Implementation and Analysis of Face Tracking Camera

AIM :

To build a hardware based project using Arduino UNO and build software to track the face movements using an External Camera.

Submitted by:

Priyansh Bhatnagar 2K18/EP/058
Tarun Arora 2K18/EP/085

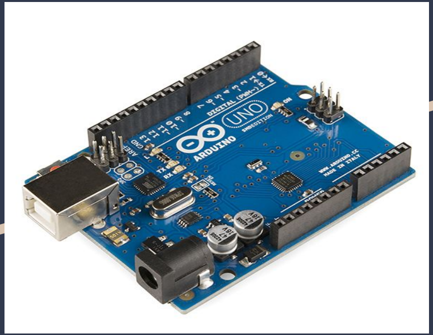
Introduction

Technologies Used

Programming Languages: Python, C

Hardware Devices: External USB Camera, Microcontroller, Servo Motors, USB cable

Software Platforms: Arduino IDE, Spyder IDE (Python IDE)



In this project, we have discussed an approach to implement concepts learned in order to build an arduino based project with an external USB camera that tracks the movement of the face in the frame.

To implement the project, we used Arduino UNO as the microcontroller which controls the movement of both servo-motors and to detect the human face in the camera's frame, we have used python programming language.

This project discusses the working project in detail along with the components used, schematic diagram, connections and programming of Arduino UNO and python program to detect the human face.

Problem Formulation

Objectives -

- To study the proposed and existing projects in the field through research papers.
- To implement Face Detection Haarcascade Classifier in Python.
- To build the working project using Arduino UNO to move the frame of the camera with real-time movement of the detect face.

Literature Review

For our project, we observed that face detection is the base for face tracking and face recognition, whose results directly affect the process and accuracy of face recognition.

In [1] and [2], we studied how authors implemented the Haarcascade Face Detection classification using MATLAB whereas in [3] the authors have tried to implement it using the Java programming language. We took the inspiration from these papers and have tried to implement the solution using Python and Arduino UNO.

Few Applications:

Face detection is used in biometrics, often as a part of (or together with) a facial recognition system.

It is also used in video surveillance, human computer interface and image database management.

Methodology

In this section, we will discuss about the concepts in action, methodologies used, project setup and give a brief but extensive review of our program.

We will first discuss briefly about Image Processing and how it has played a major role in our project.

Computer Vision and Face Detection

Computer vision is a process to give the ability to the computer to see as a human. Face detection part of the computer vision techniques. There are 3 methods for implementing face detection.

1.Knowledge-Based:-

The knowledge-based method depends on the set of rules, and it is based on human knowledge to detect the faces. Ex- A face must have a nose, eyes, and mouth within certain distances and positions with each other. The big problem with these methods is the difficulty in building an appropriate set of rules.

2.Feature-Based:-

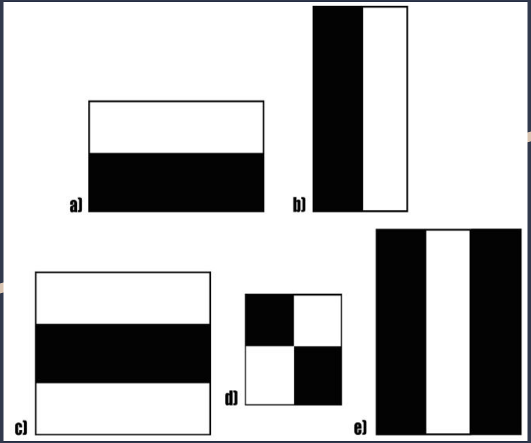
The feature-based method is to locate faces by extracting structural features of the face. It is first trained as a classifier and then used to differentiate between facial and non-facial regions.

3.Template Matching:-

Template Matching method uses pre-defined or parameterised face templates to locate or detect the faces by the correlation between the templates and input images.

HaarCascade Classifier

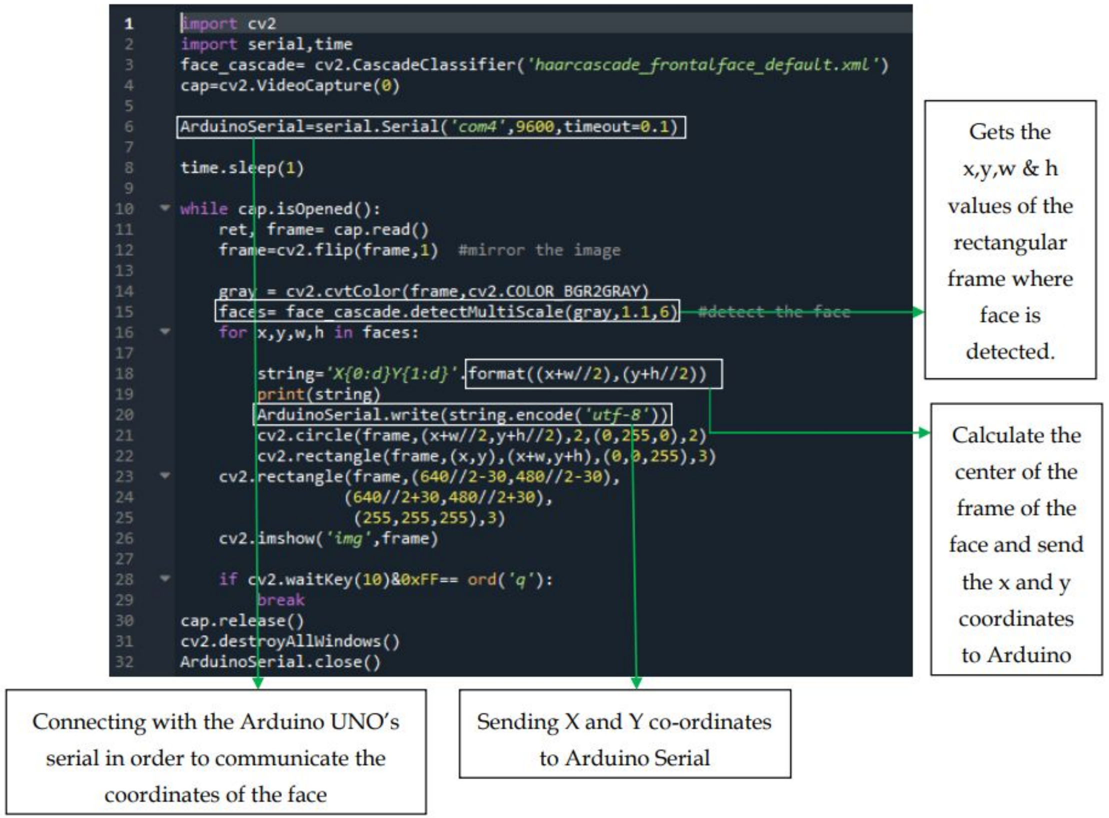
It's an Object Detection Algorithm that detects faces in images and real-time videos. The models are saved in XML files in the repository and can be accessed using OpenCV techniques. Face detection, eye detection, upper and lower body detection, licence plate detection, and other models are among them.



Haar features from the image helps to find the edges or the lines in the image, or to pick areas where there is a sudden change in the intensities of the pixels. The goal is to calculate the sum of all image pixels in the darker part of the haar feature, as well as the sum of all image pixels in the lighter portion of the haar feature. Then figure out how they differ.

Attentional Cascading is used to save time and complexity of the classifier. Features are applied on the images in stages. The stages in the beginning contain simpler features, in comparison to the features in a later stage which are complex, complex enough to find the nitty gritty details on the face.

The second stage processing would start, only when the features in the first stage are detected in the image. The process continues like this, i.e. if one stage passes, the window is passed onto the next stage, if it fails then the window is discarded.



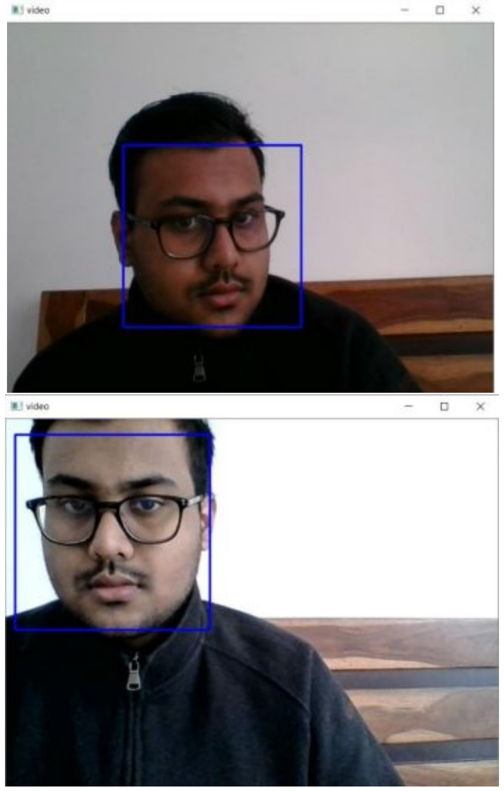
```python
import cv2
import serial,time
face_cascade= cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cap=cv2.VideoCapture(0)

ArduinoSerial=serial.Serial('com4',9600,timeout=0.1)

time.sleep(1)

while cap.isOpened():
    ret, frame= cap.read()
    frame=cv2.flip(frame,1)  #mirror the image

    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces= face_cascade.detectMultiScale(gray,1.1,6)  #detect the face
    for x,y,w,h in faces:

        string='X{0:d}Y{1:d}'.format((x+w//2),(y+h//2))
        print(string)
        ArduinoSerial.write(string.encode('utf-8'))
        cv2.circle(frame,(x+w//2,y+h//2),2,(0,255,0),2)
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),3)
    cv2.rectangle(frame,(640//2-30,480//2-30),
                 (640//2+30,480//2+30),
                  (255,255,255),3)
    cv2.imshow('img',frame)

    if cv2.waitKey(10)&0xFF== ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
ArduinoSerial.close()
```

Gets the x,y,w & h values of the rectangular frame where face is detected.

Calculate the center of the frame of the face and send the x and y coordinates to Arduino

Connecting with the Arduino UNO's serial in order to communicate the coordinates of the face

Sending X and Y co-ordinates to Arduino Serial



This operation was done with different face angles, lighting conditions and different positions to test the efficiency and accuracy of the HaarCascade Classifier.

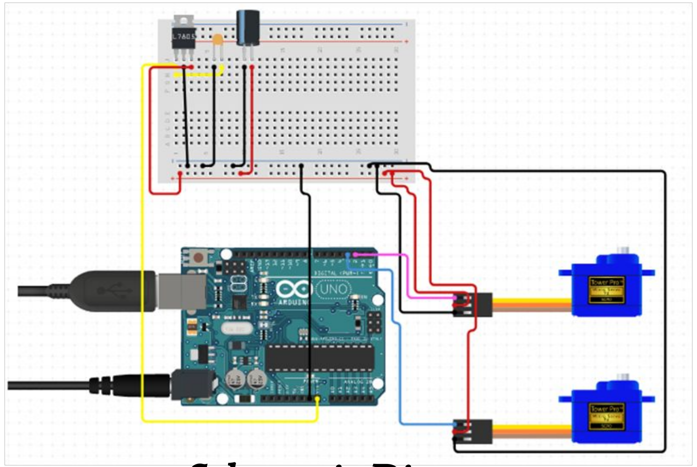
Arduino UNO Code and Implementation

- Basics of Arduino Programming
- Schematic Diagram and Connections
- Programming and Uploading the sketch on the Arduino UNO board

Arduino Programs are written in C/C++. The two important functions that are an essential part of the code are:

- setup() – this function run once at the beginning of the program and can be used to initialise certain variables and settings.
- loop() – this function is called repeatedly till the board is turned off.

The written sketches are first compiled to check if there are any compilation errors and then the board is connected to which the sketch is uploaded.

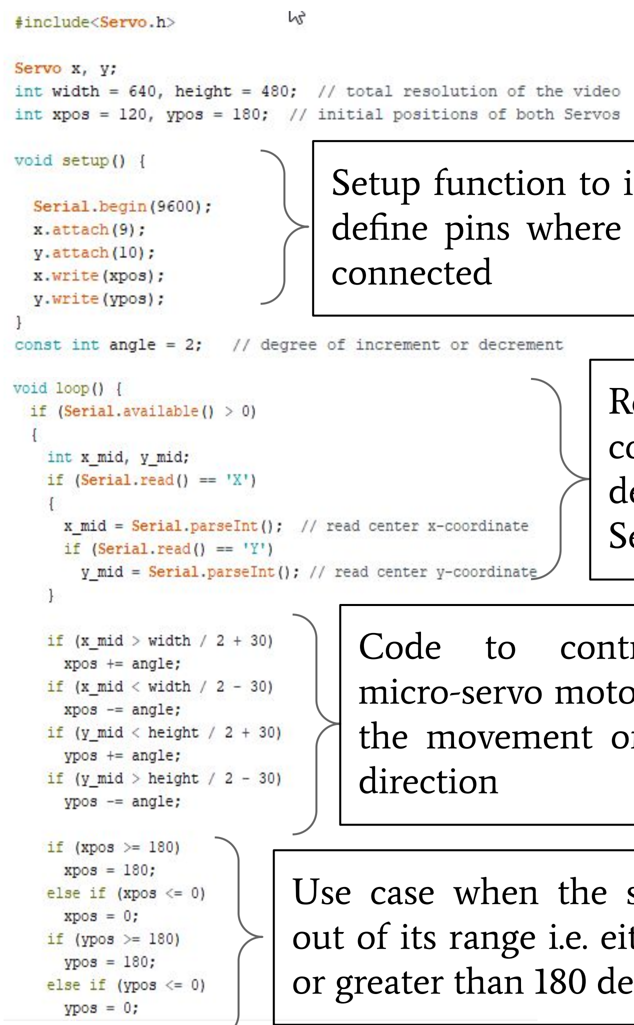


Schematic Diagram

The micro-servo motor has three wires where

- '**Red**' needs to be connected to a DC power supply of +5V,
- '**Brown**' is grounded and
- '**Orange**' is used to give instructions through Arduino to drive the motor.

The power supply of +5V is supplied through the Arduino Uno board "5V pin" to both the servo motors.



```
#include<Servo.h>

Servo x, y;
int width = 640, height = 480;  // total resolution of the video
int xpos = 120, ypos = 180;  // initial positions of both Servos

void setup() {

  Serial.begin(9600);
  x.attach(9);
  y.attach(10);
  x.write(xpos);
  y.write(ypos);
}
const int angle = 2;   // degree of increment or decrement

void loop() {
  if (Serial.available() > 0)
  {
    int x_mid, y_mid;
    if (Serial.read() == 'X')
    {
      x_mid = Serial.parseInt();  // read center x-coordinate
      if (Serial.read() == 'Y')
        y_mid = Serial.parseInt(); // read center y-coordinate
    }

    if (x_mid > width / 2 + 30)
      xpos += angle;
    if (x_mid < width / 2 - 30)
      xpos -= angle;
    if (y_mid < height / 2 + 30)
      ypos += angle;
    if (y_mid > height / 2 - 30)
      ypos -= angle;

    if (xpos >= 180)
      xpos = 180;
    else if (xpos <= 0)
      xpos = 0;
    if (ypos >= 180)
      ypos = 180;
    else if (ypos <= 0)
      ypos = 0;
```

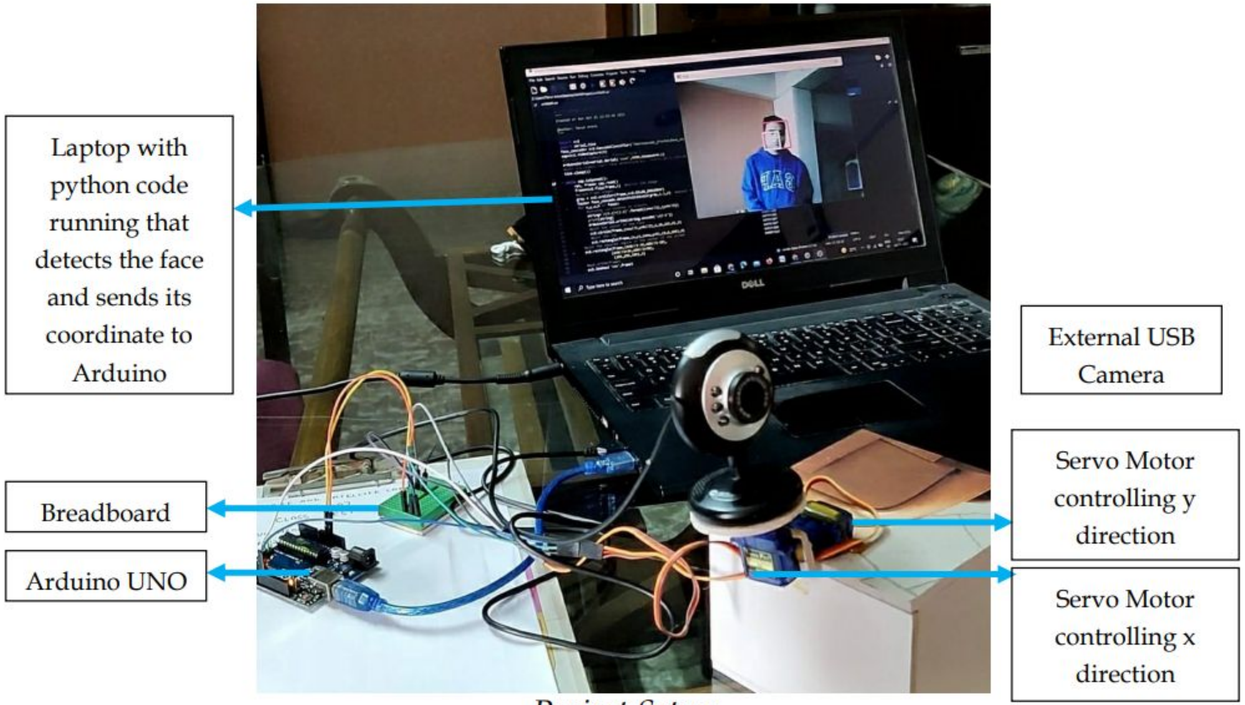
Setup function to initiate values and define pins where servo motors are connected

Reading the x & y coordinates of the face detected from Arduino Serial

Code to control the movement of micro-servo motors that eventually controls the movement of the camera in X and Y direction

Use case when the servo degree is out of its range i.e. either less than 0 or greater than 180 degrees.

Project Setup


Laptop with python code running that detects the face and sends its coordinate to Arduino
Breadboard
Arduino UNO
External USB Camera
Servo Motor controlling y direction
Servo Motor controlling x direction

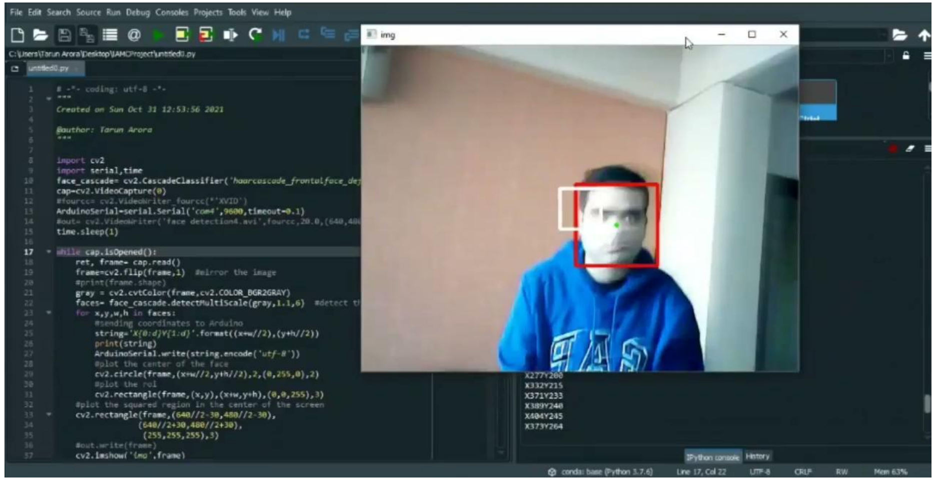
Project Setup

Both the Arduino UNO and the external USB camera are connected to the laptop through USB cable. The breadboard is used to supply +5V DC supply and ground to both the servo motors through Arduino Uno pins.

Through the laptop, we executed the Python program which detected the face in the frame of the camera and send the coordinates of the center of the face to Arduino IDE through Serial.

These coordinates are used by the Arduino UNO program to rotate the servo motors with precision in order to ensure that the detected face is always in the frame of the camera.

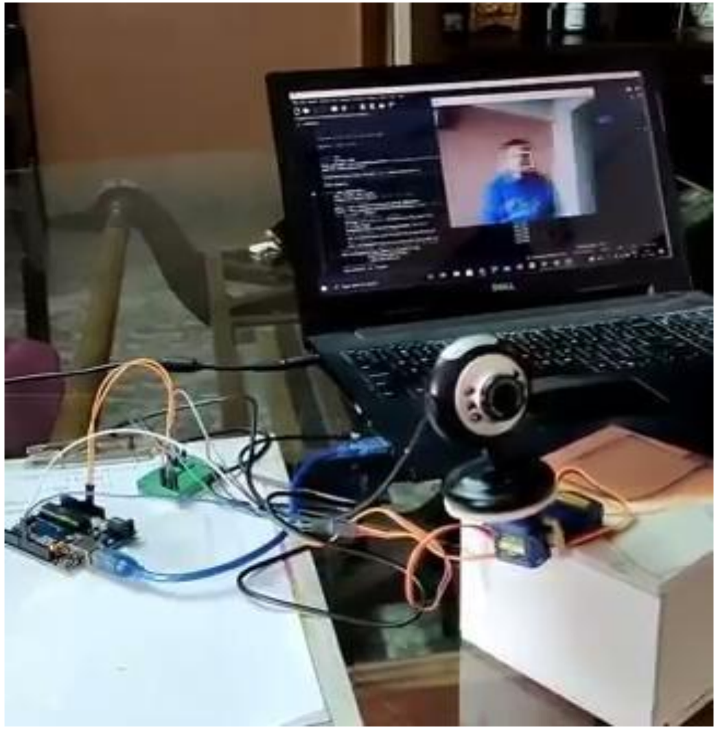
Results & Discussion


File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Tarun Arora\Desktop\IAMCProject\untitled0.py
untitled0.py
img
IPython console
History
conda: base (Python 3.7.6)
Line 17, Col 22
UTF-8
CRLF
RW

```python
# -*- coding: utf-8 -*-
"""
Created on Sun Oct 31 12:53:56 2021

@author: Tarun Arora
"""

import cv2
import serial,time
face_cascade= cv2.CascadeClassifier('haarcascade_frontalface_de
cap=cv2.VideoCapture(0)
#fourcc= cv2.VideoWriter_fourcc(*'XVID')
ArduinoSerial=serial.Serial('com4',9600,timeout=0.1)
#out= cv2.VideoWriter('face detection4.avi',fourcc,20.0,(640,48
time.sleep(1)

while cap.isOpened():
    ret, frame= cap.read()
    frame=cv2.flip(frame,1)  #mirror the image
    #print(frame.shape)
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces= face_cascade.detectMultiScale(gray,1.1,6)  #detect th
    for x,y,w,h in faces:
        #sending coordinates to Arduino
        string='X{0:d}Y{1:d}'.format((x+w//2),(y+h//2))
        print(string)
        ArduinoSerial.write(string.encode('utf-8'))
        #plot the center of the face
        cv2.circle(frame,(x+w//2,y+h//2),2,(0,255,0),2)
        #plot the roi
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),3)
    #plot the squared region in the center of the screen
    cv2.rectangle(frame,(640//2-30,480//2-30),
                 (640//2+30,480//2+30),
                  (255,255,255),3)
    #out.write(frame)
    cv2.imshow('img',frame)
```

```
X277Y200
X332Y215
X371Y233
X389Y240
X404Y245
X373Y264
```

Screen Recording of Laptop running the project



Video recording of entire setup

Summary

In this project, we have designed a prototype system for automatic face detection and tracking after studying various research papers and taking inspiration from them. This project is intersection of fields of Image processing, Computer Vision and embedded systems. We have implemented this project using OpenCV, HaarCascade Classifier, and Arduino UNO. The programming languages used to achieve our objectives are Python and C.

References

[1] PAMULAPATI, VENKATA SASANK, YEKULA SUMITH ROHAN, V. S. Kiran, SARANU SANDEEP, and MARAM SRINIVASA RAO. "Real-Time Face Tracking Using Matlab And Arduino." *Electronics And Communication Engineering, Vasireddy Venkatadri Institute Of Technology* (2018).

[2] Ayi, Maneesh, Ajay Kamal Ganti, Maheswari Adimulam, Badiganti Karthik, Manisha Banam, and G. Vimala Kumari. "Face Tracking and Recognition Using MATLAB and Arduino." *International Journal for Research in Applied Science & Engineering Technology (IJRASET)* (2017).

[3] Viraktamath, S. V., Mukund Katti, Aditya Khatawkar, and Pavan Kulkarni. "Face detection and tracking using OpenCV." *The SIJ Transactions on Computer Networks & Communication Engineering (CNCE)* 1, no. 3 (2013): 45-50.