EMOTION RECOGNITION SYSTEM

MINOR PROJECT REPORT

Submitted in partial fulfilment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

by

Rahul Nebhnani	**Mayank Kathuria**	**Siddharth Kumra**	**Rohan Bisht**
12015602714	**11715602714**	**10915602714**	**12115602714**

Guided by

Ms Pinky Yadav, Asst. Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
NORTHERN INDIA ENGINEERING COLLEGE
(AFFILIATED TO GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY, DELHI)
NEW DELHI – 110053
DEC 2017

CANDIDATE'S DECLARATION

It is hereby certified that the work which is being presented in the B. Tech Minor Project Report entitled **"EMOTION RECOGNITION SYSTEM"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** and submitted in the **Department of Computer Science & Engineering** of **Northern India Engineering College, New Delhi (Affiliated to Guru Gobind Singh Indraprastha University, Delhi)** is an authentic record of our own work carried out during a period from **August 2017 to November 2017** under the guidance of **Ms. Pinky Yadav, Asst. Prof., Deptt. Of Computer Science and Engineering.**

The matter presented in the B. Tech Minor Project Report has not been submitted by me for the award of any other degree of this or any other Institute.

Rahul Nebhnani	**Mayank Kathuria**	**Siddharth Kumra**	**Rohan Bisht**
12015602714	**11715602714**	**10915602714**	**12115602714**

This is to certify that the above statement made by the candidate(s) is correct to the best of my knowledge. They are permitted to appear in the External Minor Project Examination.

Ms Pinky Yadav
Asst. Professor (CSE)

Dr. Saurabh Gupta
Head of Deptt., CSE

The B. Tech. Minor Project Viva-Voice Examination of **Mayank Kathuria(117015602714), Rahul Nebhnani(120015602714), Siddharth Kumra(109015602714) and Rohan Bisht(121015602714)** has been held on ……………………………….

Dr. Anupam Kumar Sharma
Project Coordinator

(Signature of External Examiner)

ABSTRACT

The human face plays a prodigious role for automatic recognition of emotion in the field of identification of human emotion and the interaction between human and computer for some real application like driver state surveillance, personalized learning, health monitoring etc. Most reported facial emotion recognition systems, however, are not fully considered subject-independent dynamic features, so they are not robust enough for real life recognition tasks with subject (human face) variation.

For human-computer interaction facial expression makes a platform for non-verbal communication. The emotions are effectively changeable happenings that are evoked as a result of impelling force. So in real life application, detection of emotion is very challenging task. Facial expression recognition system requires to overcome the human face having multiple variability such as color, orientation, expression, posture and texture so on.

In our system we train our dataset on different facial expressions viz happy, angry, surprise, disgust. They are trained by making use of FisherFace Recognizer of OpenCV. The live feed of the subject is taken from web cam and then his/her expression is then predicted by the trained system. According to the type of the expressions the tasks are automated such as Emoji Face Swap and Change of Wallpapers.

ACKNOWLEDGEMENT

We express our deepest gratitude to **Ms Pinky Yadav**, Asst. Professor, Department of Computer Science & Engineering for her valuable guidance and suggestion throughout our project work. We are thankful to **Dr. Anupam Sharma,** Project Coordinator for his valuable guidance.

We would like to extend our sincere thanks to **Dr. Saurabh Gupta, Head of the Department,** for his time to time suggestions to complete our project work. We are also thankful to **Prof. (Dr.) G. P. Govil, Director** for providing us the facilities to carry out our project work.

Rahul Nebhnani	**Mayank Kathuria**	**Siddharth Kumra**	**Rohan Bisht**
12015602714	**11715602714**	**10915602714**	**12115602714**

Table of Contents

List of Figures

List of Tables

1.
Introduction

1.1 What is Machine Learning?

Machine learning is when a computer uses an algorithm to understand the data it has been given and recognizes patterns. We call the process of learning “training,” and the output that this process produces is called a “model.” A model can be provided new data and reason against it based on what the model has previously learned. Machine learning models determine a set of rules using vast amounts of computing power that a human brain would be incapable of processing. The more data a machine learning model is fed, the more complex the rules and the more accurate the predictions. Whereas a statistical model is likely to have an inherent logic that can be understood by most people, the rules created by machine learning are often beyond human comprehension because our brains are incapable of digesting and analyzing enormous data sets.
The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly. [1]

1.1.1 Some Machine Learning Methods

Machine learning algorithms are often categorized as supervised or unsupervised.

- **Supervised machine learning** algorithms can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.
- In contrast, **unsupervised machine learning** algorithms are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn’t figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.
- **Semi-supervised machine** learning algorithms fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring unlabeled data generally doesn’t require additional resources.
- **Reinforcement machine learning** algorithms is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search

and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information. [1]

1.2 Emotion Recognition

Trying to interpret a person's emotional state in a nonverbal form, usually requires decoding his/hers facial expressions. Often, body language and especially facial expressions, tells us more than words about a person's state of mind.

1.2.1 The Importance of Emotion Recognition

Understanding the human facial expressions and the study of expressions has many aspects, from computer analysis, emotion recognition, lie detectors, airport security, nonverbal communication and even the role of expressions in art.

Improving the skills of reading expressions is an important step towards successful relations. [2]

1.2.2 Expressions and Emotions

Expressions and emotions go hand in hand, i.e. special combinations of face muscular actions reflect a particular emotion. For certain emotions, it is very hard, and maybe even impossible, to avoid it's fitting facial expression.

For example, a person who is trying to ignore his boss's annoying offensive comment by keeping a neutral expression might nevertheless show a brief expression of anger. This phenomenon of a brief, involuntary facial expression shown on the face of humans according to emotions experienced is called 'micro-expression'. Micro-expressions express the seven universal emotions: happiness, sadness, anger, surprise, contempt, fear and disgust. Micro-expression lasts only 1/25-1/15 of a second. Nonetheless, capturing it can illuminate one's real feelings, whether he wants it or not. That is exactly what Paul Ekman did. [2]

1.2.3 A Brief History

Back in the 80's, Ekman was already known as a specialist for study of facial expressions, when approached by a psychiatrist, asking if Ekman has the ability to detect liars. The psychiatrist wanted to detect if a patient is lying by threatening to suicide. Ekman watched a tape of a patient over and over again, looking for a clue until he found a split second of desperation, meaning that

the patient's threat wasn't empty. Since then, Ekman have found those critical split seconds in almost every liar's documentation. The leading character in the TV series "Lie to me" is based on Paul Ekman himself, the man who dedicated his life to read people's expressions- the "human polygraph".
The research of facial expressions and emotions began many years before Ekman's work. Charles Darwin published his book, called "The Expression of the Emotions in Man and Animals" in 1872. This book was dedicated to nonverbal patterns in humans and animals and to the source of expressions.
Darwin's two former books- "The Descent of Man, and Selection in Relation to Sex" and "On the Origin of Species" represented the idea that man did not came into existence in his present condition, but in a gradual process- Evolution.
This was, of course, a revolutionary theory since in the middle of the 19th century no one believed that man and animal "obey to the same rules of nature".
Darwin's work attempted to find parallels between behaviors and expressions in animals and humans. Ekman's work supports Darwin's theory about universality of facial expressions, even across cultures.
The main idea of "The Expression of the Emotions in Man and Animals" is that the source of nonverbal expressions of man and animals is functional, and not communicative, as we may have thought. This means that facial expressions creation was not for communication purposes, but for something else.
An important observation was that individuals who were born blind had similar facial expressions to individuals who were born with the ability to see.
This observation was intended to contradict Sir Charles Bell's idea (a Scottish surgeon, anatomist, neurologist and philosophical theologian, who influenced Darwin's work), who claimed that human facial muscles were created to provide humans the unique option to express emotions, meaning, for communicational reasons.

According to Darwin, there are three "chief principles", which are three general principles of expression:

1. The first one is called "principle of serviceable habits". He described it as a habit that was reinforced at the beginning and then inherited by offspring.
   For example: he noticed a serviceable habit of raising the eyebrows in order to increase the vision field. He connected it to a person who is trying to remember something, while performing those actions, as though he could "see" what he is trying to remember.
2. The second principle is called "antithesis". Darwin suggested that some actions or habits might not be serviceable themselves, but carried out only because they are opposite in nature to a serviceable habit.
3. The third principle is called "The principle of actions due to the constitution of the Nervous System". This principle is independent from will or a certain extent of habit. [2]

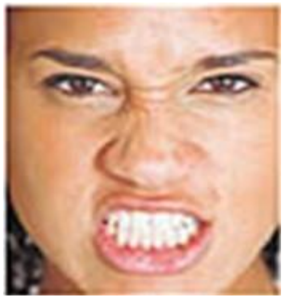
1.2.4 The "Antithesis" Principle

As mentioned earlier, the antithesis phenomenon refers to the way that some muscle movements represent an emotion, and the opposite muscle movements represent the opposite emotion.
An impressive explanation for the facial expression represents 'helplessness' can be done using antithesis:

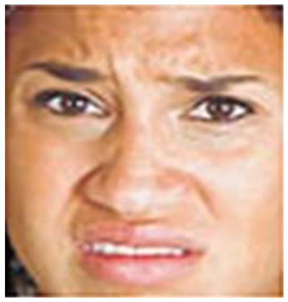
Helplessness body gesture involves hands spreading to the sides, fingers spreading and shoulders shrugging. Its facial expressions involves pulling down the bottom lip and raising eyebrows, like you can see in the followed images.
Darwin explained the features of this expression using the antithesis principle. He discovered that all of those movements opposing to the movements of a man who is ready to face something. The movements of a person who is preparing himself for something will look like that: closed hands and fingers (as if he is preparing for a fight, for example), hands close to the body for protection and the neck is raised and tight. At a helplessness situation the shrugging of the shoulders releases the neck.
As for the face, eyebrows are low (like in a mode of attack or firmness), upper lip might reveal teeth. The functional source of the antithesis can be explained with the investigation of muscles, and to be precise- the antagonist's muscles. Every muscle has an antagonist muscle that performs the opposite movement. Spreading fingers is a movement done by some muscles, and closing the fingers is done by the antagonist muscles. For some expressions we can't always tell just by looking at it, what is the opposite expression, but if we'll look at the muscles involving in the process then it becomes very clear.
An interesting explanation to the antithesis functional source relies on inhibition. If a person or an animal is trying to prevent itself doing a particular action, one way is to use the antagonistic muscles. In fact, when a stimuli signal is send to a muscle, an inhibitory signal is send automatically to the antagonist muscle. Facial expressions that can be explained with antithesis usually relates to aggression and avoiding it. [2]

1.2.5 Facial Expressions Evolutionary Reasons

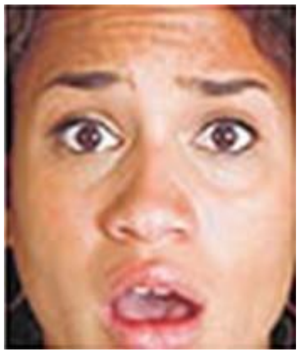
A common assumption is that facial expressions initially served a functional role and not a communicative one. I will try to justify each one of the seven classical expressions with its functional initially role:



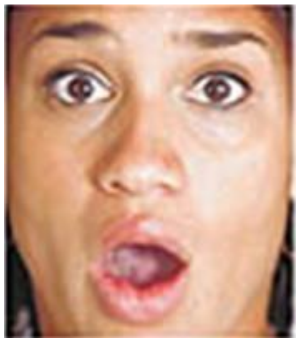
1. **Anger:** involves three main features- teeth revealing, eyebrows down and inner side tightening, squinting eyes. The function is clear- preparing for attack. The teeth are ready to bite and threaten enemies, eyes and eyebrows squinting to protect the eyes, but not closing entirely in order to see the enemy.



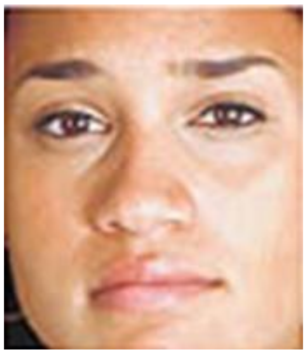
2. **Disgust:** involves wrinkled nose and mouth. Sometimes even involves tongue coming out. This expression mimics a person that tasted bad food and wants to spit it out, or smelling foul smell.



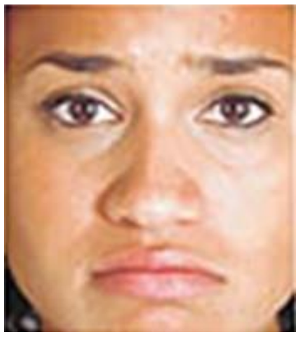
3. **Fear:** involves widened eyes and sometimes open mouth.
The function- opening the eyes so wide is supposed to help increasing the visual field (though studies show that it doesn't actually do so) and the fast eye movement, which can assist finding threats. Opening the mouth enables to breath quietly and by that not being revealed by the enemy.



4. **Surprise:** very similar to the expression of fear.
Maybe because a surprising situation can frighten us for a brief moment, and then it depends whether the surprise is a good or a bad one. Therefore the function is similar.



5. **Sadness:** involves a slight pulling down of lip corners, inner side of eyebrows is rising. Darwin explained this expression by suppressing the will to cry. The control over the upper lip is greater than the control over the lower lip, and so the lower lip drops. When a person screams during a cry, the eyes are closed in order to protect them from blood pressure that accumulates in the face. So, when we have the urge to cry and we want to stop it, the eyebrows are rising to prevent the eyes from closing.



6. **Contempt:** involves lip corner to rise only on one side of the face. Sometimes only one eyebrow rises. This expression might look like half surprise, half happiness.
This can imply the person who receives this look that we are surprised by what he said or did (not in a good way) and that we are amused by it. This is obviously an offensive expression that leaves the impression that a person is superior to another person.

7. **Happiness:** usually involves a smile- both corner of the mouth rising, the eyes are squinting and wrinkles appear at eyes corners. The initial functional role of the smile, which represents happiness, remains a mystery. Some biologists believe that smile was initially a sign of fear. Monkeys and apes clenched teeth in order to show predators that they are harmless. A smile encourages the brain to release endorphins that assist lessening pain and resemble a feeling of wellbeing.
Those good feeling that one smile can produce can help dealing with the fear. A smile can also produce positive feelings for someone who is witness to the smile, and might even get him to smile too.
Newborn babies have been observed to smile involuntarily, or without any external stimuli while they are sleeping. A baby's smile helps his parents to connect with him and get attached to him. It makes sense that for evolutionary reasons, an involuntary smile of a baby helps creating positive feelings for the parents, so they wouldn't abandon their offspring. [1]

1.3 Motivation

In the beginning, facial expression analysis was essentially a research topic for psychologists. However, recent progresses in image processing and pattern recognition have motivated significantly research works on automatic facial expression recognition. In the past, a lot of effort was dedicated to recognize facial expression in still images. For this purpose, many techniques have been applied: neural networks, Gabor wavelets and active appearance models. A very important limitation to this strategy is the fact that still images usually capture the apex of the expression, i.e., the instant at which the indicators of emotion are most marked. In their daily life, people seldom show apex of their facial expression during normal communication with their counterparts, unless for very specific cases and for very brief periods of time. The automatic facial expression recognition system includes:
•Face Detector.
•Facial feature extractor for mouth, left and right eye.
•Facial expression recognizer.

1.4 Objective

This project’s main objective is to come up with a solution of emotion detection by first using algorithms to first detect individual faces from images and then predict the emotion displayed on those faces.
This is achieved through first creating a model to detect emotions. Then train it and finally apply it in real world for emotion recognition. After accurately detecting the emotion exhibited by a particular individual, automated scripts are executed to perform a variety of tasks.

1. A face swap feature : a webcam window is opened where a person's face is shown in real time and an automated script swaps the face of the person with an emoji of the emotion the person is showing
2. Based on the person's emotion the wallpaper of the desktop changes

2.
Project Design and Description

2.1 Facial Feature Detection Using Haar Classifiers

2.1.1 Introduction

The human face poses even more problems than other objects since the human face is a dynamic object that comes in many forms and colors. However, facial detection and tracking provides many benefits. Facial recognition is not possible if the face is not isolated from the background. Human Computer Interaction (HCI) could greatly be improved by using emotion, pose, and gesture recognition, all of which require face and facial feature detection and tracking.
Although many different algorithms exist to perform face detection, each has its own weaknesses and strengths. Some use flesh tones, some use contours, and other are even more complex involving templates, neural networks, or filters. These algorithms suffer from the same problem; they are computationally expensive.
An image is only a collection of color and/or light intensity values. Analyzing these pixels for face detection is time consuming and difficult to accomplish because of the wide variations of shape and pigmentation within a human face. Pixels often require reanalysis for scaling and precision. Viola and Jones devised an algorithm, called Haar Classifiers, to rapidly detect any object, including human faces, using AdaBoost classifier cascades that are based on Haar-like features and not pixels. [3]

2.1.2 Haar Cascade Classifiers

The core basis for Haar classifier object detection is the Haar-like features. These features, rather than using the intensity values of a pixel, use the change in contrast values between adjacent rectangular groups of pixels. The contrast variances between the pixel groups are used to determine relative light and dark areas.
Two or three adjacent groups with a relative contrast variance form a Haar-like feature. Haar-like features, as shown in Figure 2.1 are used to detect an image. Haar features can easily be scaled by increasing or decreasing the size of the pixel group being examined. This allows features to be used to detect objects of various sizes.
The simple rectangular features of an image are calculated using an intermediate representation of an image, called the integral image. The integral image is an array containing the sums of the pixels' intensity values located directly to the left of a pixel and directly above the pixel at location (x, y) inclusive.

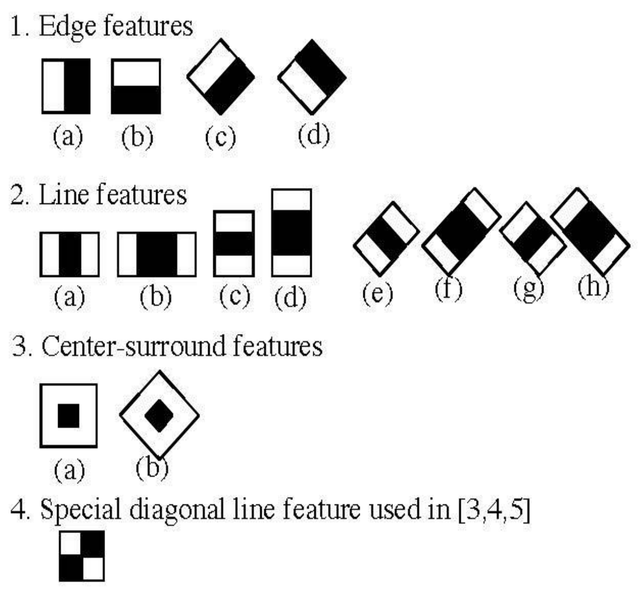

1. Edge features
(a)
(b)
(c)
(d)
2. Line features
(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
3. Center-surround features
(a)
(b)
4. Special diagonal line feature used in [3,4,5]

Figure 2.1 Common Haar Features

So if A[x,y] is the original image and AI[x,y] is the integral image then the integral image is computed as shown in equation 1 and illustrated in Figure 2.2.

$$AI[\ x\ y] = \sum A(x',y') \qquad (1)$$

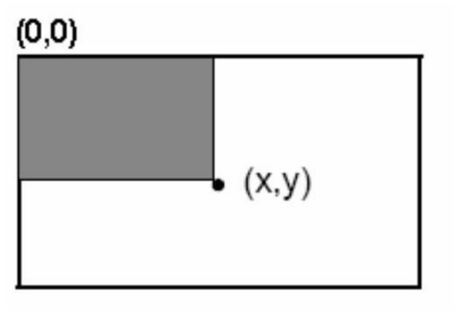


Figure 2.2 Summed Area of Integral Image

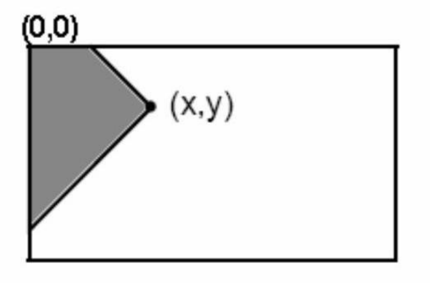


Figure 2.3 Summed Area of Rotated Integral Image

The features rotated by forty-five degrees, as introduced by Lienhart and Maydt, require another intermediate representation called the rotated integral image or rotated sum auxiliary image. The rotated integral image is calculated by finding the sum of the pixels' intensity values that are located at a forty five degree angle to the left and above for the x value and below for the y value. So if A[x,y] is the original image and AR[x,y] is the rotated integral image then the integral image is computed as shown in equation 2.

$$AR[x,y] = \sum A(x',y') \qquad (2)$$

It only takes two passes to compute both integral image arrays, one for each array. Using the appropriate integral image and taking the difference between six to eight array elements forming two or three connected rectangles, a feature of any scale can be computed. Thus calculating a feature is extremely fast and efficient. It also means calculating features of various sizes requires the same effort as a feature of only two or three pixels. The detection of various sizes of the same object requires the same amount of effort and time as objects of similar sizes since scaling requires no additional effort. [3]

2.1.3 Classifiers Cascaded

Although calculating a feature is extremely efficient and fast, calculating all 180,000 features contained within a 24 × 24 sub-image is impractical [Viola 2001, Wilson 2005]. Fortunately, only a tiny fraction of those features are needed to determine if a sub-image potentially contains the desired object [6]. In order to eliminate as many sub-images as possible, only a few of the features that define an object are used when analyzing sub-images. The goal is to eliminate a substantial amount, around 50%, of the sub-images that do not contain the object. This process continues, increasing the number of features used to analyze the sub-image at each stage. The cascading of the classifiers allows only the sub-images with the highest probability to be analyzed for all Haar-features that distinguish an object. It also allows one to vary the accuracy of a classifier. One can increase both the false alarm rate and positive hit rate by decreasing the number of stages. The inverse of this is also true. Viola and Jones were able to achieve a 95% accuracy rate for the detection of a human face using only 200 simple features . Using a 2 GHz computer, a Haar classifier cascade could detect human faces at a rate of at least five frames per second . [3]

2.1.4 Training Classifiers for Facial Features

Detecting human facial features, such as the mouth, eyes, and nose require that Haar classifier cascades first be trained. In order to train the classifiers, this gentle AdaBoost algorithm and Haar feature algorithms must be implemented. Fortunately, Intel developed an open source library devoted to easing the implementation of computer vision related programs called Open Computer Vision Library (OpenCV). The OpenCV library is designed to be used in conjunction with applications that pertain to the field of HCI, robotics, biometrics, image processing, and other areas where visualization is important and includes an implementation of Haar classifier detection and training [8]. To train the classifiers, two set of images are needed. One set contains an image or scene that does not contain the object, in this case a facial feature, which is going to be detected. This set of images is referred to as the negative images. The other set of images, the positive images, contain one or more instances of the object. The location of the objects within the positive images is specified by: image name, the upper left pixel and the height, and width of the object. For training facial features 5,000 negative images with at least a mega-pixel resolution were used for training. These images consisted of everyday objects, like paperclips, and of natural scenery, like photographs of forests and mountains.
In order to produce the most robust facial feature detection possible, the original positive set of images needs to be representative of the variance between different people, including, race, gender, and age. A good source for these images is National Institute of Standards and Technology's

(NIST) Facial Recognition Technology (FERET) database. This database contains over 10,000 images of over 1,000 people under different lighting conditions, poses, and angles. In training each facial feature, 1,500 images were used. These images were taken at angles ranging from zero to forty five degrees from a frontal view. This provides the needed variance required to allow detection if the head is turned slightly. Three separate classifiers were trained, one for the eyes, one for the nose, and one for the mouth. Once the classifiers were trained, they were used to detect the facial features within another set of images from the FERET database. The accuracy of the classifier was then computed as shown in Table 1. With the exception of the mouth classifier, the classifiers have a high rate of detection. However, as implied by, the false positive rate is also quite high. [3]

Facial Feature	Positive Hit Rate	Negative Hit Rate
Eyes	93%	23%
Nose	100%	29%
Mouth	67%	28%

Table 1 Accuracy of Classifiers

2.1.5 Regionalized Detection

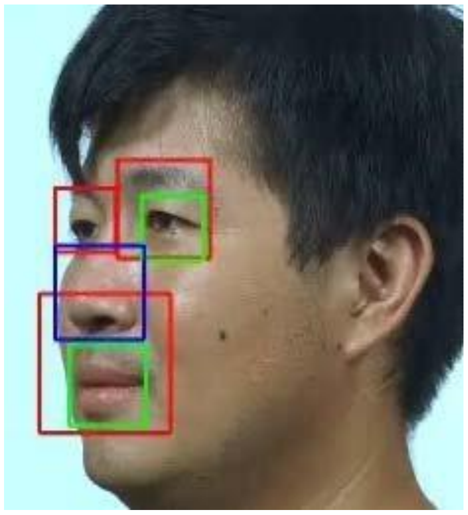


Figure 2.4 Inaccurate Detection : Eyes (red), Nose (blue), and Mouth (green)

Since it is not possible to reduce the false positive rate of the classifier without reducing the positive hit rate, a method besides modifying the classifier training attribute is needed to increase accuracy. The method proposed to is to limit the region of the image that is analyzed for the facial features. By reducing the area analyzed, accuracy will increase since less area exists to produce false positives. It also increases efficiency since fewer features need to be computed and the area of the integral images is smaller. In order to regionalize the image, one must first determine the likely area where a facial feature might exist. The simplest method is to perform facial detection on the image first. The area containing the face will also contain facial features. However, the facial feature cascades often detect other facial features as illustrated in Figure 24. The best method to eliminate extra feature detection is to further regionalize the area for facial feature detection. It can be assumed that the eyes will be located near the top of the head, the nose will be located in the center area and the mouth will be located near the bottom. The upper 5/8 of the face is analyzed for the eyes. This area eliminates all other facial features while still allowing a wide variance in the tilt angle. The center of the face, an area that is 5/8 by 5/8 of the face, was used to for detection of the nose. This area eliminates all but the upper lip of the mouth and lower eyelid. The lower half of the facial image was used to detect the mouth. Since the facial detector used sometimes eliminates the lower lip the facial image was extended by an eighth for mouth detection only. [3]

2.1.6 Results

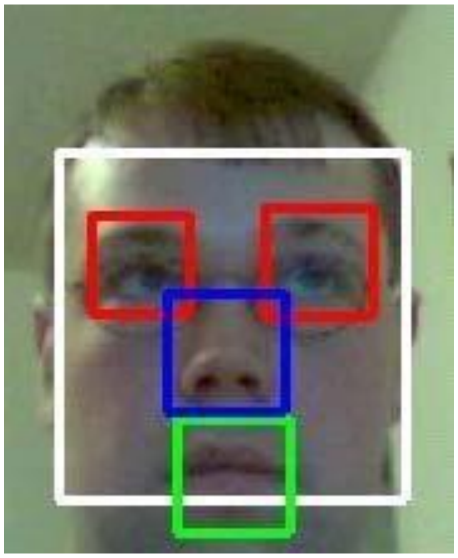


Figure 2.5Detected Objects: Eyes (red), Face (white), Nose (blue), and Mouth (green)

The first step in facial feature detection is detecting the face. This requires analyzing the entire image. The second step is using the isolated face(s) to detect each feature. The result is shown in Figure 2.5. Since each the portion of the image used to detect a feature is much smaller than that of the whole image, detection of all three facial features takes less time on average than detecting the face itself. Using a 1.2GHz AMD processor to analyze a 320 by 240 image, a frame rate of 3 frames per second was achieved. Since a frame rate of 5 frames per second was achieved in facial detection only by using a much faster processor, regionalization provides a tremendous increase in efficiency in facial feature detection. Regionalization also greatly increased the accuracy of the detection. All false positives were eliminated, giving a detection rate of around 95% for the eyes and nose. The mouth detection has a lower rate due to the minimum size required for detection. By changing the height and width parameter to more accurately represent the dimensions of the mouth and retraining the classifier the accuracy should increase the accuracy to that of the other features.

2.1.7 Future Plans

With the successful detection of facial features, the next goal is to research the ability for more precise details, like individual points, of the facial features to be gathered. These points will be use to differentiate general human emotions, like happiness and sadness. Recognition of human emotion would require detection and analysis of the various elements of a human face, like the brow and the mouth, to determine an individual’s current expression. The expression can then be compared to what is considered to be the basic signs of an emotion. This research will be used in the field human-computer interaction to analyze the emotions one exhibits while interacting with a user interface. [3]

2.1.8 Haar-Cascade Detection in OpenCV

OpenCV comes with a trainer as well as detector. If you want to train your own classifier for any object like car, planes etc. you can use OpenCV to create one. Its full details are given here: Cascade Classifier Training.
Here we will deal with detection. OpenCV already contains many pre-trained classifiers for face, eyes, smile etc. Those XML files are stored in opencv/data/haarcascades/ folder. Let's create face and eye detector with OpenCV.

First we need to load the required XML classifiers. Then load our input image (or video) in grayscale mode.

```
import numpy as np
import cv2
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
img = cv2.imread('sachin.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Figure 2.6 Code Snippet for loading image in grayscale

Now we find the faces in the image. If faces are found, it returns the positions of detected faces as Rect(x,y,w,h). Once we get these locations, we can create a ROI for the face and apply eye detection on this ROI (since eyes are always on the face!). [3]

```
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
roi_gray = gray[y:y+h, x:x+w]
roi_color = img[y:y+h, x:x+w]
eyes = eye_cascade.detectMultiScale(roi_gray)
for (ex,ey,ew,eh) in eyes:
cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
cv2.imshow('img',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Figure 2.7 Detection of ROI

Result after application looks like in Figure 2.8.

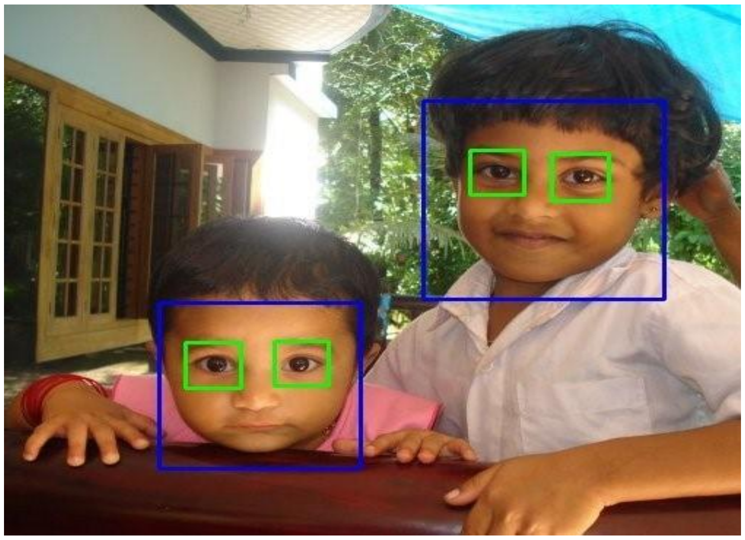


Figure 2.8 Results after application of code

2.2 Fisherface Algorithm

2.2.1 Introduction

A key problem in computer vision, pattern recognition and machine learning is to define an appropriate data representation for the task at hand.
One way to represent the input data is by finding a subspace which represents most of the data variance. This can be obtained with the use of Principal Components Analysis (PCA). When applied to face images, PCA yields a set of eigenfaces. These eigenfaces are the eigenvectors associated to the largest eigenvalues of the covariance matrix of the training data. The eigenvectors thus found correspond to the least-squares (LS) solution. This is indeed a powerful way to represent the data because it ensures the data variance is maintained while eliminating unnecessary existing correlations among the original features (dimensions) in the sample vectors.
When the goal is classification rather than representation, the LS solution may not yield the most desirable results. In such cases, one wishes to find a subspace that maps the sample vectors of the same class in a single spot of the feature representation and those of different classes as far apart from each other as possible. The techniques derived to achieve this goal are known as discriminant analysis (DA).
The most known DA is Linear Discriminant Analysis (LDA), which can be derived from an idea suggested by R.A. Fisher in 1936. When LDA is used to find the subspace representation of a set of face images, the resulting basis vectors defining that space are known as Fisherfaces. [4]

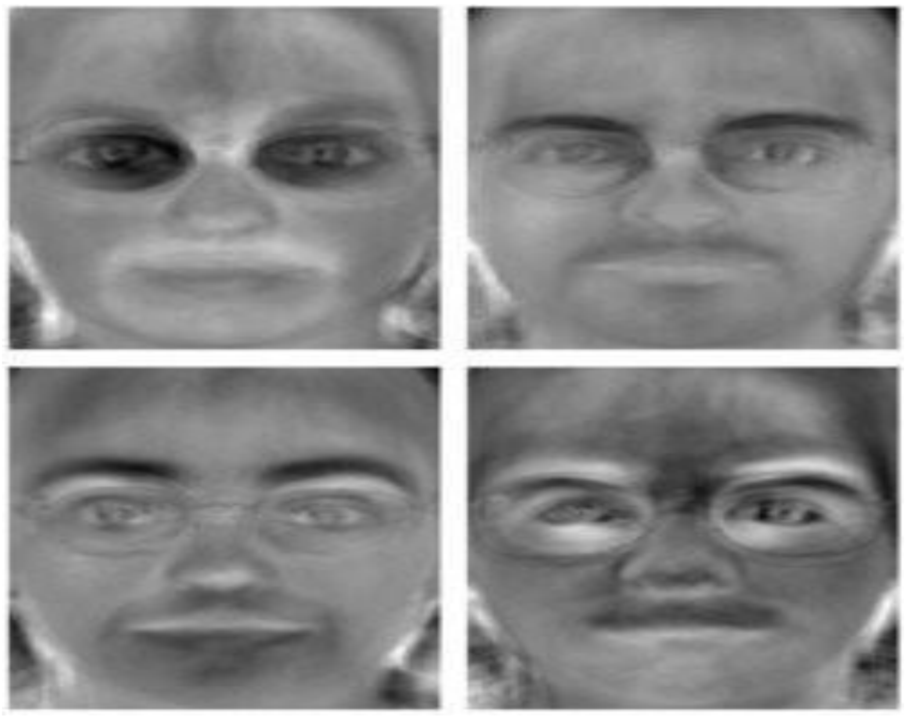


Figure 2.9 Shown here are the first four Fisherfaces from a set of 100 classes (subjects).

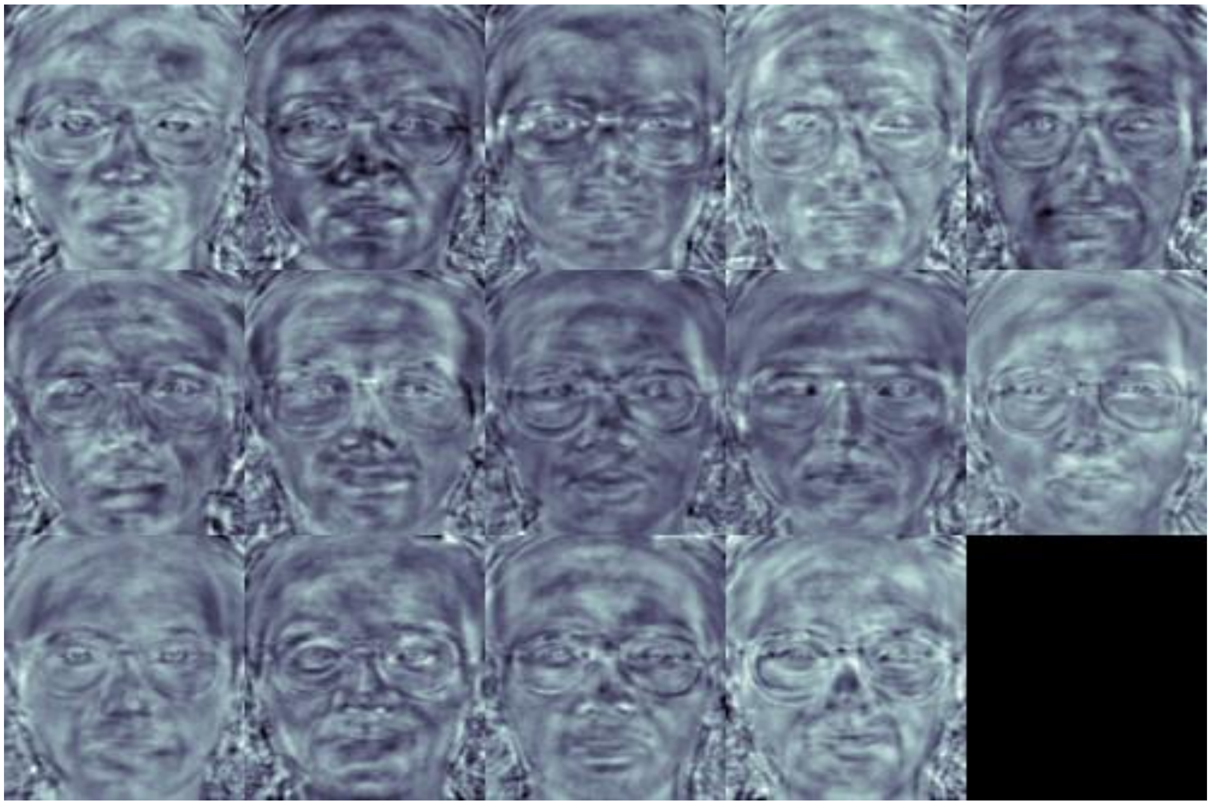


Figure 2.10 Fisherfaces for Yale Database

2.2.2 Discriminant Scores

To compute the Fisherfaces, we assume the data in each class is normally distributed. We denote the multivariate Normal distribution as $Ni(\mu i,\Sigma i)$, with mean μi and covariance matrix Σi , and its probability density function is $fi(\mathrm{x}|\mu i,\Sigma i)$.
In the C class problem, we have $Ni(\mu i,\Sigma i)$, with $i=1,\ldots,C$. Given these Normal distributions and their class prior probabilities Pi , the classification of a test sample x is given by comparing the log-likelihoods of $fi(\mathrm{x}|\mu i,\Sigma i)Pi$ for all i . That is, $\mathrm{argmin}1\leq i\leq Cdi(\mathrm{x})$,
where $di(\mathrm{x})=(\mathrm{x}-\mu i)T\Sigma-1i(\mathrm{x}-\mu i)+ln|\Sigma i|-2\ln Pi$ are known as the discriminant scores of each class. The discriminant scores thus defined yield the Bayes optimal solution.
The discriminant scores generally result in quadratic classification boundaries between classes. However, for the case where all the covariance matrices are the same, $\Sigma i=\Sigma$, $\forall i$, the quadratic parts of di cancel out, yielding linear classifiers. These classifiers are called linear discriminant bases. Hence, the name of Linear Discriminant Analysis. The case where all the covariance's are identical is known as homoscedastic Normal distributions.
Assume that $C=2$ and that the classes are homoscedastic Normals. Project the sample feature vectors onto the one-dimensional subspace orthogonal to the classification hyperplane given by the discriminant score. It follows that the number of misclassified samples in the original space of p dimensions and in this subspace of just one dimension are the same. This is easily verifiable. Since the classification boundary is linear, all the samples that where on one side of the space will remain on

the same side of the 1-dimensions subspace. This important point was first noted by R.A. Fisher and has allowed us to define the LDA algorithm and Fisherfaces. [4]

2.2.3 Computing the Fisherfaces

The theoretical argument given in the preceding section shows how to obtain the Bayes optimal solution for the 2-class homoscedastic case. In general, we will have more than 2-classes. In such a case, we reformulate the above stated problem as that of minimizing within-class differences and maximizing between-class distances.
Within class differences can be estimated using the within-class scatter matrix, given by
$\mathbf{S}w=\sum Cj=1\sum nji=1(\mathbf{x}ij-\mu j)(\mathbf{x}ij-\mu j)T,$
where $\mathbf{x}ij$ is the ith sample of class j , μj is the mean of class j , and nj the number of samples in class j .
Likewise, the between class differences are computed using the between-class scatter matrix,
$\mathbf{S}b=\sum Cj=1(\mu j-\mu)(\mu j-\mu)T,$
where μ represents the mean of all classes.
We now want to find those basis vectors $\mathbf{V}$ where $\mathbf{S}w$ is minimized and $\mathbf{S}b$ is maximized, where $\mathbf{V}$ is a matrix whose columns $\mathbf{v}i$ are the basis vectors defining the subspace. These are given by,
$|\mathbf{V}T\mathbf{S}b\mathbf{V}||\mathbf{V}T\mathbf{S}w\mathbf{V}|.$
The solution to this problem is given by the generalized eigenvalue decomposition
$\mathrm{S}b\mathrm{V}=\mathrm{S}w\mathrm{V}\Lambda,$
where $\mathbf{V}$ is (as above) the matrix of eigenvectors and $\mathbf{\Lambda}$ is a diagonal matrix of corresponding eigenvalues.
The eigenvectors of $\mathbf{V}$ associated to non-zero eigenvalues are the Fisherfaces. There is a maximum of $C-1$ Fisherfaces. This can be readily seen from the definition of $\mathbf{S}b$. Note that in our definition, $\mathbf{S}b$ is a combination of C feature vectors. Any C vectors define a subspace of $C-1$ or less dimensions. The equality holds when these vectors are linearly independent from one another. Figure 1 shows the first four Fisherfaces obtained when using the defined algorithm on a set of frontal face image of 100 different subjects. Images were selected to have a neutral expression. [4]

2.2.4 Technicalities

To obtain the Fisherfaces, we need to compute the inverse of $\mathbf{S}w$, i.e., $\mathbf{S}-1w$. If the sample feature vectors are defined in a p-dimensional space and p is larger than the total number of samples n , then $\mathbf{S}w$ is singular. There are two typically used solutions to this problem. In the first solution, we project the sample vectors (or, equivalently, the between- and within-class scatter matrices) onto the PCA space of r dimensions, with $r\leq rank(\mathbf{S}w)$ and compute the Fisherfaces in this PCA space. The second solution is to add a regularising term to $\mathbf{S}w$. That is, $\mathbf{S}w+\epsilon\mathbf{I}$, where $\mathbf{I}$ is the identity matrix and ϵ is a small constant.
One can also substitute the between- and within-class scatter matrices for other measurements that do a similar job. First, note that these two matrices are symmetric and positive semi-definite. Hence, each defines a metric. This means we can substitute these matrices with others as far as they define metrics whose goals are to minimize within-class variance and maximize between-

class distances. For example, we can substitute the within-class scatter matrix for the sample covariance matrix.
The Fisherfaces obtained with the approach described thus far are based on the linear assumption mentioned above. This assumption holds when the classes are homoscedastic Normals. In general, this assumption is violated. To resolve this problem, one can add another metric into the equation. The goal of this new metric is to map the original heteroscedastic (meaning with different covariances) problem into a homoscedastic one. This mapping function can be converted into a kernel mapping thanks to the Representer's Theorem. And, the metric is given by the Gram (or Kernel) matrix. For this reason, this alternative method is called Kernel LDA (or, KLDA for short). Several authors have also defined heteroscedastic measures of within- and between-class differences. Yet, another alternative to lessen the Normal assumption is to represent the samples in each class as a mixture of Normal distributions. In this approach, the trick is how to determine the number of mixtures per class. A popular solution is given by the algorithm Subclass Discriminant Analysis (SDA) and its kernel extension KSDA. Kernel methods are generally preferred when the number of training samples is sufficiently large to facilitate the learning of the nonlinear mapping. [4]

2.2.5 Fisherface extensions

Recently, Two-dimensional LDA (2DLDA), a tensor extension of LDA, is proposed. Different from the LDA which requires the input patterns to be converted to one-dimensional vectors, the 2DLDA directly extracts the proper features from image matrices based on Fisher’s Linear Discriminant Analysis. [4]

2.2.6 Algorithmic Description

The Fisherfaces algorithm we are going to implement basically goes like this:

- Construct the Imagematrix X with each column representing an image. Each image is a assigned to a class in the corresponding class vector C.
- Project X into the (N-c)-dimensional subspace as P with the rotation matrix WPcaidentified by a Principal Component Analysis, where

  (a) N is the number of samples in X
  (b) c is unique number of classes (length(unique(C)))

- Calculate the between-classes scatter of the projection P as Sb = \sum_{i=1}^{c} N_i*(mean_i - mean)*(mean_i - mean)^T, where

  (a) mean is the total mean of P
  (b) mean is the mean of class i in P
  (c) N_i is the number of samples for class i

- Calculate the within-classes scatter of P as Sw = \sum_{i=1}^{c} \sum_{x_k \in X_i} (x_k - mean_i) * (x_k - mean_i)^T, where

    (a) X_i are the samples of class i
    (b) x_k is a sample of X_i
    (c) mean_i is the mean of class i in P

- Apply a standard Linear Discriminant Analysis and maximize the ratio of the determinant of between-class scatter and within-class scatter. The solution is given by the set of generalized eigenvectors Wfld of Sb and Sw corresponding to their eigenvalue. The rank of Sb is atmost (c-1), so there are only (c-1) non-zero eigenvalues, cut off the rest.

- Finally obtain the Fisherfaces by W = WPca * Wfld.

The Fisherfaces method learns a class-specific transformation matrix, so they do not capture illumination as obviously as the Eigenfaces method. The Discriminant Analysis instead finds the facial features to discriminate between the persons. It's important to mention, that the performance of the Fisherfaces heavily depends on the input data as well. Practically said: if you learn the Fisherfaces for well-illuminated pictures only and you try to recognize faces in bad-illuminated scenes, then method is likely to find the wrong components (just because those features may not be predominant on bad illuminated images). This is somewhat logical, since the method had no chance to learn the illumination.

The Fisherfaces allow a reconstruction of the projected image, just like the Eigenfaces did. But since we only identified the features to distinguish between subjects, you can't expect a nice reconstruction of the original image. For the Fisherfaces method we'll project the sample image onto each of the Fisherfaces instead. So you'll have a nice visualization, which feature each of the Fisherfaces describes. [5]

2.2.7 Automation

Automation can be defined as the technology by which a process or procedure is performed without human assistance.
In other words, Automation or automatic control, is the use of various control systems for operating equipment such as machinery, processes in factories, boilers and heat treating ovens, switching on telephone networks, steering and stabilization of ships, aircraft and other applications and vehicles with minimal or reduced human intervention, with some processes have been completely automated.
Automation has been achieved by various means including mechanical, hydraulic, pneumatic, electrical, electronic devices and computers, usually in combination. Complicated systems, such as modern factories, airplanes and ships typically use all these combined techniques. The benefit of automation include labor savings, savings in electricity costs, savings in material costs, and improvements to quality, accuracy and precision.

The term automation, inspired by the earlier word automatic (coming from automaton), was not widely used before 1947, when Ford established an automation department. It was during this time that industry was rapidly adopting feedback controllers, which were introduced in the 1930s. [6]

2.2.8 Open-loop and closed-loop (feedback) control

Fundamentally, there are two types of control loop; open loop control, and closed loop (feedback) control.
In open loop control, the control action from the controller is independent of the "process output" (or "controlled process variable"). A good example of this is a central heating boiler controlled only by a timer, so that heat is applied for a constant time, regardless of the temperature of the building. (The control action is the switching on/off of the boiler. The process output is the building temperature).
In closed loop control, the control action from the controller is dependent on the process output. In the case of the boiler analogy this would include a thermostat to monitor the building temperature, and thereby feed back a signal to ensure the controller maintains the building at the temperature set on the thermostat. A closed loop controller therefore has a feedback loop which ensures the controller exerts a control action to give a process output the same as the "Reference input" or "set point". For this reason, closed loop controllers are also called feedback controllers.
The definition of a closed loop control system according to the British Standard Institution is 'a control system possessing monitoring feedback, the deviation signal formed as a result of this feedback being used to control the action of a final control element in such a way as to tend to reduce the deviation to zero.'
Likewise, a Feedback Control System is a system which tends to maintain a prescribed relationship of one system variable to another by comparing functions of these variables and using the difference as a means of control.
The advanced type of automation that revolutionized manufacturing, aircraft, communications and other industries, is feedback control, which is usually continuous and involves taking measurements using a sensor and making calculated adjustments to keep the measured variable within a set range. The theoretical basis of closed loop automation is control theory. [6]

2.2.9 Control actions

2.2.9.1 Discrete control (on/off)

One of the simplest types of control is on-off control. An example is the thermostat used on household appliances which either opens or closes an electrical contact. (Thermostats were originally developed as true feedback-control mechanisms rather than the on-off common household appliance thermostat.)Sequence control, in which a programmed sequence of discrete operations is performed, often based on system logic that involves system states. An elevator control system is an example of sequence control. [6]

2.2.9.2 PID controller

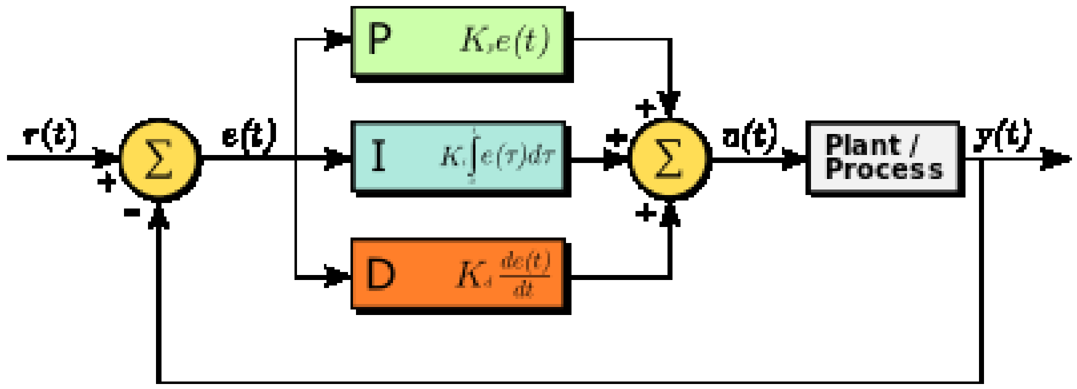


Figure 2.11 PID Controller Architecture

A block diagram of a PID controller in a feedback loop, r(t) is the desired process value or "set point", and y(t) is the measured process value. A proportional–integral–derivative controller (PID controller) is a control loop feedback mechanism (controller) widely used in industrial control systems. A PID controller continuously calculates an error value {\displaystyle e(t)} as the difference between a desired set point and a measured process variable and applies a correction based on proportional, integral, and derivative terms, respectively (sometimes denoted P, I, and D) which give their name to the controller type.

The theoretical understanding and application dates from the 1920s, and they are implemented in nearly all analogue control systems; originally in mechanical controllers, and then using discrete electronics and latterly in industrial process computers. [6]

2.2.9.3 Sequential control and logical sequence or system state control

Sequential control may be either to a fixed sequence or to a logical one that will perform different actions depending on various system states. An example of an adjustable but otherwise fixed sequence is a timer on a lawn sprinkler.

States refer to the various conditions that can occur in a use or sequence scenario of the system. An example is an elevator, which uses logic based on the system state to perform certain actions in response to its state and operator input. For example, if the operator presses the floor n button, the system will respond depending on whether the elevator is stopped or moving, going up or down, or if the door is open or closed, and other conditions. An early development of sequential control was relay logic, by which electrical relays engage electrical contacts which either start or interrupt power to a device. Relays were first used in telegraph networks before being developed for controlling other devices, such as when starting and stopping industrial-sized electric motors or opening and closing solenoid valves. Using relays for control purposes allowed event-driven control, where actions could be triggered out of sequence, in response to external events. These were more flexible in their response than the rigid single-sequence cam timers. More complicated examples involved maintaining safe sequences for devices such as swing bridge controls, where a lock bolt needed to be disengaged before the bridge could be moved, and the lock bolt could not be released until the safety gates had already been closed.

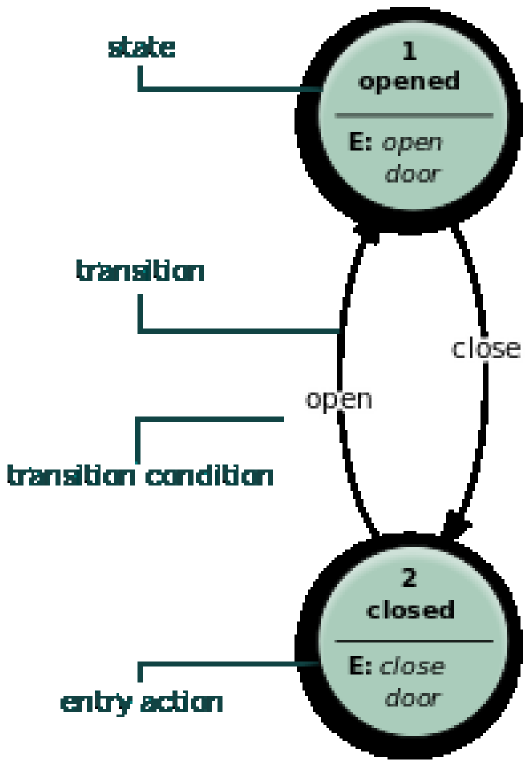


Figure 2.12 Logical Sequence of a Door Mechanism

The total number of relays, cam timers and drum sequencers can number into the hundreds or even thousands in some factories. Early programming techniques and languages were needed to make such systems manageable, one of the first being ladder logic, where diagrams of the interconnected relays resembled the rungs of a ladder. Special computers called programmable logic controllers were later designed to replace these collections of hardware with a single, more easily re-programmed unit.

In a typical hard wired motor start and stop circuit (called a control circuit) a motor is started by pushing a "Start" or "Run" button that activates a pair of electrical relays. The "lock-in" relay locks in contacts that keep the control circuit energized when the push button is released. (The start button is a normally open contact and the stop button is normally closed contact.) Another relay energizes a switch that powers the device that throws the motor starter switch (three sets of contacts for three phase industrial power) in the main power circuit. Large motors use high voltage and experience high in-rush current, making speed important in making and breaking contact. This can be dangerous for personnel and property with manual switches. The "lock in" contacts in the start circuit and the main power contacts for the motor are held engaged by their respective electromagnets until a "stop" or "off" button is pressed, which de-energizes the lock in relay. Commonly interlocks are added to a control circuit. Suppose that the motor in the example is powering machinery that has a critical need for lubrication. In this case an interlock could be added to insure that the oil pump is running before the motor starts. Timers, limit switches and electric eyes are other common elements in control circuits.

Solenoid valves are widely used on compressed air or hydraulic fluid for powering actuators on mechanical components. While motors are used to supply continuous rotary motion, actuators are typically a better choice for intermittently creating a limited range of movement for a mechanical component, such as moving various mechanical arms, opening or closing valves, raising heavy press rolls, applying pressure to presses. [7]

2.2.9.4 Computer control

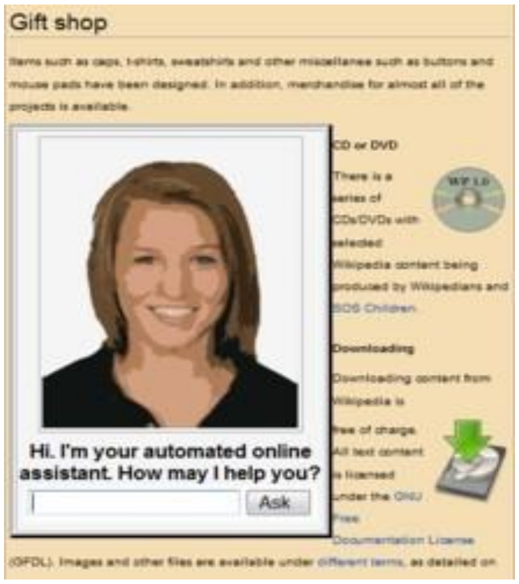
Computers can perform both sequential control and feedback control, and typically a single computer will do both in an industrial application. Programmable logic controllers (PLCs) are a type of special purpose microprocessor that replaced many hardware components such as timers and drum sequencers used in relay logic type systems. General purpose process control computers have increasingly replaced standalone controllers, with a single computer able to perform the operations of hundreds of controllers. Process control computers can process data from a network of PLCs, instruments and controllers in order to implement typical (such as PID) control of many individual variables or, in some cases, to implement complex control algorithms using multiple inputs and mathematical manipulations. They can also analyze data and create real time graphical displays for operators and run reports for operators, engineers and management.

Control of an automated teller machine (ATM) is an example of an interactive process in which a computer will perform a logic derived response to a user selection based on information retrieved from a networked database. The ATM process has similarities with other online transaction processes. The different logical responses are called scenarios. Such processes are typically designed with the aid of use cases and flowcharts, which guide the writing of the software code. [7]

2.2.10 Automation tools

Engineers can now have numerical control over automated devices. The result has been a rapidly expanding range of applications and human activities. Computer-aided technologies (or CAx) now serve as the basis for mathematical and organizational tools used to create complex systems. Notable examples of CAx include Computer-aided design (CAD software) and Computer-aided manufacturing (CAM software). The improved design, analysis, and manufacture of products enabled by CAx has been beneficial for industry.

Information technology, together with industrial machinery and processes, can assist in the design, implementation, and monitoring of control systems. One example of an industrial control system is a programmable logic controller (PLC). PLCs are specialized hardened computers which are frequently used to synchronize the flow of inputs from (physical) sensors and events with the flow of outputs to actuators and events.



Gift shop

Items such as caps, t-shirts, sweatshirts and other miscellanea such as buttons and mouse pads have been designed. In addition, merchandise for almost all of the projects is available.

CD or DVD

There is a series of CDs/DVDs with selected Wikipedia content being produced by Wikipedians and SOS Children

Downloading

Downloading content from Wikipedia is free of charge. All text content is licensed under the GNU Free Documentation License (GFDL). Images and other files are available under different terms, as detailed on

Hi. I'm your automated online assistant. How may I help you?

Figure 2.13 An automated online assistant on a website, with an avatar for enhanced human–computer interaction.

Human-machine interfaces (HMI) or computer human interfaces (CHI), formerly known as *man-machine interfaces*, are usually employed to communicate with PLCs and other computers. Service personnel who monitor and control through HMIs can be called by different names. In industrial process and manufacturing environments, they are called operators or something similar. In boiler houses and central utilities departments they are called stationary engineers. Different types of automation tools exist:

- ANN - Artificial neural network
- DCS - Distributed Control System
- HMI - Human Machine Interface
- SCADA - Supervisory Control and Data Acquisition
- PLC - Programmable Logic Controller
- Instrumentation
- Motion control
- Robotics

When it comes to factory automation, Host Simulation Software (HSS) is a commonly used testing tool that is used to test the equipment software. HSS is used to test equipment performance with respect to Factory Automation standards (timeouts, response time, and processing time). [7]

2.2.11 Cognitive Automation

Cognitive automation is an emerging genus of automation enabled by cognitive computing. Its primary concern is the automation of clerical tasks and workflows that consist of structuring unstructured data.

Cognitive automation relies on multiple disciplines: natural language processing, real-time computing, machine learning algorithms, big data analytics and evidence-based learning. According to Deloitte, cognitive automation enables the replication of human tasks and judgment “at rapid speeds and considerable scale.”

Such tasks include:

- Document redaction
- Data extraction and document synthesis / reporting
- Contract management
- Natural language search
- Customer, employee, and stakeholder onboarding
- Manual activities and verifications
- Follow up and email communications

2.3 Paradox of Automation

The paradox of automation says that the more efficient the automated system, the more crucial the human contribution of the operators. Humans are less involved, but their involvement becomes more critical.

If an automated system has an error, it will multiply that error until it’s fixed or shut down. This is where human operators come in.

A fatal example of this was Air France Flight 447, where a failure of automation put the pilots into a manual situation they were not prepared for. [8]

2.4 Advantages and Disadvantages

The main advantages of automation are:

- Increased throughput or productivity.
- Improved quality or increased predictability of quality.
- Improved robustness (consistency), of processes or product.
- Increased consistency of output.

- Reduced direct human labor costs and expenses.

The following methods are often employed to improve productivity, quality, or robustness.

- Install automation in operations to reduce cycle time.
- Install automation where a high degree of accuracy is required.
- Replacing human operators in tasks that involve hard physical or monotonous work.
- Replacing humans in tasks done in dangerous environments (i.e. fire, space, volcanoes, nuclear facilities, underwater, etc.)
- Performing tasks that are beyond human capabilities of size, weight, speed, endurance, etc.
- Reduces operation time and work handling time significantly.
- Frees up workers to take on other roles.
- Provides higher level jobs in the development, deployment, maintenance and running of the automated processes.

The main disadvantages of automation are:

- Security Threats/Vulnerability: An automated system may have a limited level of intelligence, and is therefore more susceptible to committing errors outside of its immediate scope of knowledge (e.g., it is typically unable to apply the rules of simple logic to general propositions).
- Unpredictable/excessive development costs: The research and development cost of automating a process may exceed the cost saved by the automation itself.
- High initial cost: The automation of a new product or plant typically requires a very large initial investment in comparison with the unit cost of the product, although the cost of automation may be spread among many products and over time.

In manufacturing, the purpose of automation has shifted to issues broader than productivity, cost, and time. [6]

2.5 Limitations to Automation

- Current technology is unable to automate all the desired tasks.
- Many operations using automation have large amounts of invested capital and produce high volumes of product, making malfunctions extremely costly and potentially hazardous. Therefore, some personnel are needed to ensure that the entire system functions properly and that safety and product quality are maintained.
- As a process becomes increasingly automated, there is less and less labor to be saved or quality improvement to be gained. This is an example of both diminishing returns and the logistic function.
- As more and more processes become automated, there are fewer remaining non-automated processes. This is an example of exhaustion of opportunities. New technological paradigms may however set new limits that surpass the previous limits. [6]

2.5.1 Current Limitations

Many roles for humans in industrial processes presently lie beyond the scope of automation. Human-level pattern recognition, language comprehension, and language production ability are well beyond the capabilities of modern mechanical and computer systems (but see Watson (computer)). Tasks requiring subjective assessment or synthesis of complex sensory data, such as scents and sounds, as well as high-level tasks such as strategic planning, currently require human expertise. In many cases, the use of humans is more cost-effective than mechanical approaches even where automation of industrial tasks is possible. Overcoming these obstacles is a theorized path to post-scarcity economics.[6]

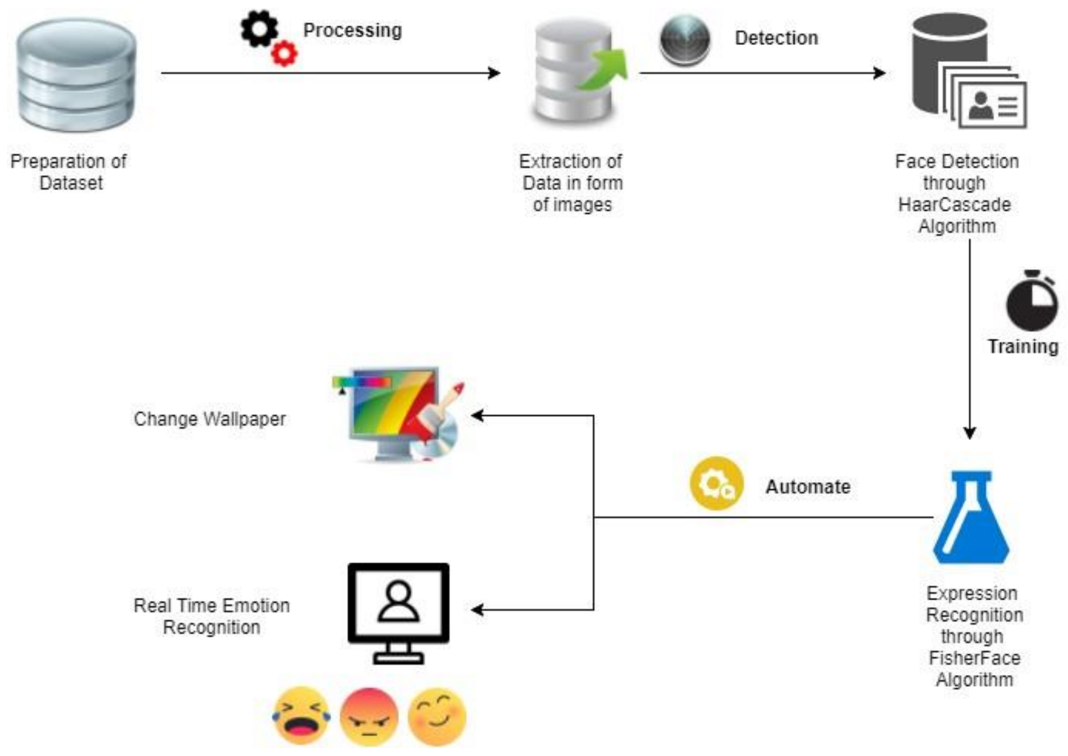

Processing
Detection
Preparation of Dataset
Extraction of Data in form of images
Face Detection through HaarCascade Algorithm
Training
Change Wallpaper
Automate
Real Time Emotion Recognition
Expression Recognition through FisherFace Algorithm

Figure 2.14 Data Flow diagram of the current project

3.
TECHNOLOGIES

3.1 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.
The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. The library is used extensively in companies, research groups and by governmental bodies.

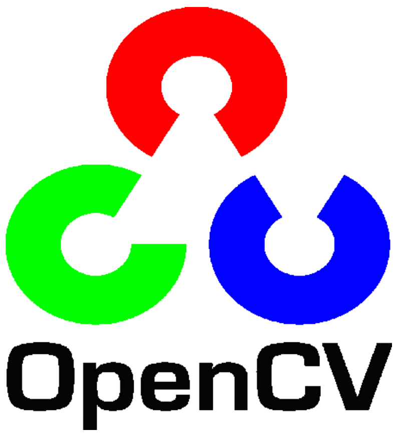


Figure 3.1 OpenCV Logo

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.
It has C++, C, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of

MMX and SSE instructions when available. A full-featured CUDAand OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written nativelyin C++ and has a templated interface that works seamlessly with STL containers. [9]

3.2 Image Processing

Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image.

There are five stages in any digital image processing application. They are broadly classified as:

- Image Acquisition
- Image Pre-processing
- Image Segmentation
- Features Extraction
- Classification and Prediction

A. Image Acquisition The image is captured by a sensor (e.g. Camera), and digitized if the output of the camera or sensor is not already in digital form, using analogue-to-digital convertor.

B. Image Pre-processing Pre-processing methods use a small neighborhood of a pixel in an input image to get a new brightness value in the output image. Such pre-processing operations are also called filtration. Image preprocessing tool uses many useful pre-processing operations which suppress information that is no relevant to the specific image processing and enhance some image features important for further processing. Some of the pre-processing includes image enhancement, cropping, de-noising, etc.

C. Image Segmentation Segmentation is the process of partitioning an image into non-intersecting regions such that each region is homogeneous and the union of no two adjacent regions is homogeneous. The goal of segmentation is typically to locate certain objects of interest which may be depicted in the image. Segmentation could therefore be seen as a computer vision problem. There are four popular segmentation approaches: threshold methods, edge-based methods, region based methods and the connectivity-preserving relaxation methods.

D. Features Extraction Feature extraction is a special form of the dimensionality reduction. Feature extraction involves simplifying the amount of resources required to describe a large set of data accurately. Feature extraction methods can be supervised or unsupervised, depending on whether or not class labels are used. Among the unsupervised methods, Principal Component Analysis (PCA), Independent Component Analysis (ICA), Multi-dimensional scaling (MDS) are the most popular ones. Supervised FE methods (and also FS methods) either use information about the current classification performance called wrappers, or use some other, indirect measure, called filters.

E. **Classification and Prediction** This final step involves classification of segmented image under various labels based on the features generated. This classification is done using the various data mining techniques. Classification consists of assigning a class label to a set of unclassified cases. There are two different classes of classification. They are supervised classification where the set of possible classes is known in advance and unsupervised classification where the set of possible classes is not known and after classification we can try to assign a name to the class. Unsupervised classification is also known as clustering. [10]

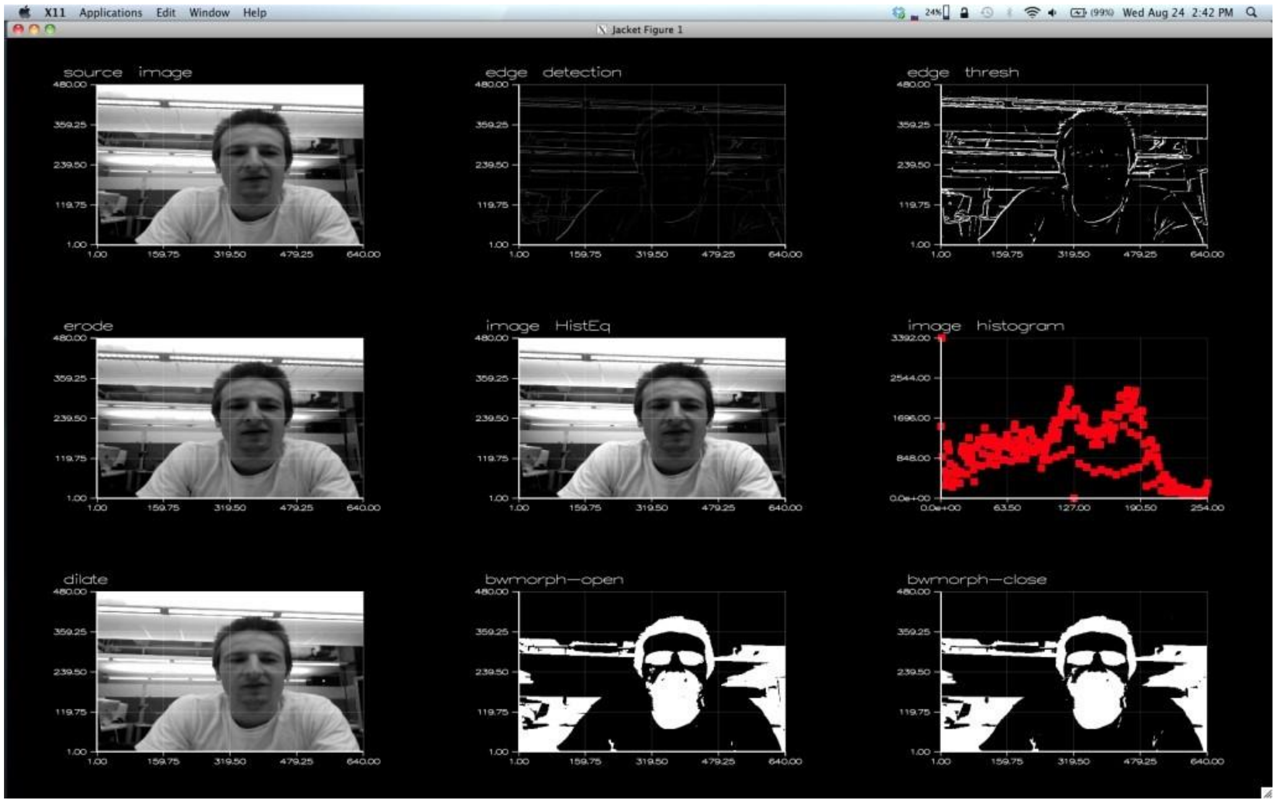


Figure 3.2 Graphs visualizing Image Processing

3.3 Histogram Equalization

3.3.1 Theory

Consider an image whose pixel values are confined to some specific range of values only. For eg, brighter image will have all pixels confined to high values. But a good image will have pixels from all regions of the image. So you need to stretch this histogram to either ends (as given in below image, from wikipedia) and that is what Histogram Equalization does (in simple words). This normally improves the contrast of the image. [11]

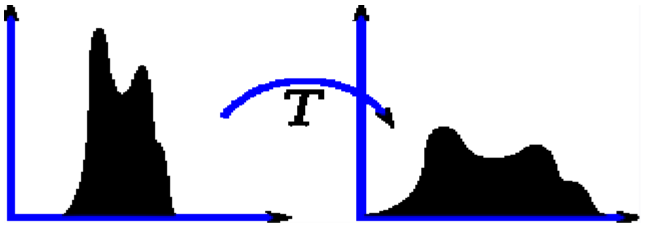


Figure 3.3 Wikipedia Image stating Histogram Equalization

3.3.2 Histograms Equalization in OpenCV

OpenCV has a function to do this, cv2.equalizeHist(). Its input is just grayscale image and output is our histogram equalized image.
Below is a simple code snippet showing its usage for same image we used :

```
img = cv2.imread('wiki.jpg',0)
equ = cv2.equalizeHist(img)
res = np.hstack((img,equ)) #stacking images side-by-side
cv2.imwrite('res.png',res)
```

Figure 3.4 Code Snippet for Histogram Equalization in OpenCV

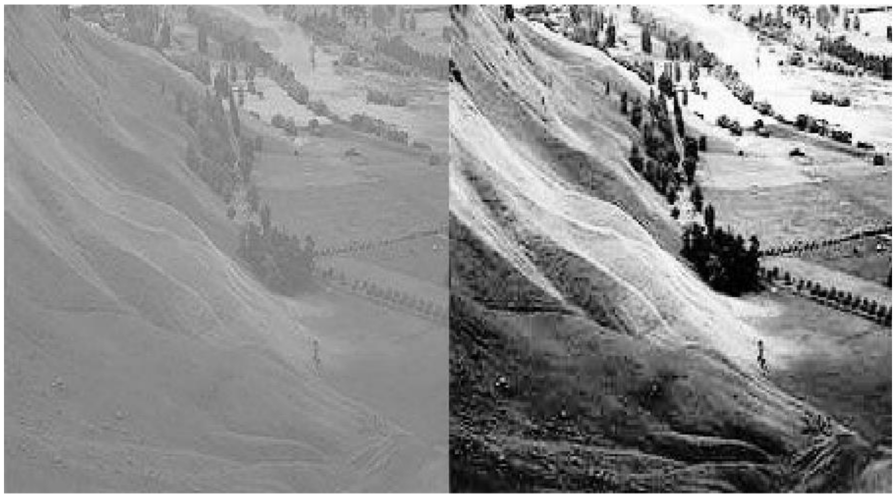


Figure 3.5 Histograms Equalization in OpenCV

So now you can take different images with different light conditions, equalize it and check the results.
Histogram equalization is good when histogram of the image is confined to a particular region. It won't work good in places where there is large intensity variations where histogram covers a large region, ie both bright and dark pixels are present. Please check the SOF links in Additional Resources. [11]

3.3.3 CLAHE (Contrast Limited Adaptive Histogram Equalization)

The first histogram equalization we just saw, considers the global contrast of the image. In many cases, it is not a good idea. For example, below image shows an input image and its result after global histogram equalization. It is true that the background contrast has improved after histogram equalization. But compare the face of statue in both images. We lost most of the information there due to over-brightness. It is because its histogram is not confined to a particular region as we saw in previous cases (Try to plot histogram of input image, you will get more intuition).

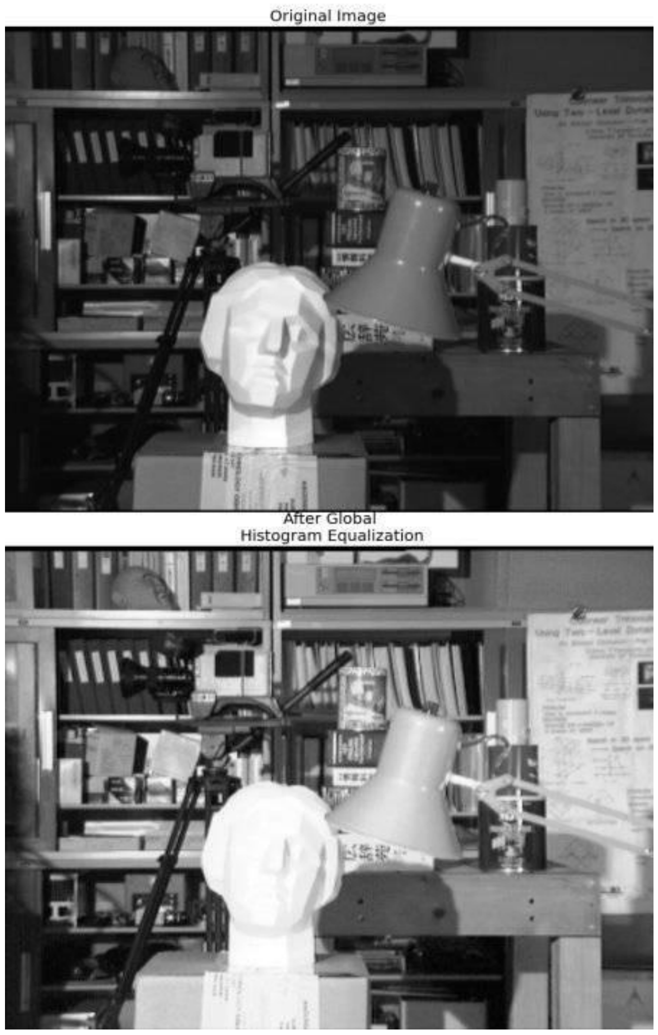


Figure 3.6 Before and After Histogram Equalization

So to solve this problem, adaptive histogram equalization is used. In this, image is divided into small blocks called "tiles" (tileSize is 8x8 by default in OpenCV). Then each of these blocks are histogram equalized as usual. So in a small area, histogram would confine to a small region (unless there is noise). If noise is there, it will be amplified. To avoid this, contrast limiting is applied. If any histogram bin is above the specified contrast limit (by default 40 in OpenCV), those pixels are clipped and distributed uniformly to other bins before applying histogram equalization. After equalization, to remove artifacts in tile borders, bilinear interpolation is applied.
Below code snippet shows how to apply CLAHE in OpenCV:

```
import numpy as np
import cv2
img = cv2.imread('tsukuba_l.png',0)
# create a CLAHE object (Arguments are optional).
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
cl1 = clahe.apply(img)
```

```
cv2.imwrite('clahe_2.jpg',cl1)
```

Figure 3.7 Code Snippet for application of CLAHE in OpenCV

See the result below and compare it with results above, especially the statue region:

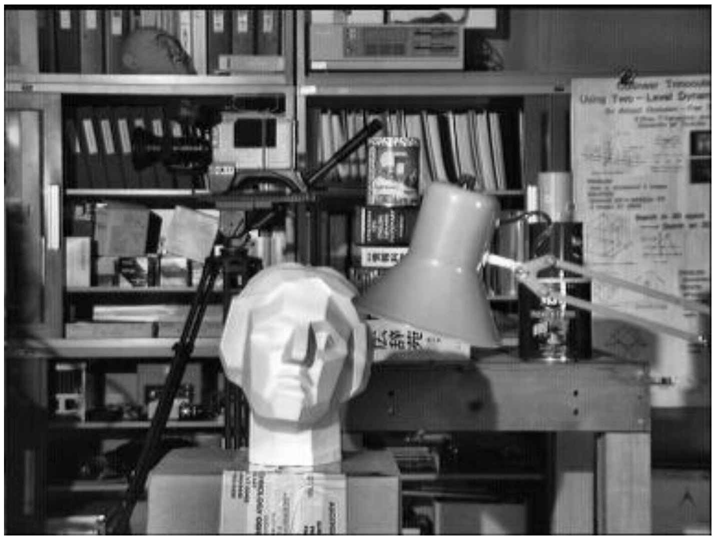


Figure 3.8 Sample Image

3.4 Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter,wxPython, Qt, or GTK+.
There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged.SciPy makes use of matplotlib.
matplotlib was originally written by John D. Hunter, has an active development community, and is distributed under aBSD-style license.
Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in 2012. [12]

3.4.1 Line Plot

```
import matplotlib.pyplot as plt
import numpy as np
a = np.linspace(0,10,100)
b = np.exp(-a)
plt.plot(a,b)
plt.show()
```

Figure 3.9 Code Snippet for Line Plot

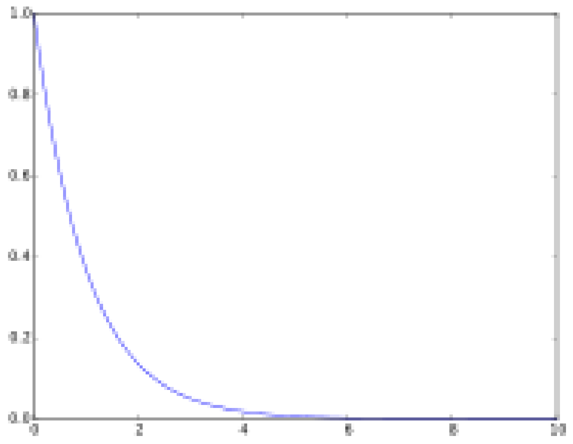


Figure 3.10 Line Plot after application of Code Snippet

3.4.2 Histogram

```
import matplotlib.pyplot as plt
from numpy.random import normal,rand
x = normal(size=200)
plt.hist(x,bins=30)
plt.show()
```

Figure 3.11 Code Snippet for a Histogram

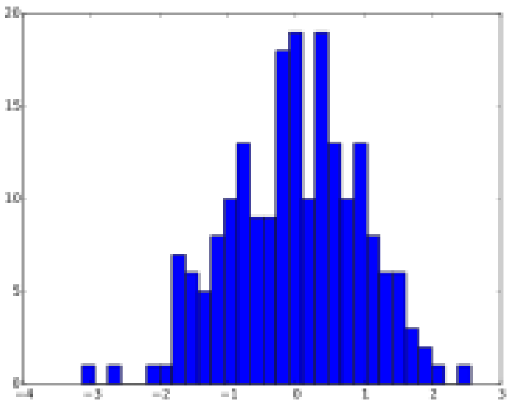


Figure 3.12 Histogram formed after application of Code Snippet

3.4.3 Scatter Plot

```
import matplotlib.pyplot as plt
from numpy.random import rand
a = rand(100)
b = rand(100)
plt.scatter(a,b)
plt.show()
```

Figure 3.13 Code Snippet for a Scatter Plot

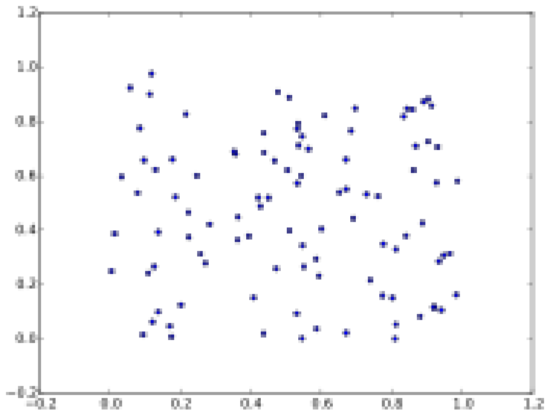


Figure 3.14 Scatter Plot formed after application of Code Snippet

3.4.4 3D Plot

```
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np
fig = plt.figure()
ax = fig.gca(projection='3d')
X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.coolwarm)
plt.show()
```

Figure 3.15 Code Snippet for a 3D Plot

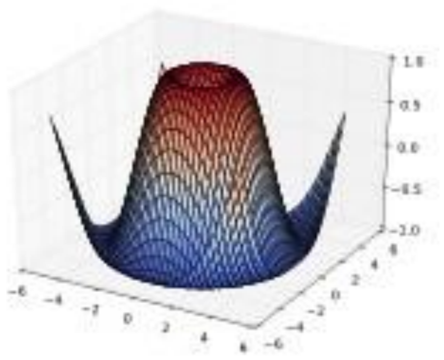


Figure 3.16 3D Plot formed after application of Code Snippet

3.5 Numpy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. [13]

3.5.1 The ndarray Data Structure

The core functionality of NumPy is its "ndarray", for *n*-dimensional array, data structure. These arrays are strided views on memory.In contrast to Python's built-in list data structure (which, despite the name, is a dynamic array), these arrays are homogeneously typed: all elements of a single array must be of the same type.
Such arrays can also be views into memory buffers allocated by C/C++, Cython, and Fortran extensions to the CPython interpreter without the need to copy data around, giving a degree of compatibility with existing numerical libraries. This functionality is exploited by the SciPy package, which wraps a number of such libraries (notably BLAS and LAPACK). NumPy has built-in support for memory-mapped ndarrays. [13]

3.5.1.1 Limitations

Inserting or appending entries to an array is not as trivially possible as it is with Python's lists. The np.pad(...) routine to extend arrays actually creates new arrays of the desired shape and padding values, copies the given array into the new one and returns it. NumPy's np.concatenate([a1,a2]) operation does not actually link the two arrays but returns a new one, filled with the entries from both given arrays in sequence. Reshaping the dimensionality of an array with np.reshape(...) is only possible as long as the number of elements in the array does not change. These circumstances originate from the fact that NumPy's arrays must be views on contiguous memory buffers. A replacement package called Blaze attempts to overcome this limitation.
Algorithms that are not expressible as a vectorized operation will typically run slowly because they must be implemented in "pure Python", while vectorization may increase memory complexity of some operations from constant to linear, because temporary arrays must be created that are as large as the inputs. Runtime compilation of numerical code has been implemented by several groups to avoid these problems; open source solutions that interoperate with NumPy include scipy.weave, numexpr and Numba. Cython is a static-compiling alternative to these. [13]

3.5.2 Examples

3.5.2.1 Array creation

```
import numpy as np
x = np.array([1, 2, 3])
y = np.arange(10)  # like Python's range, but returns an array
```

Figure 3.17 Code Snippet for Array Creation

3.5.2.2 Basic operations

```
a = np.array([1, 2, 3, 6])
b = np.linspace(0, 2, 4)  # create an array with four equally spaced points starting with 0 and ending with 2
c = a - b
a**2
```

Figure 3.18 Code Snippet for Basic Operations in Python

3.5.2.3 Universal functions

```
a = np.linspace(-np.pi, np.pi, 100)
b = np.sin(a)
c = np.cos(a)
```

Figure 3.19 Code Snippet for Universal Functions

3.6 OS Module in Python

The OS module in python provides functions for interacting with the operating system. OS, comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality. The *os* and *os.path* modules include many functions to interact with the file system. [14]

Following are some functions in OS module:

1. Executing a shell command	**os.system()**
2. Get the users environment	**os.environ()**
3. Returns the current working directory.	**os.getcwd()**
4. Return the real group id of the current process.	**os.getgid()**
5. Return the current process's user id.	**os.getuid()**

6. Returns the real process ID of the current process.	**os.getpid()**
7. Set the current numeric umask and return the previous umask.	**os.umask(mask)**
8. Return information identifying the current operating system.	**os.uname()**
9. Change the root directory of the current process to path.	**os.chroot(path)**
10. Return a list of the entries in the directory given by path.	**os.listdir(path)**
11. Create a directory named path with numeric mode mode.	**os.mkdir(path)**
12. Recursive directory creation function.	**os.makedirs(path)**
13. Remove (delete) the file path.	**os.remove(path)**
14. Remove directories recursively.	**os.removedirs(path)**
15. Rename the file or directory src to dst.	**os.rename(src, dst)**
16. Remove (delete) the directory path.	**os.rmdir(path)**

3.7 Python2.7

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale.
Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.
Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems. [15]

4.
Implementation

4.1 Project Description

This project aims at recognizing emotions of a person and using that information, to further automate tasks such as changing desktop wallpaper or swapping the face with an appropriate emoji. We have restricted ourselves to three emotions – Angry, Happy and Surprise.
Each emotion represents a class which is given a supervising signal. This supervising signal is the label for each of the three classes. Label of the classes Angry, Happy and Surprise are 0, 1 and 3 respectively.
Images of each class and their corresponding labels are passed in the FisherFace Classifier.
The Classifier then gets trained on the three expressions and we get our trained model, "final_trained_model".

4.1.1 Code Snippet for Training using FisherFace

```
fisherface = cv2.createFisherFaceRecognizer()
fisherface.train(images, labels)
fisherface.save("final_trained_model.xml")
```

Figure 4.1 Code Snippet for training using FisherFace

4.2 Prediction and Automation

After the classifier gets trained on the expressions, we get our trained model "final_trained_model.xml". We then make use of this trained model for the prediction of expressions.
So the image whose expression is to be predicted is passed on to the trained model and the model then returns the label which represents the class or the expression of the person. Based on these returned labels, tasks get automated.
Tasks performed are –
1. Face Swap:
This involves swapping of the face of the person with the corresponding emoji which represents the person's facial expressions.
2. Wallpaper Changer:
This involves automatically changing the wallpaper of the desktop according to the predicted expressions of the person.

4.2.1 Code Snippet for Prediction and Automation

```
while 1:
     cv2.namedWindow("RawWindow")
    cv2.namedWindow("FaceSwappedWindow")
    ret,img=cap.read()
```

```

            cv2.imshow("RawWindow",img_gray)
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
            roi_gray=img_gray[y:y+h,x:x+w]

            roi_gray=cv2.resize(roi_gray,(268,268))


        key=cv2.waitKey(30) & 0xff
        if key==27:    #esc key

            break
        if key==ord("s"):
            if pred==0:
                print "Angry"
                emoji=cv2.imread("Emoji/anger.png")
                emoji_gray=np.array(cv2.cvtColor(emoji,cv2.COLOR_BGR2GRAY))
                emoji_gray=cv2.resize(emoji_gray,(h,w))
                img_gray[y:y+h,x:x+w]=emoji_gray
                cv2.imshow("FaceSwappedWindow",img_gray)
                command = 'gsettings set org.gnome.desktop.background pictu  re-uri "path"'
                status, output = commands.getstatusoutput(command)
            if pred==1:
                print "Happy"
                emoji=cv2.imread("Emoji/happy.png")
                emoji_gray=np.array(cv2.cvtColor(emoji,cv2.COLOR_BGR2GRAY))
                emoji_gray=cv2.resize(emoji_gray,(h,w))
                img_gray[y:y+h,x:x+w]=emoji_gray
                cv2.imshow("FaceSwappedWindow",img_gray)
                command = 'gsettings set org.gnome.desktop.background picture-uri "path"'
                status, output = commands.getstatusoutput(command)

            if pred==3:
                print "Surprise"
                emoji=cv2.imread("Emoji/surprise.png")
                emoji_gray=np.array(cv2.cvtColor(emoji,cv2.COLOR_BGR2GRAY))
                emoji_gray=cv2.resize(emoji_gray,(h,w))
                img_gray[y:y+h,x:x+w]=emoji_gray
                cv2.imshow("FaceSwappedWindow",img_gray)
                command = 'gsettings set org.gnome.desktop.background picture-uri "path"'
                status, output = commands.getstatusoutput(command)

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

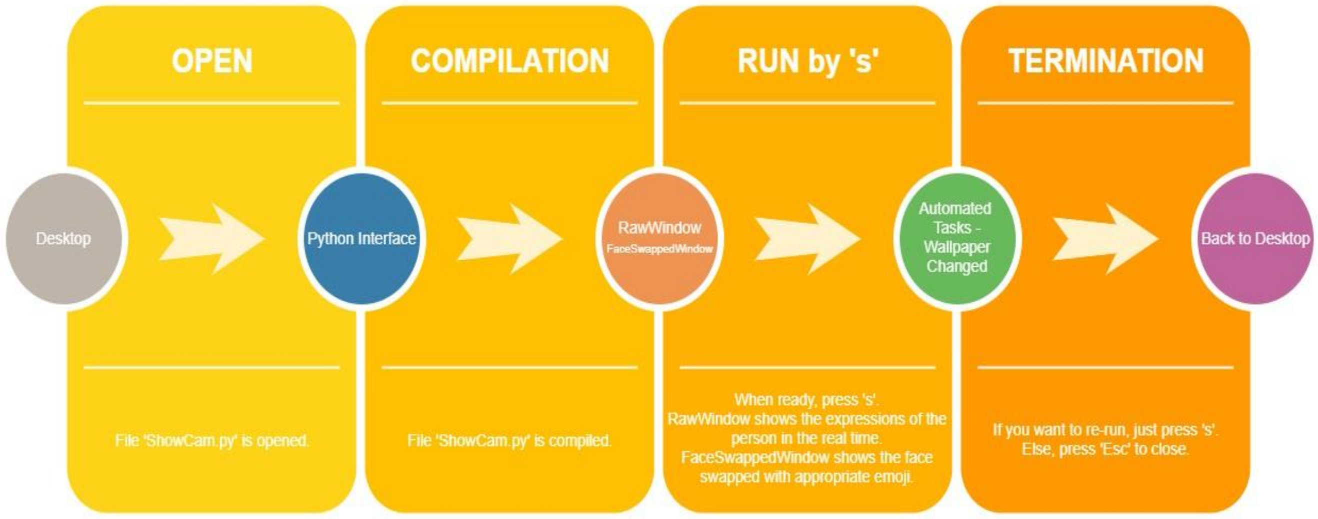
Figure 4.2 Code Snippet for Prediction and Automation

4.3 Project Step – Wise User Case

1. First compile the file ShowCam.py
   This file contains the trained fisherface model which is "final_trained_model.xml"
   The model returns the label which corresponds to the expressions of the person.
   Once the file ShowCam.py is compiled then we get 2 windows namely RawWindow and FaceSwappedWidow.
2. The RawWindow shows the expressions of the person in the real time.

Once the person presses the key "s" then his/her expressions are captured at that instant and the face of the person gets swapped in the FaceSwappedWindow with the Emoji that reprsents the person's emotions or expressions.

3. Simultaneously the wallpaper of the system changes according to the predicted expressions of the person.
4. The user can repeat this process any number of times by pressing the key "s". The process gets terminated once the user presses "Esc" key.



OPEN
COMPILATION
RUN by 's'
TERMINATION
Desktop
Python Interface
RawWindow
FaceSwappedWindow
Automated Tasks - Wallpaper Changed
Back to Desktop
File 'ShowCam.py' is opened.
File 'ShowCam.py' is compiled.
When ready, press 's'. RawWindow shows the expressions of the person in the real time. FaceSwappedWindow shows the face swapped with appropriate emoji.
If you want to re-run, just press 's'. Else, press 'Esc' to close.

Figure 4.3 Project Step-Wise overview

5.
Testing

5.1 Introduction

Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects), and verifying that the software product is fit for use. Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test which meets the requirements that guided its design and development, responds correctly to all kinds of inputs, performs its functions within an acceptable time, is sufficiently usable, can be installed and run in its intended environments, and achieves the general result its stakeholders desire. [16]

5.2 Screenshots

5.2.1 For Class Happy – Specimen 1

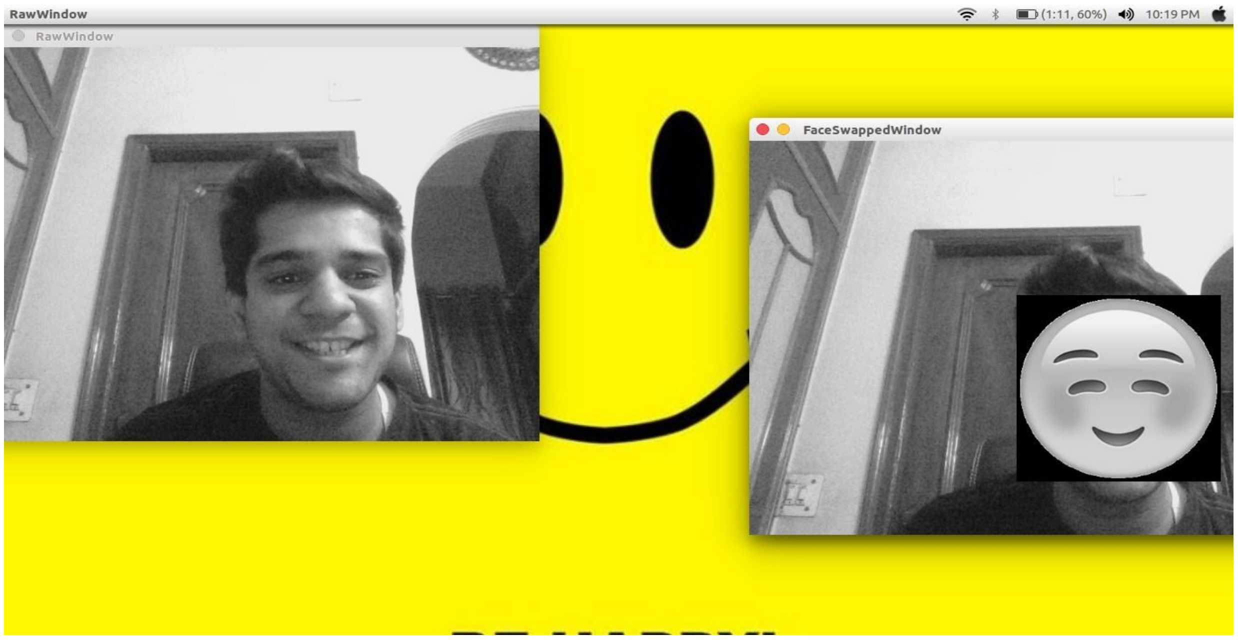


Figure 5.1 Screenshot for class Happy - Specimen 1

5.2.2 For Class Angry – Specimen 1

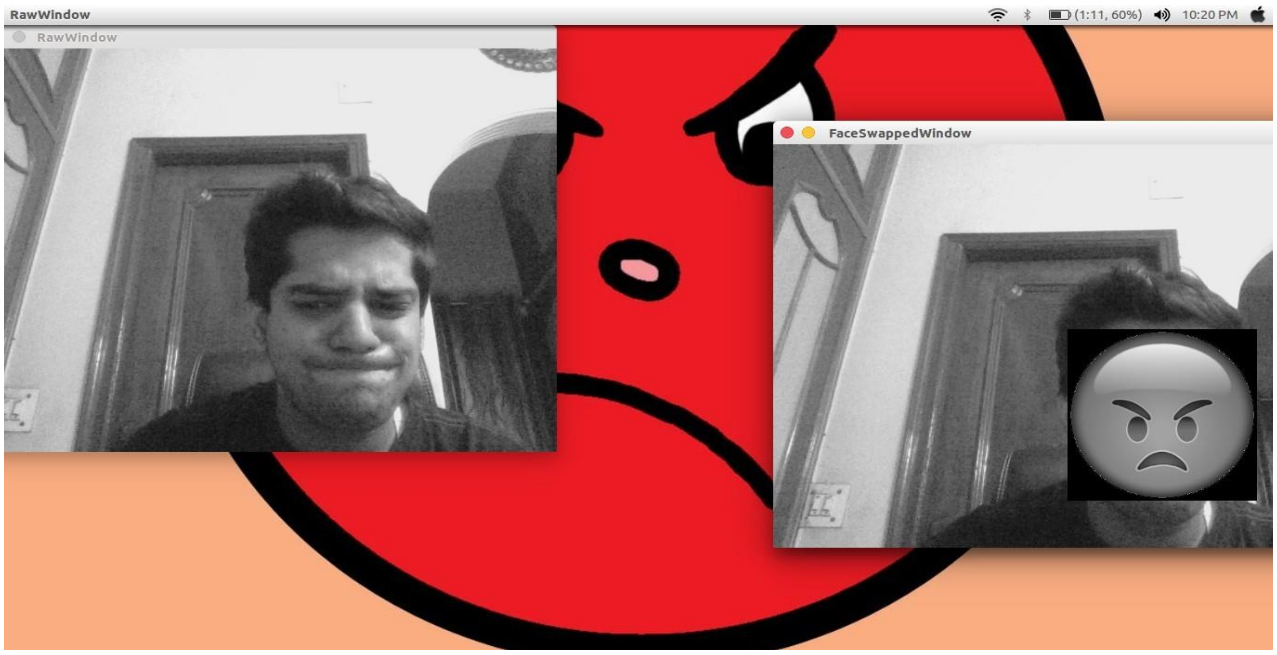


Figure 5.2 Screenshot for class Angry - Specimen 1

5.2.3 For Class Surprise – Specimen 1

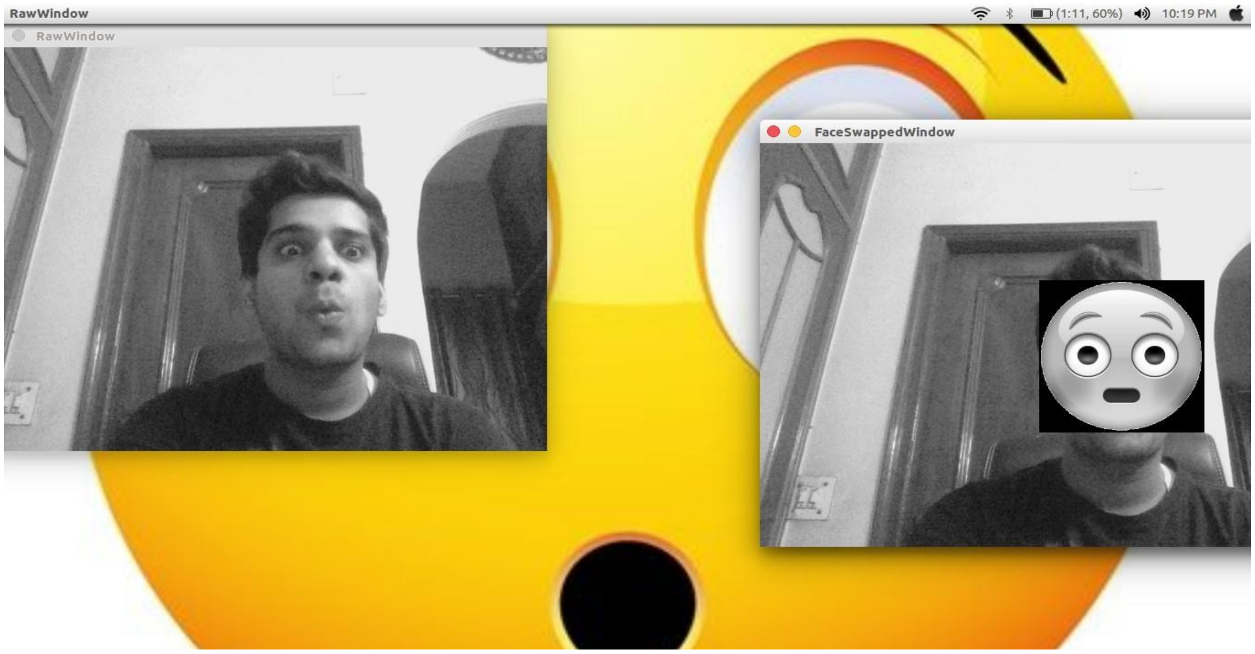


Figure 5.3 Screenshot for class Surprise - Specimen 1

5.2.4 For Class Surprise – Specimen 2

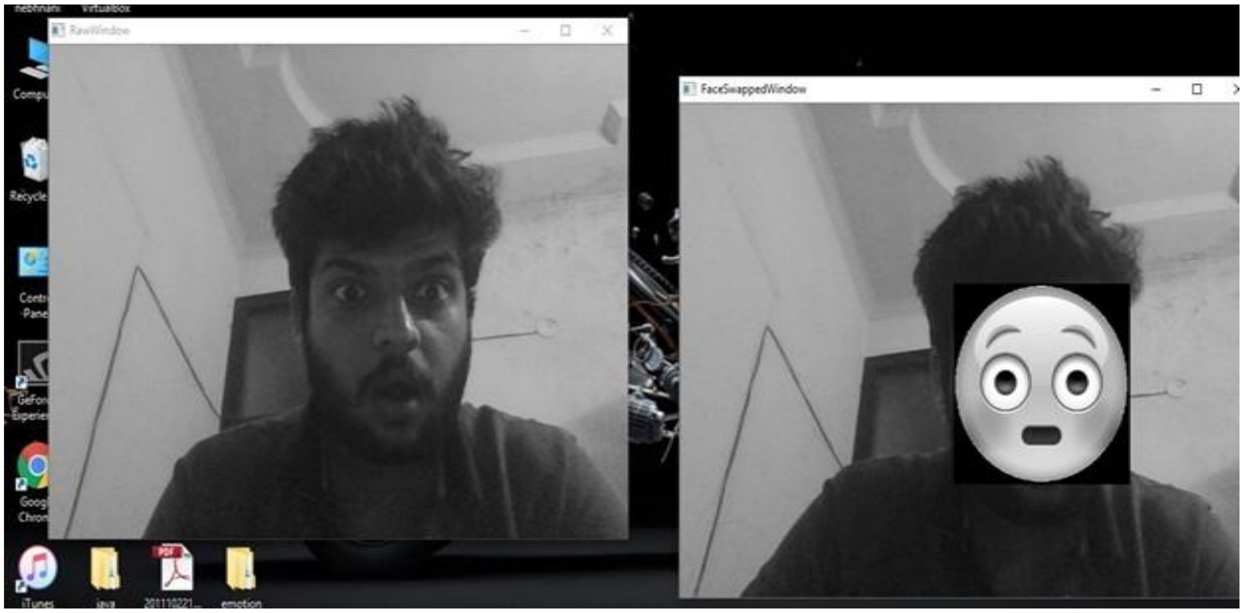


Figure 5.4 Screenshot for class Surprise - Specimen 2

5.2.5 For Class Happy – Specimen 2

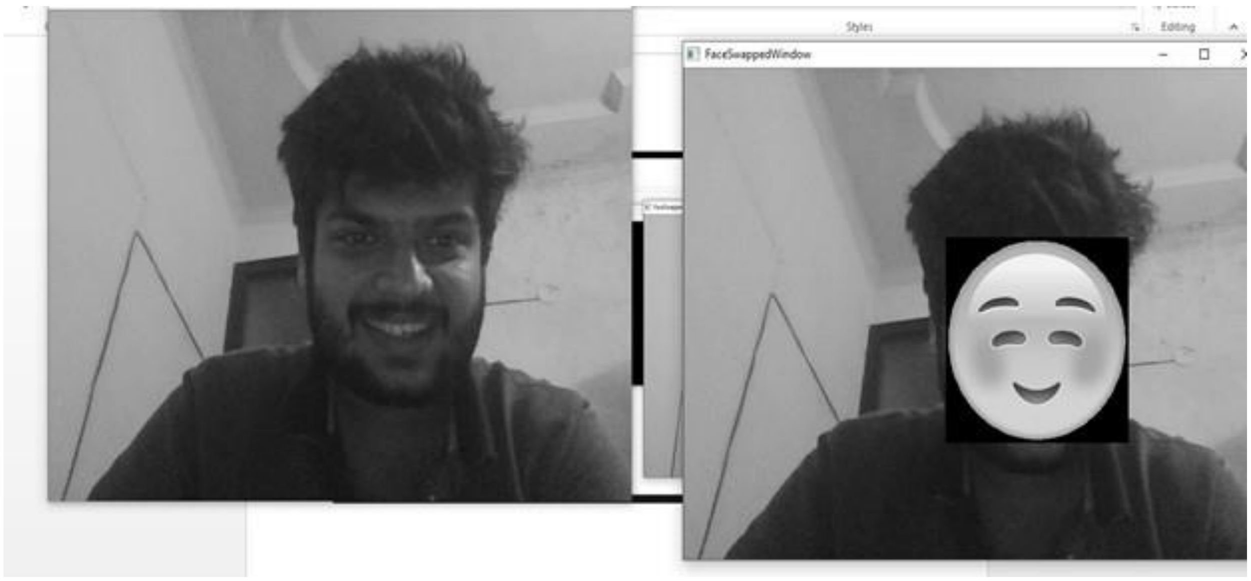


Figure 5.5 Screenshot for class Happy - Specimen 2

5.2.6 For Class Angry – Specimen 2

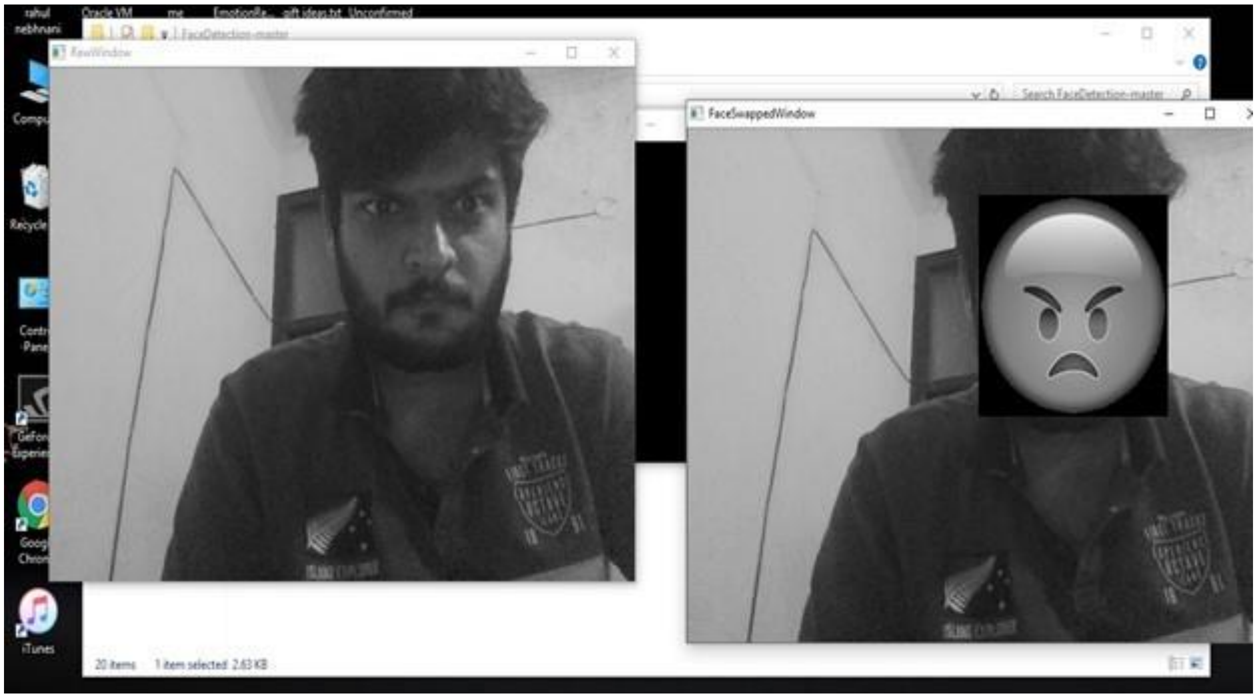


Figure 5.6 Screenshot for class Angry - Specimen 2

6.
Conclusion and Future Scope

6.1 Conclusion

The human face plays a prodigious role for automatic recognition of emotion in the field of identification of human emotion and the interaction between human and computer for some real application like driver state surveillance, personalized learning, health monitoring etc.
For human-computer interaction facial expression makes a platform for non-verbal communication. Facial expression recognition system requires to overcome the human face having multiple variability such as color, orientation, expression, posture and texture so on.
Through this project we were able to detect the emotions showed by multiple individuals and were able to predict them in real time application windows.
Automated execution of tasks such as face swapping and wallpaper changing were also implemented which depended on the emotion shown by a person.
Our system has been able to predict the expressions with the descent accuracy.
However by making use of larger datasets and availability of better computational resources would further enhance the accuracy of the system. Moreover , implementing complex neural networks like Convolutional neural networks which consist of multiple intricate layers would significantly increase the accuracy of the system.

6.2 Future Scope

With the successful completion of the detection of various emotions of a person, the future goals involve increasing the number of classes for expressions.
Currently we have just three classes namely happy,angry,surprise which represent the emotions of the person. In future we shall increase number of classes such as sad,disgust,fear,contempt.
Advanced model such as Convolutional Neural Networks can be used with complex layers
Which would significantly increase the accuracy of the prediction.
With the availability of better resources we can provide Colored Images as well as Colored Real Time Monitoring.
Furthermore, our system can be extended in the field of security systems, which can identify a person in any form of the expressions he presents himself.
Room in homes can set the lights, television to a person’s taste when they enter the room.

References

[1] SAS, "https://www.sas.com/en_us/insights/analytics/machine-learning.html," [Online].

[2] N. Cooper, "Facial Expression Recognition," 2013.

[3] P. I. Wilson, "FACIAL FEATURE DETECTION USING HAAR CLASSIFIERS," 2006.

[4] A. Martinez, "Fisherfaces".

[5] P. Wagner, 03 june 2012. [Online]. Available: https://www.bytefish.de/blog/fisherfaces/#introduction.

[6] J. M. Pearce, "Introduction to Open Hardware for Science," 2014.

[7] J. Boyd, "Robotic Laboratory Automation," 2002.

[8] J. Kaufman, "Paradox of Automation".

[9] IEEE, "ieeexplore.ieee.org/document/6240859/," [Online].

[10] A. R, "FACIAL EMOTIONS RECOGNITION SYSTEM".

[11] [Online]. Available: https://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html.

[12] " matplotlib.org.," [Online].

[13] D. Pine, "Python Resources," 2014.

[14] "pythonforbeginners.com," [Online]. Available: http://www.pythonforbeginners.com/os/pythons-os-module.

[15] D. M. Beazley, " Python Essential Reference".

[16] C. Kaner, "Exploratory Testing," in *Quality Assurance Institute Worldwide Annual Software Testing Conference*, Florida, November 17, 2006.