Final production

OpenCVClient.py

```
import cv2

import numpy as np

import socket

host, port = "127.0.0.1", 25001

data = "true"

lowerParameter = 2.50;

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)


#the database of faces

faceDetect = cv2.CascadeClassifier('haarcascade/haarcascade_frontalface_default.xml')

smileDetect = cv2.CascadeClassifier('haarcascade/haarcascade_smile.xml')

eyeDetect = cv2.CascadeClassifier('haarcascade/haarcascade_eye.xml')


#a variable for the webcam

cam = cv2.VideoCapture(0)


s.connect((host, port))


while True:

    count = 0

    countFace = 0

    #two variables that is reading the camera

    ret,img = cam.read()


    #converting the img variable to grayscale

    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

```
   #first argument is for the image, second is for the vector of the rectangles, and third argument is for how
big the blob needs to be
   faces = faceDetect.detectMultiScale(gray,1.1,5)


   for(x,y,w,h) in faces:
      #drawing a square with x and y coordinates and adding the color and stroke
      cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
      roi_grey = gray[y:y+h,x:x+w]
      roi_color = img[y:y+h,x:x+w]
      countFace +=1
      smiles = smileDetect.detectMultiScale(roi_grey,lowerParameter,20)
      #eyes = eyeDetect.detectMultiScale(roi_grey,1.5,9)


      for(sx,sy,sw,sh) in smiles:
         count +=1
         cv2.rectangle(roi_color, (sx, sy), ((sx + sw), (sy + sh)), (0, 255, 0), 2)
     # for(ex,ey,ew,eh) in eyes:
         #cv2.rectangle(roi_color,(ex,ey), ((ex + ew), (ey + eh)), (255, 0, 0), 2)
   #creating a window with a name and what should be displayed
   cv2.imshow("Face",img)
   print(lowerParameter)
   if  countFace > 0:
      data = "Face without smile"


      if count > 0:


         data = "Face with smile"


   else :
        data = "No face detected"


    s.sendall(data.encode("utf-8"))
    #Data = s.recv(1024).decode("utf-8")
    #print(countFace)
    if cv2.waitKey(1) == ord('w'):
        lowerParameter +=0.10
    if cv2.waitKey(1) == ord('s'):
        lowerParameter =lowerParameter-0.10
    #killing the windows with webcam feedback
    if cv2.waitKey(1) == ord('q'):
        break
cam.release()
cv2.destroyAllWindows()
```

RealtimeEnvironment