Real time face detection using OpenCV, basing on a webcam mounted onservomechanism controlled by Arduino board

Author

Adam Stępień

Table of contents

Introduction

Main function of this device is face detection. The way the face is detected is implementation of Python code using OpenCV library. More precisely, to detect actual face I used haarcascade which is a pre- trained model for detecting human faces, and is very useful in such solutions. This model let us define the face in a precise way, detect eyes, smile, full body etc. Combing haarcascade and Python together, I made function that in real time follows our face, surrounding it in a red square with green point on the nose. Then, I created Arduino code which is responsible to control Pan-Tilt mechanism (2 axis servo mechanism) with webcam mounted on the top. Way that Arduino controls the servomechanism is quite simple. Every time the center of the camera (white box) doesn't fit the center of the face, servomechanism moves the camera up or down and left or right. We can see the preview on the screen of our laptop. Of course, due to the quality of the webcam (or the refresh rate of the camera) sometimes the centers do not match exactly together. This causes that servomechanism moves quickly in many directions, but even though, purpose of the device is kept.

Python Code

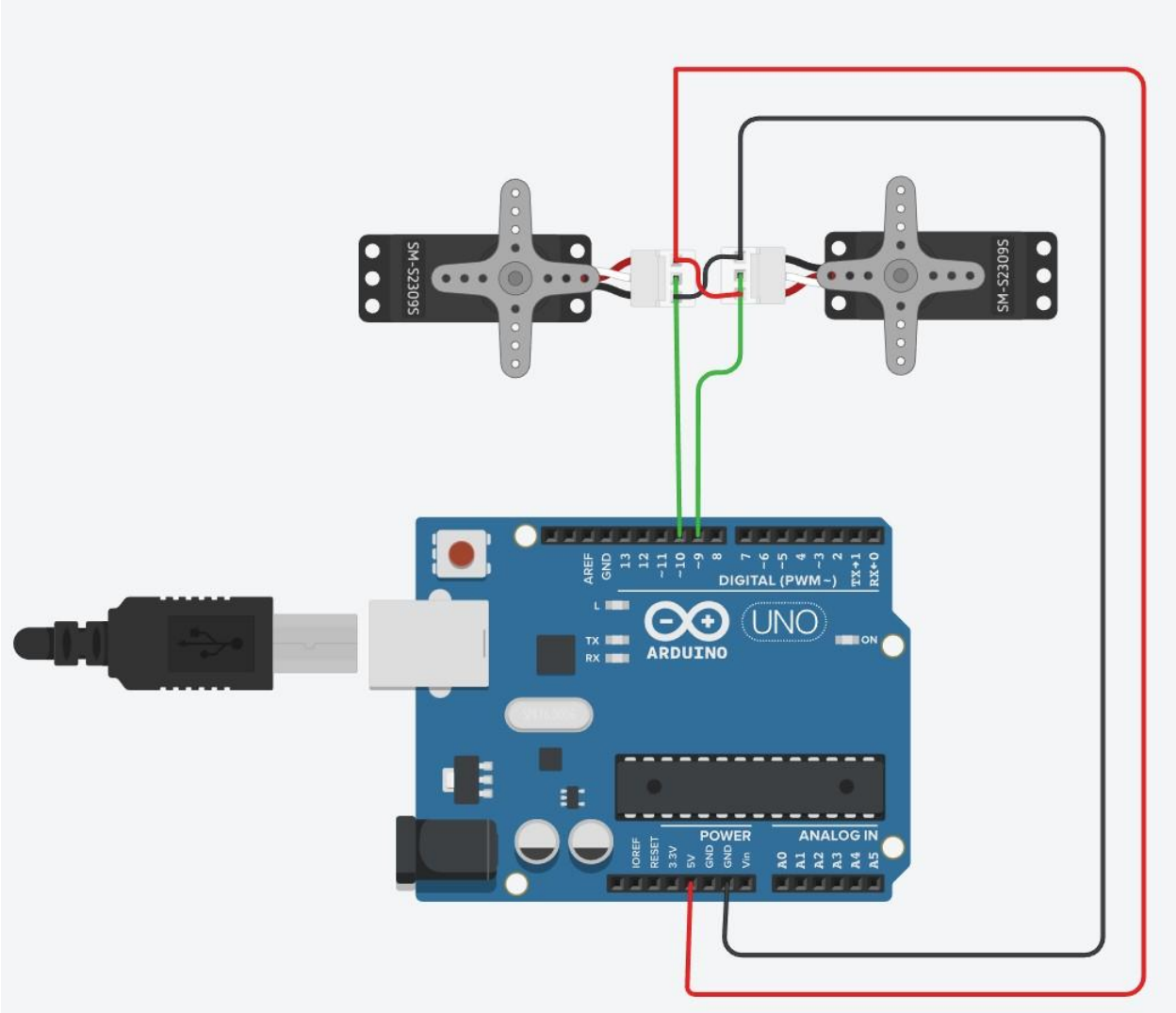
```
import cv2
import serial,time
face= cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
capture=cv2.VideoCapture(1)

ArduinoSerial=serial.Serial('com5',9600,timeout=0.1)

time.sleep(1)

while capture.isOpened():
    ret, frame= capture.read()
    frame=cv2.flip(frame,1) #mirror the image
    print(frame.shape)
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces= face.detectMultiScale(gray,1.1,6)  #detect the face
    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 on the nose
        cv2.circle(frame,(x+w//2,y+h//2),2,(0,255,0),2)
        #plot the rectangle
        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)

    cv2.imshow('I follow U',frame)
    #for testing purpose
    read= str(ArduinoSerial.readline(ArduinoSerial.inWaiting()))
    time.sleep(0.05)
    print('data from arduino:'+read)

    # press q to Quit
    if cv2.waitKey(10)&0xFF== ord('q'):
        break
capture.release()
cv2.destroyAllWindows()
```

Arduino Code

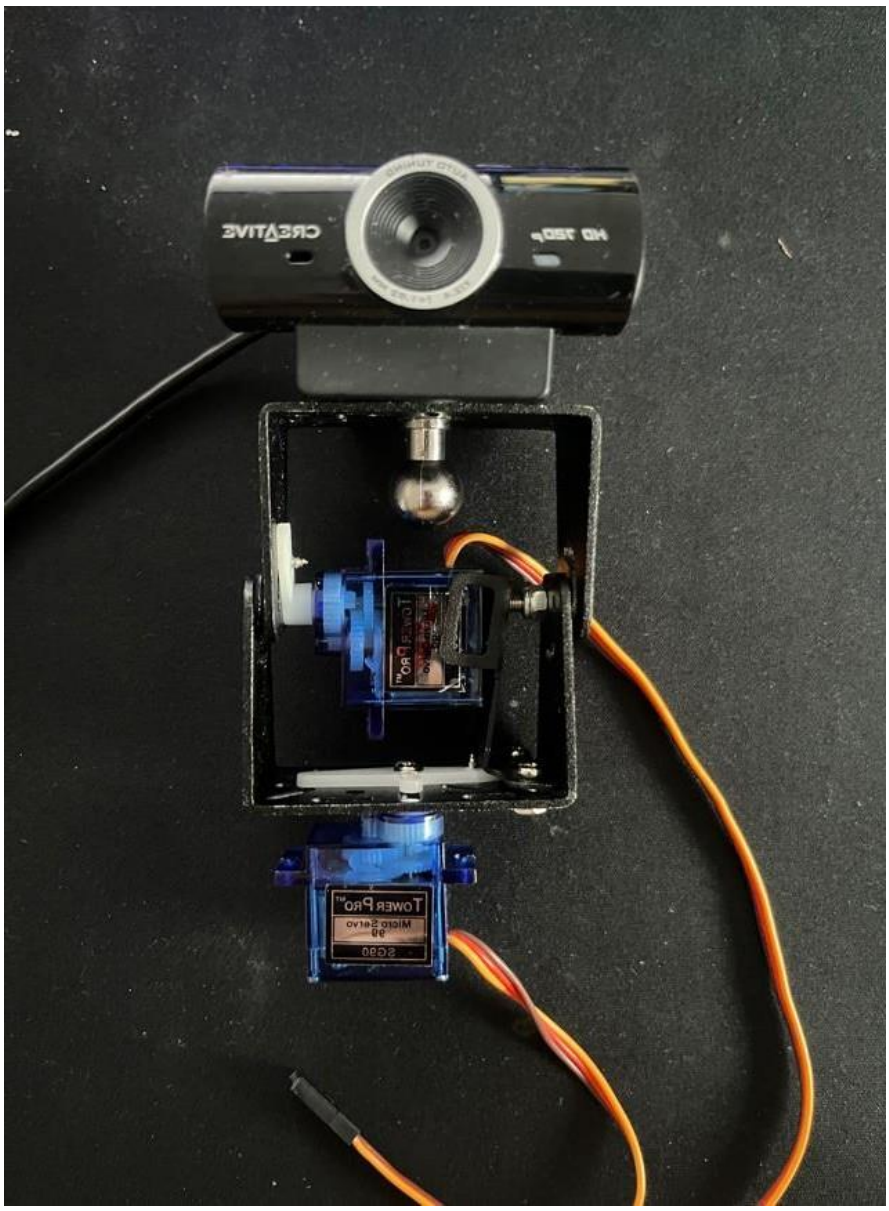
```
#include<Servo.h>
Servo x, y;
int width = 640, height = 480;  // total resolution of the video
int xpos = 90, ypos = 90; // initial positions of both Servos
void setup() {
 Serial.begin(9600);
  x.attach(9);
  y.attach(10);
  x.write(xpos);
  y.write(ypos);
}
const int angle = 2;   // degree of increment 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) // if the servo degree is outside its range
      xpos = 180;
    else if (xpos <= 0)
      xpos = 0;
    if (ypos >= 180)
      ypos = 180;
    else if (ypos <= 0)
      ypos = 0;
    x.write(xpos);
    y.write(ypos);
  }
}
```

Assembly

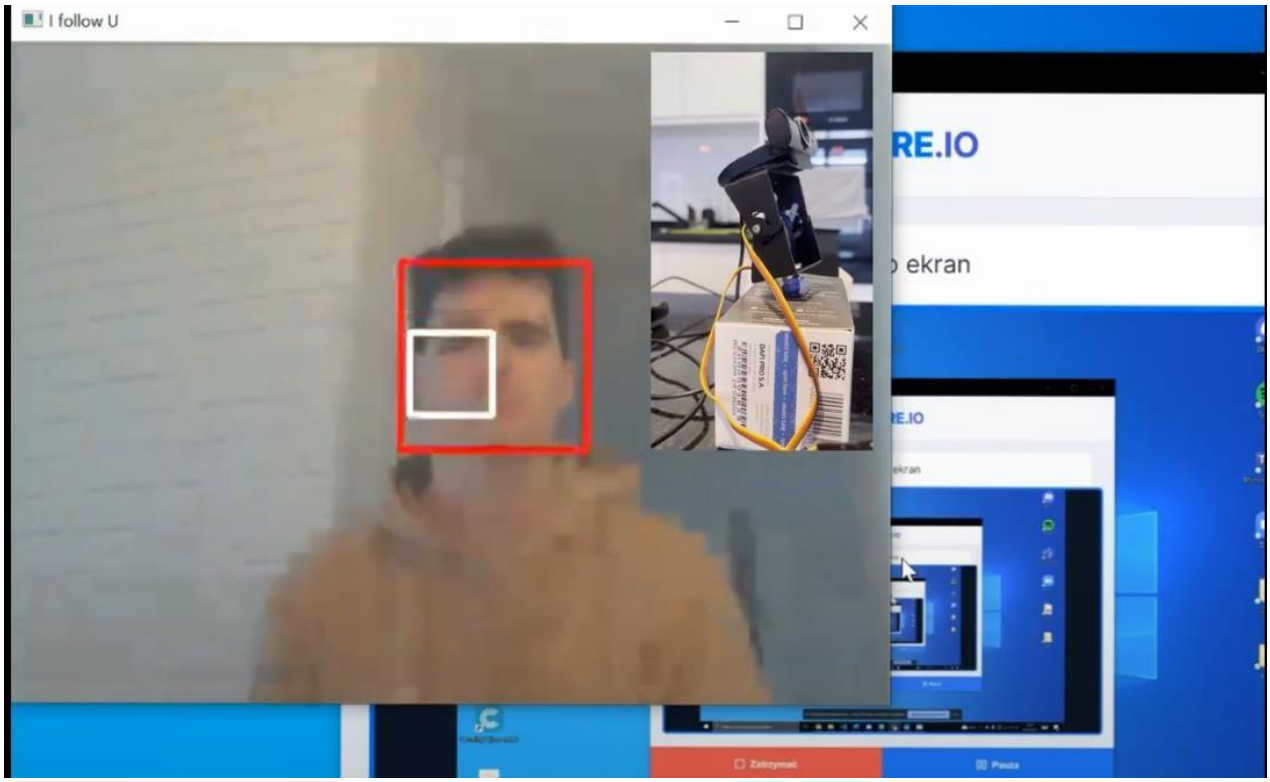
Servos connected to Arduino:



Pan-tilt configuration :



Youtube video, how the system works :



https://youtu.be/aS_7EnSCQEk

Summary

Summarizing all the work I have performed, I can tell that this project was fun. Not only the results are better then expected, but also I have learned many new things. First, usage of haarcascade module. The most important feature of the haarcascade classifier is its ability to detect objects in images at different scales, which means it can detect objects of different sizes in the same image. This is achieved by using a technique called "cascading," which involves applying the classifier in a sequence of stages, each stage using a different set of features to detect the object. Secondly, combining Python and Arduino. Way I did this is using the Arduino IDE to upload the code to the Arduino board and then use a python library like *serial* to communicate with the Arduino board. By opening a serial connection with the Arduino board you can send and receive data between python and Arduino. When it comes to coordinates calculation, OpenCV returns the face coordinates in terms of pixel values. The video resolution is set to 640*480. The coordinates describe the top-left pixel values (x and y) along with the height and width. I have used the center coordinates of the face for reference and can be calculated using x + width/2 and y + height/2 and can be seen as a green dot. These coordinates are sent to the Arduino for moving the angle of the camera. All in all, in my opinion project is made from the beginning till the end with precision. Video is a proof, that the system is working, i.e. camera is detecting the face and servomechanism are helping the camera to point center of the screen on the center of the face