1. **Methodology**

There are various face recognition algorithms available like

* DeepFace
* DeepID series of systems
* FaceNet
* VVGFace

Face recognizers that are based on the landmarks take face images and try to find essential feature points such as eyebrows, corner of the mouth, eyes, nose, lips, etc.

There are more than 60 points.



**6. SOFTWARE AND HARDWARE REQUIREMENTS**

**Software required:**

* Language – Python
* Libraries – OpenCV Library, Numpy Library
* Utilities - Haarcascade\_eye, Haarcascade\_frontalface\_default, haarcascade\_smile
* An IDLE - Spyder

**Hardware required:**

* **Any webcam**

**7.CODE**

**import** **numpy** **as** **np**

**import** **cv2**

cap = cv2.VideoCapture(**0**)

cap.set(**3**,**640**) # set Width

cap.set(**4**,**480**) # set Height

**while**(True):

ret, frame = cap.read()

#frame = cv2.flip(frame, -1) # Flip camera vertically

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

cv2.imshow('frame', frame)

cv2.imshow('gray', gray)

k = cv2.waitKey(**30**) & **0xff**

**if** k == **27**: # press 'ESC' to quit

**break**

cap.release()

cv2.destroyAllWindows()

**import** **numpy** **as** **np**

#numpy to convert python lists to numpy arrays as it is needed by OpenCV face recognizers

**import** **cv2**

faceCascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

eye\_cascade = cv2.CascadeClassifier('haarcascade\_eye.xml')

cap =cv2.VideoCapture(**0**)

**while** True:

ret, img = cap.read()

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

faces = faceCascade.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] #this is the region of gray so y comes first then x

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)

k = cv2.waitKey(**30**) & **0xff**

**if** k == **27**: # press 'ESC' to quit

**break**

cap.release()

cv2.destroyAllWindows()

**import** **numpy** **as** **np**

**import** **cv2**

# Haarcascade are old mechine learning libraries

faceCascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

smileCascade = cv2.CascadeClassifier('haarcascade\_smile.xml')

#cap is to read the image using the webcam

#using 0 to access the default cam...we can use 1,2,.. for other webcam

cap = cv2.VideoCapture(**0**)

**while** True:

ret, img = cap.read() #ret is a flag to check the image is stored prooperly

#converting(cvtColor) the gray haarcascade to color as they were predesigned gray

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

#detect the image that it is looking for

faces = faceCascade.detectMultiScale(gray,**1.3**,**5**,minSize=(**30**, **30**))

#scaleFactor=1.3

#minNeighbors=5

#minSize=(30, 30)

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

#we are using cv2.rectangle to detect the x and y corrdn and

#heigh and width then according to that info the bonding box will be created

#here we are letting openCV know (x,y) is starting position and (x+w,y+h) is ending position

#(255,0,0) is the color of the bonding box

# 2 is the thickness of the box

cv2.rectangle(img,(x,y),(x+w,y+h),(**255**,**0**,**0**),**2**)

#Region of interest(roi)

#we are letting opencv know i m only interested in the gray region of the face

#using [y:y+h, x:x+w]

#it is only trained to detect face image not the main gray image

roi\_gray = gray[y:y+h, x:x+w]

smile = smileCascade.detectMultiScale(

roi\_gray,

scaleFactor= **1.5**,

minNeighbors=**15**,

minSize=(**25**, **25**),

)

**for** i **in** smile: #trying to find if one smile is detected or not

**if** len(smile)>**1**: #if there is atleast one smile the next statement will print

#(x,y-30) is the corrdn for the txt smiling on the detection box same as for the image of the main image

#cv2.FONT\_HERSHEY\_SIMPLEX is the font for the txt

#2 is the size of the font

#(0,255,0) the colors used on the box

#3 is the width of the detection box

#cv2.LINE\_AA the type of line the font should be using to inorder to render the word smiling

cv2.putText(img,"Smiling",(x,y-**30**),cv2.FONT\_HERSHEY\_SIMPLEX,

**2**,(**0**,**255**,**0**),**3**,cv2.LINE\_AA)

cv2.imshow('video', img) #to show the image

k = cv2.waitKey(**30**) & **0xff** #without waitkey the imshow won't work

**if** k == **27**: # press 'ESC' to quit

**break**

cap.release()

cv2.destroyAllWindows()

