains largely unchanged under different illumination conditions (Sinha, 1994) is used to construct a spatial template of the face which closely corresponds to facial features. In other words, the average grey-scale intensities in human faces are used as a basis for face detection. For example, almost always an individual’s eye region is darker than his forehead or nose. Therefore, an image will match the template if it satisfies the 'darker than' and 'brighter than' relationships (Sung and Poggio, 1994).

2 REAL-TIME FACE DETECTION

Real-time face detection involves detection of a face from a series of frames from a video capturing device. While the hardware requirements for such a system are far more stringent, from a computer vision stand point, real-time face detection is actually a far simpler process than detecting a face in a static image. This is because unlike most of our surrounding environment, people are continually moving. We walk around, blink, fidget, wave our hands about, etc.

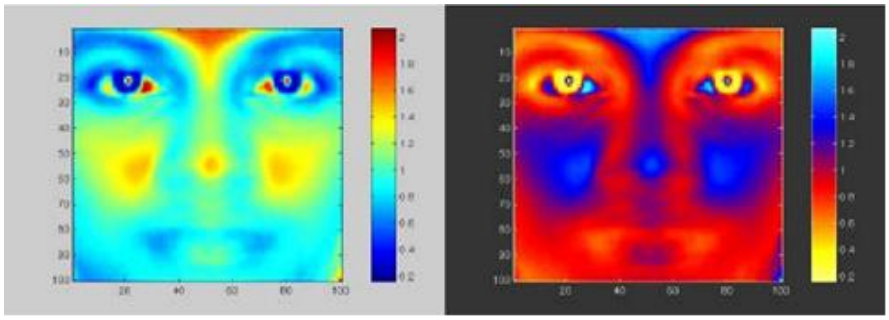
Since in real-time face detection, the system is presented with a series of frames in which to detect a face, by using spatiotemporal filtering (finding the difference between subsequent frames), the area of the frame that has changed can be identified and the individual detected (Wang and Adelson, 1994 and Adelson and Bergen 1986). Furthermore as seen in Figure exact face locations can be easily identified by using a few simple rules, such as,

1)the head is the small blob above a larger blob -the body

2)head motion must be reasonably slow and contiguous -heads won't jump around erratically (Turk and Pentland 1991a, 1991b). Real-time face detection has therefore become a relatively simple problem and is possible even in unstructured and uncontrolled environments using these very simple image processing techniques and reasoning rules.

3.4 FACE DETECTION PROCESS

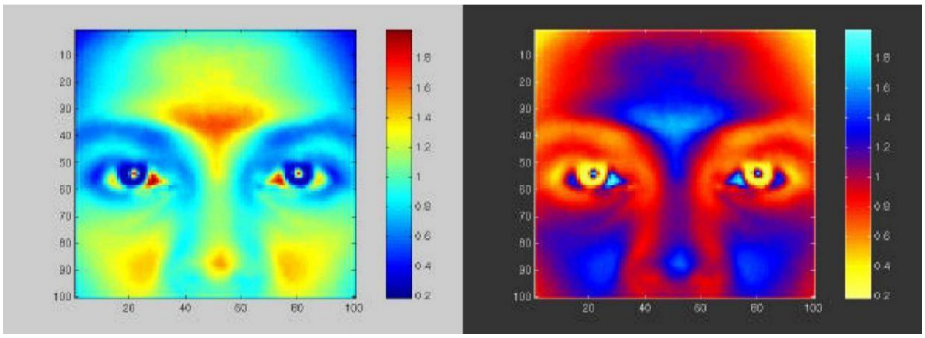
It is process of identifying different parts of human faces like eyes, nose, mouth, etc… this process can be achieved by using PYC codeIn this project the author will attempt to detect faces in still images by using image invariants. To do this it would be useful to study the grey-scale intensity distribution of an average human face. The following 'average human face' was constructed from a sample of 30 frontal view human faces, of which 12 were from females and 18 from males. A suitably scaled colormap has been used to highlight grey-scale intensity differences.



scaled colormap scaled(negative)

Average human face in grey-scale

The grey-scale differences, which are invariant across all the sample faces are strikingly apparent. The eye-eyebrow area seem to always contain dark intensity (low) gray-levels while nose forehead and cheeks contain bright intensity (high) grey-levels. After a great deal of experimentation, the researcher found that the following areas of the human face were suitable for a face detection system based on image invariants and a deformable template



scaled colormap scaled colormap (negative)

Area chosen for face detection (indicated on average human face in gray scale

The above facial area performs well as a basis for a face template, probably because of the clear divisions of the bright intensity invariant area by the dark intensity invariant regions. Once this pixel area is located by the face detection system, any particular area required can be segmented based on the proportions of the average human face After studying the above images it was subjectively decided by the author to use the following as a basis for dark intensity sensitive and bright intensity sensitive templates. Once these are located in a subject's face, a pixel area 33.3% (of the width of the square window) below this. Basis for a bright intensity invariant sensitive template.

Note the slight differences which were made to the bright intensity invariant sensitive template which were needed because of the pre-processing done by the system to overcome irregular lighting (chapter six). Now that a suitable dark and bright intensity invariant templates have been decided on, it is necessary to find a way of using these to make 2 A-units for a perceptron, i.e. a computational model is needed to assign neurons to the distributions displayed.

A. Huffman Coding

Huffman coding is considered as lossless data compression algorithm. The motive behind this is allocating variable-length codes for inputting characters; length of allocated codes is constituted over frequency of associated characters. The most frequently used character will be

having smallest code & least frequent character will be having biggest code.

Huffman coding is considered as some of the most prominent techniques for elimination of redundancy in coding. It is been implemented in several compression algorithms, incorporating image compression. It is the most basic, but yet elegant, compression methodology will supplement multiple compression algorithms. It is also implemented in CCITT Group 3 compression. Group 3 is referred as compression algorithm that was produced by International Telegraph & Telephone Consultative Committee in th year 1958 for encoding & compression of 1-bit (monochrome) image data. Group 3 & 4 compressions are generally implemented in TIFF format. It will make use of statistical characteristics of alphabets in a source stream & further generates associated codes for such alphabets. These codes have a variable code length while making use of integral number of bits. The alphabetical codes processing higher probability for occurrence has short length than the codes for alphabets possessing lesser probability.

These codes have a variable code length while making use of integral number of bits. The alphabetical codes processing higher probability for occurrence has short length than the codes for alphabets possessing lesser probability. Hence it is considered over frequency of occurrence of a data item (pixels or small blocks of pixels in images). It requires lesser number of bits for encoding frequency used information. The codes will be accumulated in a code book. A code book will be made for every image or set of images.

Huffman coding is considered as most optimal lossless schema for compression of a bit stream. It operates by firstly making calculations of probabilities. Defining permutations {0,1} n by allocating symbols, termed as A;B;C;D. The bit stream may be seemed as AADAC. As an illustration. Now the symbols are allocated newer codes, higher will be the probability, lower will be the number of bits in code [3]. These codes serve as outcome of Huffman coder in form of bit stream. Now stopping point of code must be known & point for starting a new code. This problem is solved through enforcement of unique prefix condition: no code is prefix of any other code. The initial codes are referred as 01; 11; 001; 101; 0000; 1000l; 1001. In the Huffman coding schema, shorter codes are allotted to the symbols that are incorporated on frequent basis & longer codes to those which seems to occur less frequently [1].

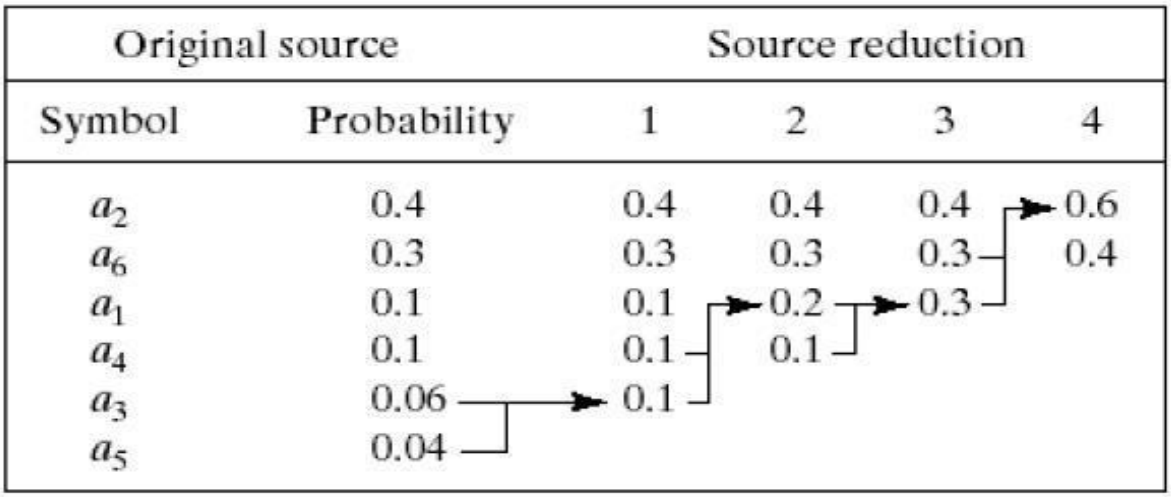


Fig.4.5.4.workspace

B. Huffman Code Construction

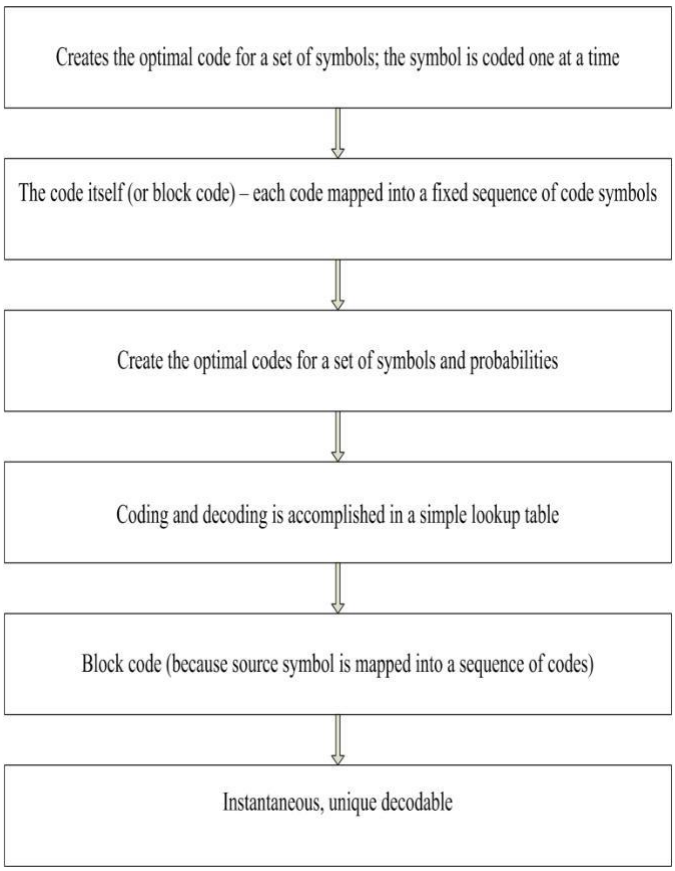
Huffman Coding Algorithm works as bottom-up approach.

- Algorithm

Steps of Huffman Coding algorithm are mentioned as below [9]:

1.Generating a series of source reduction: combining two minimal probability symbol to a single symbol; it is repeated till a minimized source having two symbols is obtained.

Coding every reduced symbol: starting from smallest source & coming back to actual source



Huffman source reduction

Huffman codes are considered as optional prefix codes produced through a set of probabilities by a definite algorithm i.e., Huffman Coding Algorithm.

.

IV.RESULTS

Image compression is an important technique for reducing the size of the image and for sending it in low size. Image has many types in this lossy compression is widely using in network-related applications. The applications of lossless compression is file image/video/audio compression, big size file data zip and much more. In this research, we are comparing two types of images

1: - Gray Scale Colour Image

2: - Colour Image

As fig 9 is showing the grey colour image. It is out original image which we have to compress

Figure 9 : - Original Image

Fig 10 is showing the Compressed image of the original image.

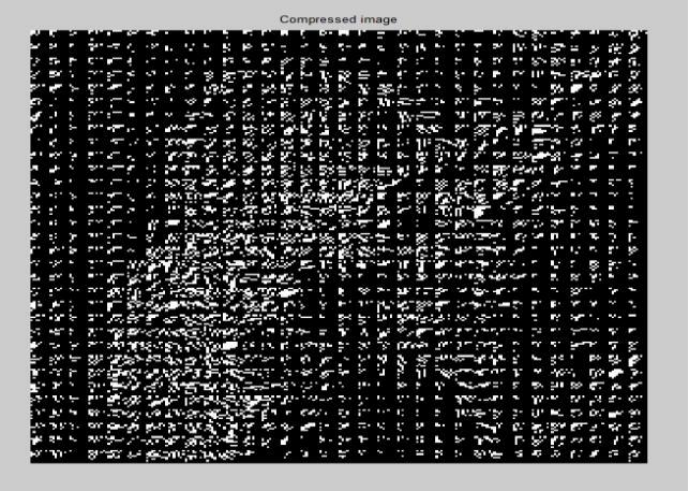


Figure10 : - Compressed image

Fig 11 is showing the Decompressed image of the compressed image by the DCT technique

Figure 11 :- Decompressed Image

Fig 12 is showing the original colour image. For show the results we will compress the colour image also. In the figure 5.5 and 5.6 we will show the compressed and decompressed image.

Figure: - Original Image

Figure is showing the compressed colour image of the original image.

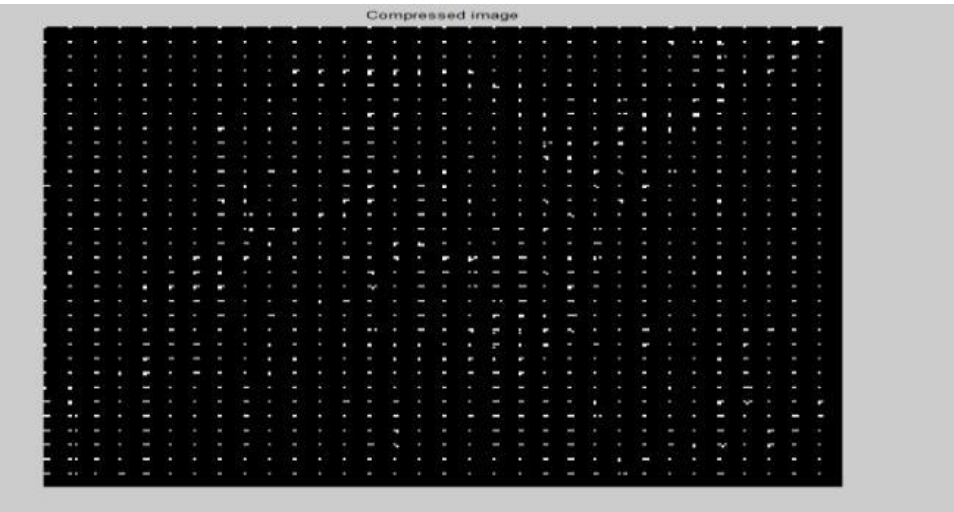


Figure 13:- Compressed Image

In fig 14 is showing the decompressed image which get from the compressed image

Method	JPEG Existing	JPEG Proposed
MSE	17502.21	63.63
PSNR	5.70	30.09
NK	0.00	1.00
AD	123.58	-0.02
SC	64979.04	1.00
MD	239.00	58.00
NAE	1.00	0.04

Table 1 :- Comparison Table

CHAPTER 4

Code for app-gui.py

```
from Detector import main_app
from create_classifier import train_classifer
from create_dataset import start_capture
#from huffman import start_compress
import tkinter as tk
from tkinter import font as tkfont
from tkinter import messagebox,PhotoImage
#from PIL import ImageTk, Image
#from gender_prediction import emotion,ageAndgender
names = set()
class MainUI(tk.Tk):
    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)
        global names
        with open("nameslist.txt", "r") as f:
            x = f.read()
            z = x.rstrip().split(" ")
            for i in z:
                names.add(i)
        self.title_font = tkfont.Font(family='Helvetica', size=16, weight="bold")
        self.title("Face Recognizer")
        self.resizable(False, False)
        self.geometry("500x300")
        self.protocol("WM_DELETE_WINDOW", self.on_closing)
        self.active_name = None
        container = tk.Frame(self)
        container.grid(sticky="nsew")

container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)
        self.frames = {}
        for F in (StartPage, PageOne, PageTwo, PageThree, PageFour):

page_name = F.__name__
            frame = F(parent=container, controller=self)
            self.frames[page_name] = frame
            frame.grid(row=0, column=0, sticky="nsew")
```

```
 self.show_frame("StartPage")

    def show_frame(self, page_name):
            frame = self.frames[page_name]
            frame.tkraise()

    def on_closing(self):

        if messagebox.askokcancel("Quit", "Are you sure?"):
            global names
            f =  open("nameslist.txt", "a+")


for i in names:
            f.write(i+" ")
        self.destroy()


class StartPage(tk.Frame):

     def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller
        #load = Image.open("homepagepic.png")
        #load = load.resize((250, 250), Image.ANTIALIAS)
        render = PhotoImage(file='homepagepic.png')
        img = tk.Label(self, image=render)
        img.image = render
        img.grid(row=0, column=1, rowspan=4, sticky="nsew")
        label = tk.Label(self, text="       GROUP NO- 06    ", font=self.controller.title_font,fg="#ff6f00")
        label.grid(row=0, sticky="ew")
        button1 = tk.Button(self, text="   Add a User  ", fg="#ffffff", bg="#263942",command=lambda:
self.controller.show_frame("PageOne"))
        button2 = tk.Button(self, text="   Check a User  ", fg="#ffffff", bg="#263942",command=lambda:
self.controller.show_frame("PageTwo"))
        button3 = tk.Button(self, text="Quit", fg="#263942", bg="#ffffff", command=self.on_closing)
        button1.grid(row=1, column=0, ipady=3, ipadx=7)
        button2.grid(row=2, column=0, ipady=3, ipadx=2)
        button3.grid(row=3, column=0, ipady=3, ipadx=32)

def on_closing(self):
        if messagebox.askokcancel("Quit", "Are you sure?"):
            global names
            with open("nameslist.txt", "w") as f:
                for i in names:
                    f.write(i + " ")
            self.controller.destroy()


class PageOne(tk.Frame):
   def __init__(self, parent, controller):
      tk.Frame.__init__(self, parent)
      self.controller = controller
      tk.Label(self, text="Enter the name", fg="#263942", font='Helvetica 12 bold').grid(row=0, column=0, pady=10,
padx=5)
      self.user_name = tk.Entry(self, borderwidth=3, bg="green", font='Helvetica 11')
      self.user_name.grid(row=0, column=1, pady=10, padx=10)
```

```
self.buttoncanc = tk.Button(self, text="Cancel", bg="#ffffff", fg="#263942", command=lambda:
controller.show_frame("StartPage"))
        self.buttonext = tk.Button(self, text="Next", fg="#ffffff", bg="#263942", command=self.start_training)
        self.buttoncanc.grid(row=1, column=0, pady=10, ipadx=5, ipady=4)
        self.buttonext.grid(row=1, column=1, pady=10, ipadx=5, ipady=4)
    def start_training(self):
        global names
        if self.user_name.get() == "None":
            messagebox.showerror("Error", "Name cannot be 'None'")
            return
        elif self.user_name.get() in names:
            messagebox.showerror("Error", "User already exists!")
            return
        elif len(self.user_name.get()) == 0:
            messagebox.showerror("Error", "Name cannot be empty!")
            return
        name = self.user_name.get()
        names.add(name)
        self.controller.active_name = name
        self.controller.frames["PageTwo"].refresh_names()
        self.controller.show_frame("PageThree")


class PageTwo(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        global names
        self.controller = controller
        tk.Label(self, text="Select user", fg="#263942", font='Helvetica 12 bold').grid(row=0, column=0, padx=10,
pady=10)
        self.buttoncanc = tk.Button(self, text="Cancel", command=lambda: controller.show_frame("StartPage"),
bg="#ffffff", fg="#263942")
        self.menuvar = tk.StringVar(self)
        self.dropdown = tk.OptionMenu(self, self.menuvar, *names)
        self.dropdown.config(bg="lightgrey")
        self.dropdown["menu"].config(bg="lightgrey")
        self.buttonext = tk.Button(self, text="Next", command=self.nextfoo, fg="#ffffff", bg="#263942")
        self.dropdown.grid(row=0, column=1, ipadx=8, padx=10, pady=10)
        self.buttoncanc.grid(row=1, ipadx=5, ipady=4, column=0, pady=10)
        self.buttonext.grid(row=1, ipadx=5, ipady=4, column=1, pady=10)

def nextfoo(self):
        if self.menuvar.get() == "None":
            messagebox.showerror("ERROR", "Name cannot be 'None'")
            return
        self.controller.active_name = self.menuvar.get()
        self.controller.show_frame("PageFour")

    def refresh_names(self):
        global names
        self.menuvar.set('')
        self.dropdown['menu'].delete(0, 'end')

for name in names:
            self.dropdown['menu'].add_command(label=name, command=tk._setit(self.menuvar, name))
```

```
class PageThree(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller
        self.numimglabel = tk.Label(self, text="Number of images captured = 0", font='Helvetica 12 bold', fg="#263942")
        self.numimglabel.grid(row=0, column=0, columnspan=2, sticky="ew", pady=10)
        self.capturebutton = tk.Button(self, text="Capture Data Set", fg="#ffffff", bg="#263942", command=self.capimg)

        self.trainbutton = tk.Button(self, text="Train The Model", fg="#ffffff", bg="#263942",command=self.trainmodel)
        self.compress = tk.Button(self, text="Compress The Capture Data Set ", fg="#ffffff",
bg="#263942",command=self.compressmodel)

        self.capturebutton.grid(row=1, column=0, ipadx=5, ipady=4, padx=10, pady=20)
        self.trainbutton.grid(row=1, column=1, ipadx=5, ipady=4, padx=10, pady=20)
        self.compress.grid(row=1, column=2, ipadx=5, ipady=4, padx=10, pady=20)


def capimg(self):
        self.numimglabel.config(text=str("Captured Images = 0 "))
        messagebox.showinfo("INSTRUCTIONS", "We will Capture 300 pic of your Face.")
        x = start_capture(self.controller.active_name)
        self.controller.num_of_images = x
        self.numimglabel.config(text=str("Number of images captured = "+str(x)))

    def trainmodel(self):
        if self.controller.num_of_images < 300:
            messagebox.showerror("ERROR", "No enough Data, Capture at least 300 images!")
            return
        train_classifer(self.controller.active_name)
        messagebox.showinfo("SUCCESS", "The modele has been successfully trained!")
        self.controller.show_frame("PageThree")

    def compressmodel(self):
        if self.controller.num_of_images < 300:
            start_compress(self.controller.active_name)
        messagebox.showinfo("SUCCESS", "The modele has been successfully Compressed!")
        self.controller.show_frame("PageFour")

class PageFour(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller

        label = tk.Label(self, text="Face Recognition", font='Helvetica 16 bold')
        label.grid(row=0,column=0, sticky="ew")

        button1 = tk.Button(self, text="Face Recognition", command=self.openwebcam, fg="#ffffff", bg="#263942")
        #button2 = tk.Button(self, text="Emotion Detection", command=self.emot, fg="#ffffff", bg="#263942")
        #button3 = tk.Button(self, text="Gender and Age Prediction", command=self.gender_age_pred, fg="#ffffff",
bg="#263942")

        button4 = tk.Button(self, text="Go to Home Page", command=lambda: self.controller.show_frame("StartPage"),
bg="#ffffff", fg="#263942")
        button1.grid(row=1,column=0, sticky="ew", ipadx=5, ipady=4, padx=10, pady=10)
        #button2.grid(row=1,column=1, sticky="ew", ipadx=5, ipady=4, padx=10, pady=10)
        #button3.grid(row=2,column=0, sticky="ew", ipadx=5, ipady=4, padx=10, pady=10)
        button4.grid(row=1,column=1, sticky="ew", ipadx=5, ipady=4, padx=10, pady=10
```

```
def nextfoo(self):
        if self.menuvar.get() == "None":
            messagebox.showerror("ERROR", "Name cannot be 'None'")
            return
        self.controller.active_name = self.menuvar.get()
        self.controller.show_frame("PageFour")

    def refresh_names(self):
        global names
        self.menuvar.set('')
        self.dropdown['menu'].delete(0, 'end')

        for name in names:
            self.dropdown['menu'].add_command(label=name, command=tk._setit(self.menuvar, name))

class PageThree(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller
        self.numimglabel = tk.Label(self, text="Number of images captured = 0", font='Helvetica 12 bold', fg="#263942")
        self.numimglabel.grid(row=0, column=0, columnspan=2, sticky="ew", pady=10)
        self.capturebutton = tk.Button(self, text="Capture Data Set", fg="#ffffff", bg="#263942", command=self.capimg)

        self.trainbutton = tk.Button(self, text="Train The Model", fg="#ffffff", bg="#263942",command=self.trainmodel)
        self.compress = tk.Button(self, text="Compress The Capture Data Set ", fg="#ffffff",
bg="#263942",command=self.compressmodel)

        self.capturebutton.grid(row=1, column=0, ipadx=5, ipady=4, padx=10, pady=20)
        self.trainbutton.grid(row=1, column=1, ipadx=5, ipady=4, padx=10, pady=20)
        self.compress.grid(row=1, column=2, ipadx=5, ipady=4, padx=10, pady=20)

    def capimg(self):
        self.numimglabel.config(text=str("Captured Images = 0 "))
        messagebox.showinfo("INSTRUCTIONS", "We will Capture 300 pic of your Face.")
        x = start_capture(self.controller.active_name)
        self.controller.num_of_images = x
        self.numimglabel.config(text=str("Number of images captured = "+str(x)))

    def trainmodel(self):
        if self.controller.num_of_images < 300:
            messagebox.showerror("ERROR", "No enough Data, Capture at least 300 images!")
            return
        train_classifer(self.controller.active_name)
        messagebox.showinfo("SUCCESS", "The modele has been successfully trained!")
        self.controller.show_frame("PageThree")

def compressmodel(self):
        if self.controller.num_of_images < 300:
            start_compress(self.controller.active_name)
        messagebox.showinfo("SUCCESS", "The modele has been successfully Compressed!")
        self.controller.show_frame("PageFour")

class PageFour(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        label = tk.Label(self, text="Face Recognition", font='Helvetica 16 bold')
        label.grid(row=0,column=0, sticky="ew")
```

```
        button1 = tk.Button(self, text="Face Recognition", command=self.openwebcam, fg="#ffffff",
bg="#263942")
    #button2 = tk.Button(self, text="Emotion Detection", command=self.emot, fg="#ffffff", bg="#263942")
    #button3 = tk.Button(self, text="Gender and Age Prediction", command=self.gender_age_pred,
fg="#ffffff", bg="#263942")
    button4 = tk.Button(self, text="Go to Home Page", command=lambda:
self.controller.show_frame("StartPage"), bg="#ffffff", fg="#263942")
    button1.grid(row=1,column=0, sticky="ew", ipadx=5, ipady=4, padx=10, pady=10)
    #button2.grid(row=1,column=1, sticky="ew", ipadx=5, ipady=4, padx=10, pady=10)
    #button3.grid(row=2,column=0, sticky="ew", ipadx=5, ipady=4, padx=10, pady=10)
    button4.grid(row=1,column=1, sticky="ew", ipadx=5, ipady=4, padx=10, pady=10)


  def openwebcam(self):
    main_app(self.controller.active_name)




app = MainUI()
app.iconphoto(False, tk.PhotoImage(file='icon.png'))
app.mainloop()
```

Code for creat_dataset.py

```
import cv2
import os

def start_capture(name):
    path = "./data/" + name
    num_of_images = 0
    detector = cv2.CascadeClassifier("./data/haarcascade_frontalface_default.xml")
    try:
        os.makedirs(path)
    except:
        print('Directory Already Created')
    vid = cv2.VideoCapture(0)
    while True:

        ret, img = vid.read()
        new_img = None
        grayimg = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        face = detector.detectMultiScale(image=grayimg, scaleFactor=1.1, minNeighbors=5)
        for x, y, w, h in face:
            cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 0), 2)
            cv2.putText(img, "Face Detected", (x, y-5), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255))
            cv2.putText(img, str(str(num_of_images)+" images captured"), (x, y+h+20), cv2.FONT_HERSHEY_SIMPLEX,
0.8, (0, 0, 255))
            new_img = img[y:y+h, x:x+w]
        cv2.imshow("FaceDetection", img)
        key = cv2.waitKey(1) & 0xFF
```

```
            try :
                cv2.imwrite(str(path+"/"+str(num_of_images)+name+".jpg"),
new_img)
                num_of_images += 1
            except :

                pass
            if key == ord("q") or key == 27 or num_of_images > 300:
                break
        cv2.destroyAllWindows()
        return num_of_images
```

.

Code for detector.py

```
import cv2
from time import sleep
from PIL import Image

def main_app(name):

    face_cascade = cv2.CascadeClassifier('./data/haarcascade_frontalface_default.xml')
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    recognizer.read(f"./data/classifiers/{name}_classifier.xml")
    cap = cv2.VideoCapture(0)
    pred = 0
    while True:
        ret, frame = cap.read()
        #default_img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray,1.3,5)

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


            roi_gray = gray[y:y+h,x:x+w]

            id,confidence = recognizer.predict(roi_gray)
            confidence = 100 - int(confidence)
            pred = 0
            if confidence > 50:
                #if u want to print confidence level
                    #confidence = 100 - int(confidence)
                    pred += +1
                    text = name.upper()
                    font = cv2.FONT_HERSHEY_PLAIN
                    frame = cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
                    frame = cv2.putText(frame, text, (x, y-4), font, 1, (0, 255, 0), 1, cv2.LINE_AA)

            else:
                    pred += -1
                    text = "UnknownFace"
                    font = cv2.FONT_HERSHEY_PLAIN
                    frame = cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)
                    frame = cv2.putText(frame, text, (x, y-4), font, 1, (0, 0,255), 1, cv2.LINE_AA
```

```
cv2.imshow("image", frame)


        if cv2.waitKey(20) & 0xFF == ord('q'):
            print(pred)
            if pred > 0 :
                dim =(124,124)
                img = cv2.imread(f".\\data\\{name}\\{pred}{name}.jpg", cv2.IMREAD_UNCHANGED)
                resized = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)
                cv2.imwrite(f".\\data\\{name}\\50{name}.jpg", resized)
                Image1 = Image.open(f".\\2.png")

                # make a copy the image so that the
                # original image does not get affected
                Image1copy = Image1.copy()
                Image2 = Image.open(f".\\data\\{name}\\50{name}.jpg")
                Image2copy = Image2.copy()

                # paste image giving dimensions
                Image1copy.paste(Image2copy, (195, 114))

                # save the image
                Image1copy.save("end.png")
                frame = cv2.imread("end.png", 1)

                cv2.imshow("Result",frame)
                cv2.waitKey(5000)
            break


    cap.release()
    cv2.destroyAllWindows()
```

Code for creat_classifier.py

```
import numpy as np
from PIL import Image
import os, cv2

# Method to train custom classifier to recognize face
def train_classifer(name):
    # Read all the images in custom data-set
    path = os.path.join(os.getcwd()+"/data/"+name+"/")
    faces = []
    ids = []
    labels = []
    pictures = {}

    # Store images in a numpy format and ids of the user on the same index in imageNp and id lists
    for root,dirs,files in os.walk(path):
        pictures = files
    for pic in pictures :

        imgpath = path+pic
        img = Image.open(imgpath).convert('L')
        imageNp = np.array(img, 'uint8')
        id = int(pic.split(name)[0])
        #names[name].append(id)
        faces.append(imageNp)
        ids.append(id)

    ids = np.array(ids)

    #Train and save classifier
    clf = cv2.face.LBPHFaceRecognizer_create()
    clf.train(faces, ids)
    clf.write("./data/classifiers/"+name+"_classifier.xml")
```

Code for huffman.py

```
import re
import numpy as np
from PIL import Image
print("Huffman Compression Program")
print("================================================================")
h = int(input("Enter 1 if you want to input an colour image file, 2 for default gray scale case:"))
if h == 1:
    file = input("Enter the filename:")
    my_string = np.asarray(Image.open(file),np.uint8)
    shape = my_string.shape
    a = my_string
    print ("Enetered string is:",my_string)
    my_string = str(my_string.tolist())
elif h == 2:
    array = np.arange(0, 737280, 1, np.uint8)
    my_string = np.reshape(array, (1024, 720))
    print ("Enetered string is:",my_string)
    a = my_string
    my_string = str(my_string.tolist())

else:
    print("You entered invalid input")                # taking user input

letters = []
only_letters = []
for letter in my_string:
    if letter not in letters:
        frequency = my_string.count(letter)           #frequency of each letter repetition
        letters.append(frequency)
        letters.append(letter)
      only_letters.append(letter)

nodes = []
while len(letters) > 0:
   nodes.append(letters[0:2])
   letters = letters[2:]                    # sorting according to frequency
nodes.sort()
huffman_tree = []
huffman_tree.append(nodes)                      #Make each unique character as a leaf node

def combine_nodes(nodes):
   pos = 0
   newnode = []
   if len(nodes) > 1:
      nodes.sort()
      nodes[pos].append("1")                # assigning values 1 and 0
      nodes[pos+1].append("0")
      combined_node1 = (nodes[pos] [0] + nodes[pos+1] [0])
      combined_node2 = (nodes[pos] [1] + nodes[pos+1] [1])  # combining the nodes to generate pathways
      newnode.append(combined_node1)
      newnode.append(combined_node2)
      newnodes=[]
```

```
        newnodes.append(newnode)
        newnodes = newnodes + nodes[2:]
        nodes = newnodes
        huffman_tree.append(nodes)
        combine_nodes(nodes)

    return huffman_tree                    # huffman tree generation

newnodes = combine_nodes(nodes)

huffman_tree.sort(reverse = True)
print("Huffman tree with merged pathways:")

checklist = []
for level in huffman_tree:
    for node in level:
        if node not in checklist:
            checklist.append(node)
        else:
            level.remove(node)
count = 0
for level in huffman_tree:
    print("Level", count,":",level)          #print huffman tree
    count+=1
print()

letter_binary = []
if len(only_letters) == 1:
    lettercode = [only_letters[0], "0"]
    letter_binary.append(letter_code*len(my_string))
else:
    for letter in only_letters:
        code =""
        for node in checklist:
            if len (node)>2 and letter in node[1]:        #genrating binary code
                code = code + node[2]
        lettercode =[letter,code]
        letter_binary.append(lettercode)
print(letter_binary)
print("Binary code generated:")
for letter in letter_binary:
    print(letter[0], letter[1])

bitstring =""
for character in my_string:
    for item in letter_binary:
        if character in item:
            bitstring = bitstring + item[1]
binary ="0b"+bitstring
print("Your message as binary is:")
                        # binary code generated

uncompressed_file_size = len(my_string)*7
compressed_file_size = len(binary)-2
print("Your original file size was", uncompressed_file_size,"bits. The compressed size
is:",compressed_file_size)
print("This is a saving of ",uncompressed_file_size-compressed_file_size,"bits")
output = open("compressed.txt","w+")
print("Compressed file generated as compressed.txt")
output = open("compressed.txt","w+")
print("Decoding.....")
output.write(bitstring)
```

```
bitstring = str(binary[2:])
uncompressed_string =""
code =""
for digit in bitstring:
   code = code+digit
   pos=0                              #iterating and decoding
   for letter in letter_binary:
      if code ==letter[1]:
         uncompressed_string=uncompressed_string+letter_binary[pos] [0]
         code=""
      pos+=1

print("Your UNCOMPRESSED data is:")
if h == 1:
   temp = re.findall(r'\d+', uncompressed_string)
   res = list(map(int, temp))
   res = np.array(res)
   res = res.astype(np.uint8)
   res = np.reshape(res, shape)
   print(res)
   print("Observe the shapes and input and output arrays are matching or not")
   print("Input image dimensions:",shape)
   print("Output image dimensions:",res.shape)
   data = Image.fromarray(res)
```

CHAPTER 5

ADVANTAGES AND DISADVANTAGES

CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

In this paper, we are improving the performance of the image compression. For improve the performance of the image compression, we are improving the parameter of MSE(Mean Square Error ,

PSNR(Peak Signal to Noise Ratio) , NK(Normalized Cross Correlation) , AD(Average Difference) , SC(Structural Content) , MD(Maximum Difference) , NAE(Normalized Absolute Error). As we can check from the results session, PSNR and NK value is getting increase and remaining all the values are getting the decrease . For improving the performance of the image compression, we have to decrease the value of NAE, MD, SC, AD, MSE and PSNR, NK values we have to increase.

5.2 Future Scope

For further enhance the performance of the image compression and decompression, can be done by other lossless methods of image compression because as it is concluded above, that the result of the decompressed image is almost same as that of the input image, so it indicates that there is no loss of information during transmission. So other methods of image compression, any of the type i.e., lossless or lossy can be carried out as namely JPEG method, LZW coding, etc. Use of different metrics can also take place to evaluate the performance of compression algorithms.

5.3 Advantages:-

Smaller file sizes

Use image compression on a picture and in almost every case you'll end up with a smaller file size. This means images take up less space on your computer, tablet, phone or digital camera memory card, and if you're working with hundreds of pictures at once then the difference can really begin to add up. Smaller file sizes also make pictures easier to work with in photo editing software -- the smaller the picture, the less RAM and CPU time required to process it, whether you are trying to make alterations through a program on your computer or an app on your phone.

Faster transfers

Smaller file sizes not only mean less space taken up on disk and in memory, but faster transfers too -- whether you're loading a picture from your hard drive or displaying it on a Web page, a compressed file will appear more quickly than an uncompressed one. Compressed pictures are almost always used on the Web, where transfer speed is more important than quality, but it can also make a significant difference offline if you need to back up many folders of pictures to an external hard drive**.**

5.4 Disadvantages:-

Reduction in quality

The reduction in file size when image compression is applied comes at a cost, and that's a reduction in the overall quality of the image. When using a low level of compression with a format such as JPEG, the difference can be barely noticeable, but this varies depending on the type of image you're working with (the way that light and colours combine in the picture, for example) and the level of compression you've set. Digital photos can often appear very similar even when compressed, because they don't deal with straight, fixed lines and include blocks of pixels of similar colour close to each other (making the pictures easier to compress more effectively).

- You may not be aware of it, but you're probably using image compression whenever you take a snap with your mobile phone or upload a picture to a social network.
- When using a low level of compression with a format such as JPEG, the difference can be barely noticeable, but this varies depending on the type of image you're working with (the way that light and colours combine in the picture, for example) and the level of compression you've set.

REFERENCES

1) L. Agostini, S. Bampi, “Pipelined Fast 2-D DCT Architecture for

2) JPEG Image Compression” Proceedings of the 14th Annual

3) Symposium on Integrated Circuits and Systems Design, Pirenopolis, Brazil. IEEE Computer Society 2001. pp 226-231.

4) Y. Arai, T. Agui, M. Nakajima. “A Fast DCT-SQ Scheme for Images”. Transactions of IEICE, vol. E71, nÂ. 11, 1988, pp. 1095-1097.

5) [3] D. Trang, N. Bihn, “A High-Accuracy and High-Speed 2-D 8x8

6) Discrete Cosine Transform Design”. Proceedings of ICGCRCICT

7) 2010, vol. 1, 2010, pp. 135-138

8) I. Basri, B. Sutopo, “Implementasi 1D-DCT Algoritma Feig-Winograd di FPGA Spartan-3E (Indonesian)”. Proceedings of CITEE 2009, vol. 1, 2009, pp. 198-203

9) E. Magli, “The JPEG Family of Coding Standard,” Part of “Document and Image Compression”, New York: Taylor and Francis, 2004.

10) Wallace, G. K. ,''The JPEG Still Picture Compression Standard'', Communications of the ACM, Vol. 34, Issue 4, pp.30-44. 1991.

11) Sun, M., Ting C., and Albert M., ‘‘VLSI Implementation of a 16 X

12) 16 Discrete Cosine Transform’’, IEEE Transactions on Circuits and Systems, Vol. 36, No. 4, April 1989.

13) Xilinx, Inc., “Spartan-3E FPGA Family : Data Sheet ”, Xilinx

14) Corporation, 2009.

15) Omnivision, Inc., “OV9620/9120 Camera Chip Data Sheet ”, Xilinx

16) Corporation, 2002.

17) Xilinx, Inc., “2D Discrete Cosine Transform (DCT) V2.0 ”, Logicore Product Specification, Xilinx Corporation, 2002. International Journal of Innovations in Engineering and Technology (IJIET) Vol.

18) K. Deepthi: “Design and Implementation of JPEG Image Compression and Decompression”. International Journal of Innovations in Engineering and Technology (IJIET) CVSR College of Engineering, Hyderabad, Andhra Pradesh, India Vol. 2 Issue 1 February (2013).

19) Deepak Singla and Rupali Syal: “Data Security Using LSB & DCT Steganography in Images”. Deepak Singla, Rupali Syal

20) /International Journal Of Computational Engineering Research/ ISSN: 2250–3005. IJCER | Mar-Apr 2012 | Vol. 2 | Issue No.2 |359-364.

21) Lei Wang, Jiaji Wu, Licheng Jiao, Li Zhang and Guangming Shi:

22) “Lossy to Lossless Image Compression based on Reversible Integer DCT”. Institute of Intelligent Information Processing, Xidian

23) University, Xi'an 710071, P.R.China 978-1-4244-1764-3/08/$25.00

24) ©(2008) IEEE.

25) Dipti Bhatnagar and Sumit Budhiraja proposed that Image Compression using DCT based Compressive Sensing and Vector Quantization. International Journal of Computer Applications (0975 – 8887) Volume 50– No.20, July (2012).

26) Kiran Bindu, Anita Ganpati and Aman Kumar Sharma:”A Comparative Study of Image Compression Algorithms”. International

27) Journal of Research in Computer Science eISSN 2249-8265 Volume 2 Issue 5 (2012) pp. 37-42.

28) Radha Krishna A.N and G.Shruthi: “ JPEG Encoder using Discrete Cosine Transform & Inverse Discrete Cosine Transform”. IOSR

29) Journal of Electronics and Communication Engineering (IOSR-JECE) e-ISSN: 2278-2834,p- ISSN: 2278-8735. Volume 5, Issue 4 (Mar. - Apr. 2013), PP 51-56.