Engineering Exploration

OWL
The Self-Tracking Camera

Project Report

F.Y. B.Tech.

SEMESTER II 2021-22
(COMPS)

Team:

1) Gaurish Baliga → 16010121010 → A1
2) Jagjit Singh Bhumra → 16010121022 → A1
3) Vishrut Deshmukh → 16010121043 → A2
4) Pargat Singh Dhanjal → 16010121045 → A2
5) Pratham Goenka → 16010421137 → H3

Introduction:

After attending the Quantum Computing workshop organised by the college, we observed a major flaw in the working of webinar cameras. The current webinar cameras do not move AT ALL, that means that if the whiteboard is wide and the teacher moves on to one corner of the board, they won't be visible at all. To solve this problem, we decided to create a webinar camera that tracks the person and keeps him in focus. We plan to do this with the help of image processing using OpenCV and Arduino.

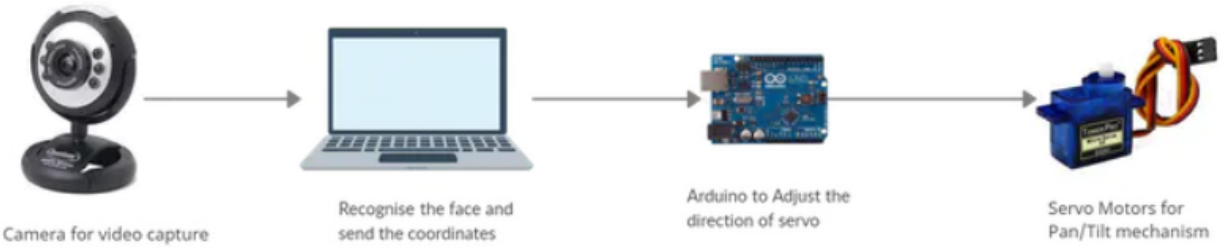
Facial recognition is a very useful tool incorporated in many modern devices to detect human faces for tracking, biometric and to recognize human activities. In this project, I have used the OpenCV's Haar-Cascade classifiers for detecting human faces and pan/tilt servo mechanism to track the user's face using Arduino UNO.

Problem Statement:

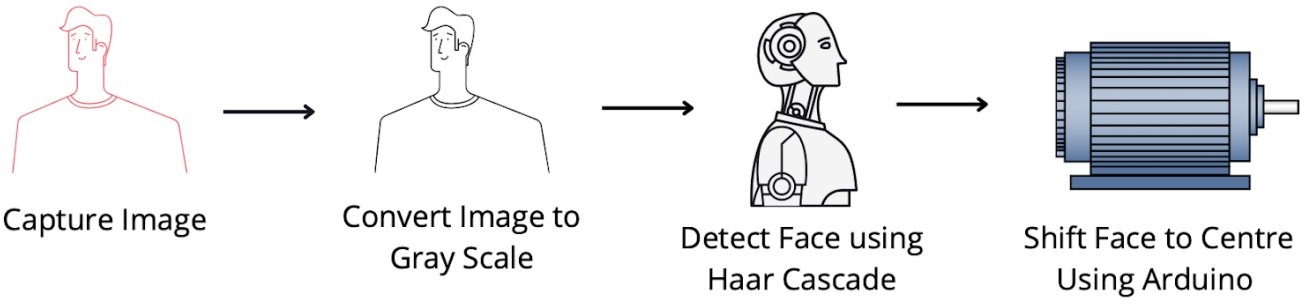
In current classrooms, conference rooms and dance centres, the main problem arises due to static cameras that need to be controlled manually to keep a track of the speaker/performer. An automated, easy to use, portable and inexpensive self tracking camera is the need of the hour in such areas. The product needs to be light (not more than 2 kilograms) but sturdy and should be able to mount itself to the wall/stand on the table with ease. Apart from this the device needs strong connectivity in terms of wifi and strong motors to carry the load, i.e the camera.

Block Diagrams:

External Working



Internal Working



Software Used:

1) Python:

- Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasises code readability with the use of significant indentation.
- Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming.
- It is often described as a "batteries included" language due to its comprehensive standard library.

2) Arduino C++:

- C++ is a general-purpose programming language created by Danish computer scientist Bjarne Stroustrup as an extension of the C programming language, or "C with Classes".
- The language has expanded significantly over time, and modern C++ now has object-oriented, generic, and functional features in addition to facilities for low-level memory manipulation.
- It is almost always implemented as a compiled language, and many vendors provide C++ compilers, including the Free Software Foundation, LLVM, Microsoft, Intel, Oracle, and IBM, so it is available on many platforms.

3) OpenCV Module:

- OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems.
- By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis.
- To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

4) Haar-Cascade Classifier:

- Haar cascades, first introduced by Viola and Jones in their seminal 2001 publication, Rapid Object Detection using a Boosted Cascade of Simple Features, are arguably OpenCV's most popular object detection algorithm.
- Sure, many algorithms are more accurate than Haar cascades (HOG + Linear SVM, SSDs, Faster R-CNN, YOLO, to name a few), but they are still relevant and useful today.
- One of the primary benefits of Haar cascades is that they are just so fast — it's hard to beat their speed.

Hardware Used:

1) Arduino Uno:

- Arduino UNO is a microcontroller board based on the **ATmega328P**. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button.
- It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

2) 9V SG90 Servo Motors:

- Tiny and lightweight with high output power.
- Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos.
- Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.

3) 1080p USB Webcam

- A webcam is a video camera that feeds or streams an image or video in real time to or through a computer network, such as the Internet.
- Webcams are typically small cameras that sit on a desk, attach to a user's monitor, or are built into the hardware.
- Webcams can be used during a video chat session involving two or more people, with conversations that include live audio and video.

4) Breadboard & Jumper Cables:

- A breadboard, solderless breadboard, protoboard, or terminal array board is a construction base used to build semi-permanent prototypes of electronic circuits.
- A jump wire is an electrical wire, or group of them in a cable, with a connector or pin at each end, which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.

How it Works :

Facial detection helps identify and localise human faces, and ignores any background objects in the surroundings. The OpenCV module uses a Haar-Cascade classifier, where each frame of the video is passed through stages of classifiers and if the frame ia accepted by all the classifiers, the face is detected, otherwise that particular frame is discarded, i.e the face is not detected.

The Python OpenCV program returns the cartesian coordinates of the image upon detection, along with its height and width. From these coordinates, the center-coordinates of the image can be calculated using '*x+width/2*' and '*y+height/2*'.

These coordinates are passed to the Arduino UNO using the pyserial library when the face is detected. The servos connected to the Arduino provide a pan/tilt mechanism where the camera is connected to one of the servos. When the coordinates of the face are away from the center, then the servo will align degree by degree (increment or decrement) to bring it towards the center of the screen.

Detecting the face

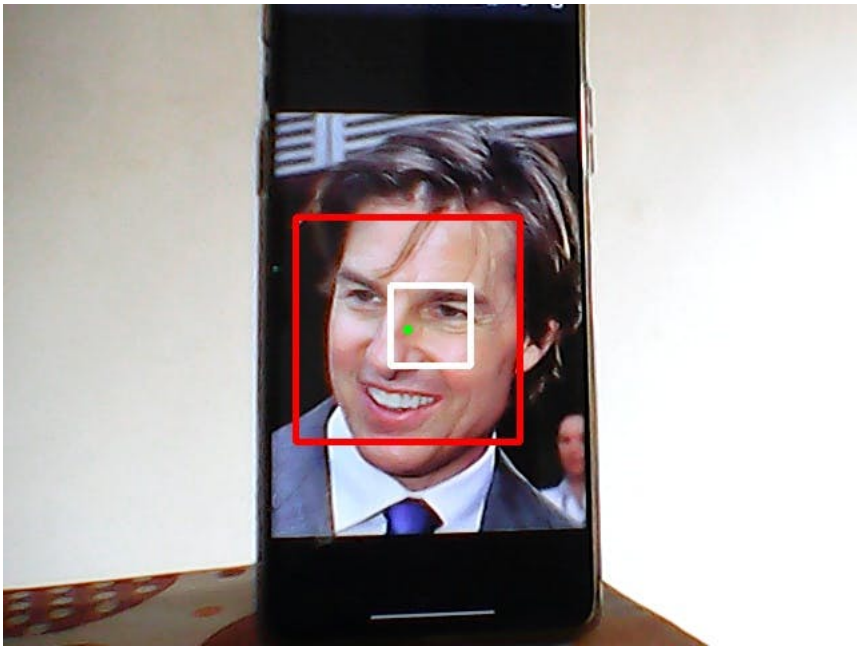
We have used '*haarcascade_frontalface_default.xml*' which is a pre- trained model for detecting human faces and can be downloaded from their repository on GitHub (here). Upon downloading, the xml file can be loaded using *cv2.CascadeClassifier()* function.

The function used for face detection is *cv2.CascadeClassifier.detectMultiScale()* with the *scale-factor* value and *minNeighbour* values set as needed (will differ according to webcam). This returns the cartesian coordinates of the image, along with the height and width. Increasing the *minNeighbour* can improve facial detection, but it sacrifices execution speeds, which would lead to delayed responses from the servos..

Calculating the Coordinates

The OpenCV program returns the face coordinates in terms of pixel values. The video resolution should be set to camera resolution. The coordinates describe the top-left pixel values(x and y) along with the height and width. We have used the center-coordinates of the face for reference, which can be calculated using '*x+width/2*' and '*y+height/2*'. These coordinates are sent to the arduino for changing the angle of the camera.

The square in the center of the frame in white describes the region within which the green dot must be. If it is outside the square when the face is moved, the servo will align the camera to bring it inside the region.



Sending Serial data to Arduino

This was challenging since sending the coordinates sequentially to the Arduino had slow response times. After spending some time figuring it out, we began looking for similar projects online until we found this project (here). We came to know about the Serial function *Serial.parseInt()* which takes integer inputs from an incoming serial of bytes (check here). Our method of sending the serial data is similar to the one used in that project.

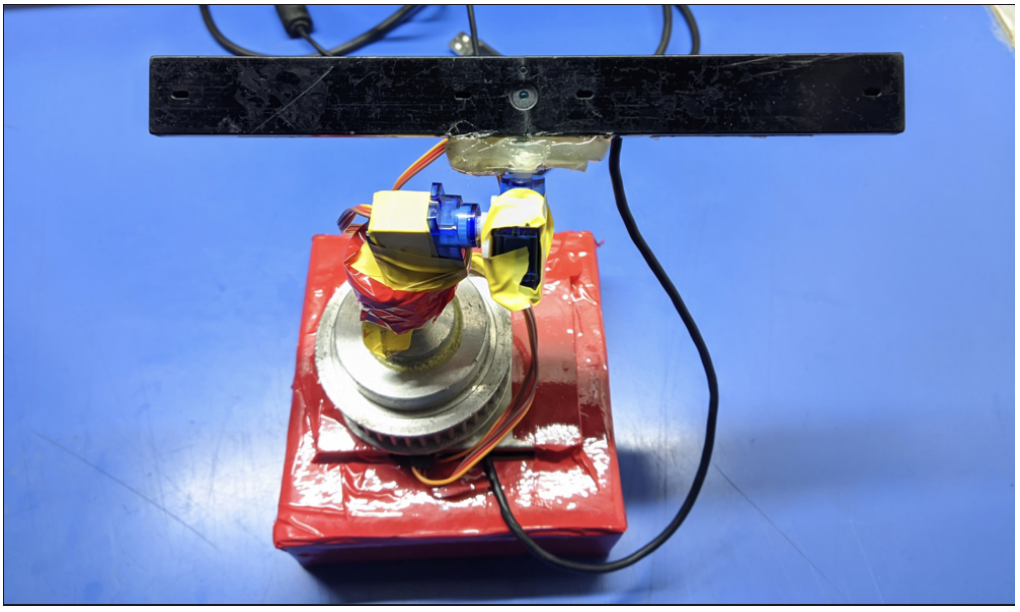
Python sends the center-coordinates in a single string. For example: "X110Y190", the value 110 after X represents center x-coordinate and 190 represents center-y coordinate.

Arrangement of Servo

We have attached the horizontal-moving servo on the shaft of the vertical-moving servo in which the camera is mounted. All the attachments are using simple rubber bands and some electrical tape (It is not recommended as we made use of existing material available to us, for a finished product it is recommended to use more permanent methods, and to solder wires instead of using jumper cables and breadboard).

Libraries and Installations

This project requires pyserial and OpenCV libraries which we downloaded using pip. We used similar or higher versions of Python (3.8) and OpenCV (4.4.0). We also made sure that the XML file for face detection is saved in the same directory as the python script.



The python script also required some modification by entering the correct COM port of the Arduino before execution (whenever it was ejected and reconnected).

As we are using 2 servos for tracking, an additional 9V supply was recommended (by means of an adapter) to the Arduino to provide sufficient current for both the servos. Since we didn't have it already, we made it with just the Arduino's 5V supply. In the absence of the extra power supply though, we noticed the motors wobbling when trying to move the camera.

Code:

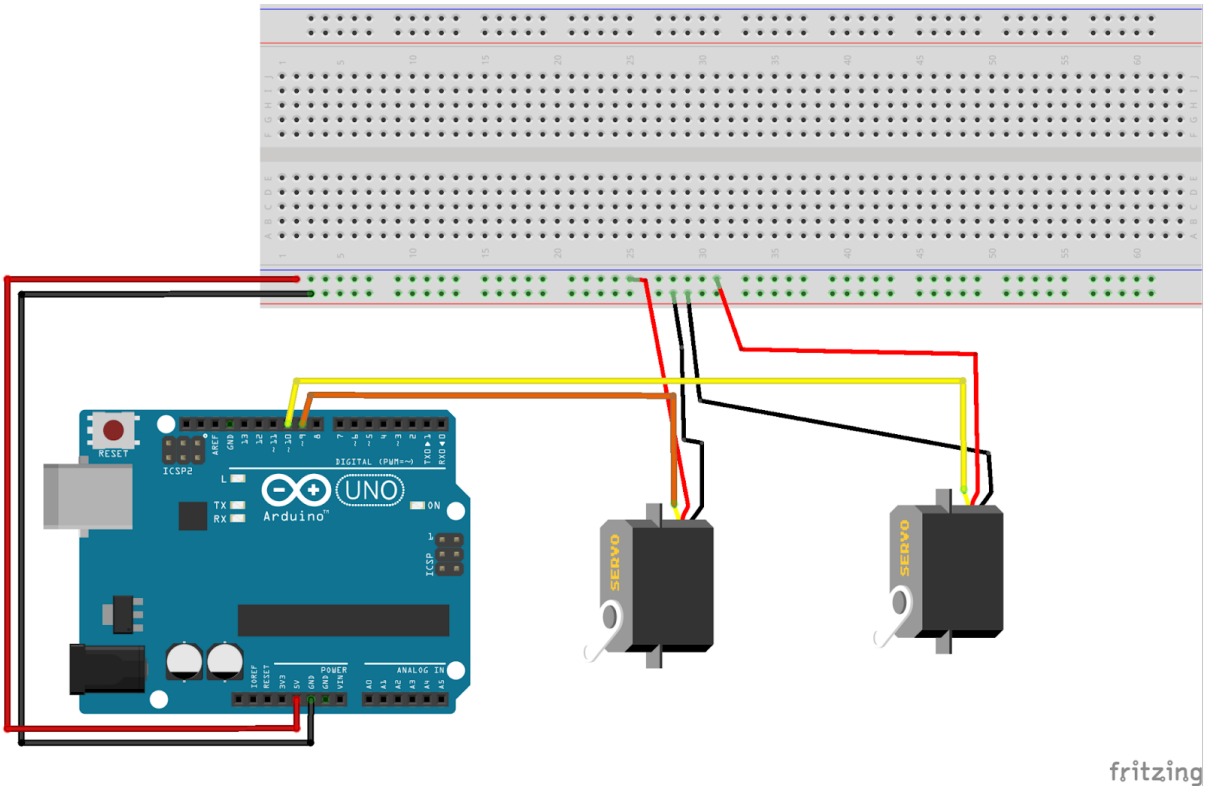
Python:

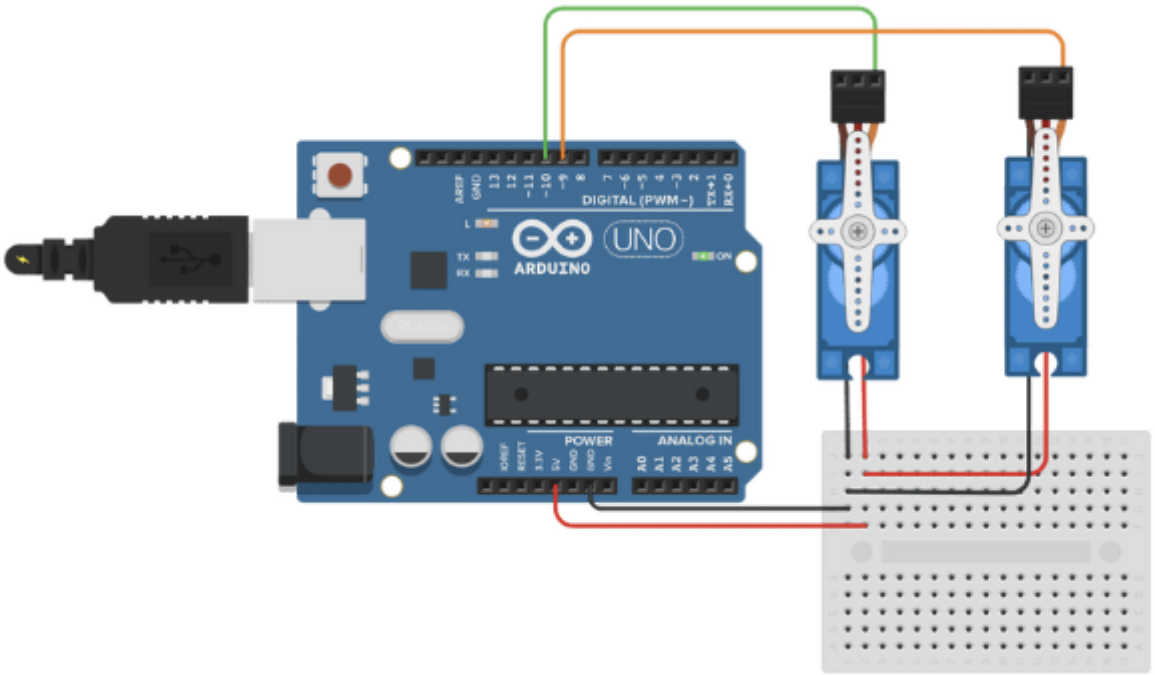
```
import cv2
import serial,time
face_cascade= cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cap=cv2.VideoCapture(0)
ArduinoSerial=serial.Serial('/dev/cu.usbmodem11301',9600,timeout=0.1)

while True:
   i = True
   ret, frame= cap.read()
   frame=cv2.flip(frame,1)  # vertical flip
   gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
   faces= face_cascade.detectMultiScale(gray,1.1,8)  #detect the face
   for x,y,w,h in faces:
       # sending coordinates to Arduino
       coordinates='X{0:d}Y{1:d}'.format((x+w//2),((y+h//2)))
       if i == True:
           print(coordinates)
           ArduinoSerial.write(coordinates.encode('utf-8'))
           i = False
       cv2.rectangle(frame,(x,y),(x+w,y+h),(100,255,255),1)
   cv2.imshow('img',frame)
   # press q to Quit
   if cv2.waitKey(10)&0xFF== ord('q'):
       break
cap.release()
cv2.destroyAllWindows()
```

Arduino:

```
#include <Servo.h>
Servo x, y;
int width = 640, height = 480; // Camera resolution
int xPos = 90, yPos = 90;      // initial positions of Servos

void setup()
{
 Serial.begin(9600);
 x.attach(9);
 y.attach(10);
 x.write(xPos);
 y.write(yPos);
}

const int angle = 1; // Change in degree

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

   // Bring servo to the center tiny square
   if (x_mid > width / 2 + 35)
     xPos += angle;
   if (x_mid < width / 2 - 35)
     xPos -= angle;
   if (y_mid < height / 2 + 35)
     yPos -= angle;
   if (y_mid > height / 2 - 35)
     yPos += angle;

   // if outside y servo is out of its range
   if (yPos >= 180)
     yPos = 180;
   else if (yPos <= 0)
     yPos = 0;

   // Writing the calculated values
   x.write(xPos);
   y.write(yPos);
 }
}
```

Schematic:


SERVO
SERVO
fritzing


DIGITAL (PWM~)
UNO
ARDUINO
POWER
ANALOG IN

Advantages and Future Applications:

- We have achieved facial tracking using openCV which keeps the camera stable when the user's face is not towards the camera.
- The OWL has 360° range of motion which allows tracking around the entire workspace/classroom.
- It has a simple USB interface which increases the connectivity
- The design is very stable to reduce the imbalance and wobble due to the momentum generated by movement of the camera.
- The setup despite being stable is very much portable with a new weight og just around 660 grams,
- The Owl is very easy to use with just a click of a button. There is no need to make several tweaks to make it work perfectly.
- The software is also lightweight with easy and quick setup.
- The estimated selling price of our product is very low compared to other competitors present in the market as of now.

GitHub & Presentation Links:

Github :

Pargat-Dhanjal/Smart-Camera: Engineering Exploration project for FY BTech (github.com)

Owl-Smart Webcam (pargat-dhanjal.github.io)

Presentation :

OWL Smart Camera Presentation (canva.com)

(Gantt Chart and Survey Reports are in the PPT given above)

Conclusion:

We would like to thank all of our teachers for Engineering Exploration for helping us to make the most out of this course and helping us with our final project. Your support has surely made us grow and become capable of developing such projects while coordinating as a team. The OWL wouldn't have been possible without your help and required materials from our esteemed college. Such projects help our community in day to day tasks which as students is our responsibility to develop. We hope that the OWL becomes a successful product one day under your guidance.
Thank you for giving us this opportunity.