SCHOOL OF ELECTRONICS ENGINEERING

ECE2010

CONTROL SYSTEMS

A REPORT ON

Smart Surveillance System

Using OpenCV and Linear SVM

Dr. Bagubali A

(School of Electronics and Communication Engineering)

By-
Sudhanshu Singh (20BEC0151)
Vinyas Shetty (20BEC0780)
Ajinkya Bagmar (20BEC0781)

ACKNOWLEDGEMENT

We would like to thank our professor, **Dr. BAGUBALI A** for providing us with this opportunity to learn and explore the topic which we never would have done alone. We also are thankful to all the writers of the mentioned references because of which we were able to understand the topic with ease and thorough knowledge and thus were able to present this report.
Thank you again for guiding us towards the right direction and pointing out suggestions that significantly helped us in improving the project.

DECLARATION

I hereby declare that the project work entitled "**Smart Surveillance System- Face recognition Using OpenCV and Linear SVM"** submitted to **Vellore Institute of Technology, Vellore** in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in **Electronics and Communication Engineering,** original and not copied from any source without proper citation. I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Abstract

The modernization of technology in the field of security and biometrics has taken a big jump. Face recognition is one of its biggest achievements. Face recognition is one of the most sought-after technologies in the field of machine learning. In this project, we have approached the concept of human face tracking through the method of gradients. And these obtained faces are then encoded to be compared with the database and eventually get recognized or in simple terms Facial Recognition. Using this concept, we proposed a surveillance system that uses facial recognition. This system could recognize the faces of different persons based on the database that it has been given. These recognized faces can have many applications such as marking attendance and recognizing any fugitive or an intruder in a house. We have also implemented an algorithm that allows the camera to move and follow the recognized person through servo motors which provides it a greater field of view We have also explored the ideasof improvement we could import into our project.

Introduction

Our Project focuses on detecting faces, recognizing and tracking them. We have used the Histogram of faces method to find the face and deep learning to recognize them. Libraries of facial characteristic measurements are obtained by comparing thousands ofdifferent faces through deep learning help us to obtain measurements for the test face in milliseconds and then we compare it with our database of saved faces using an SVM classifier to recognize the name of the person. SVM classifier isused because it gives very accurate results and is asimple linear system that just compares the encoding of the images on the database with that of the image we uploaded after which we have also mounted the camera onto two servo motors to keep the focus on a moving face and obtain better results.We also made our project able to move with only a detected face. The project has wide applications not only in security and surveillance but also in the Smart attendance system.

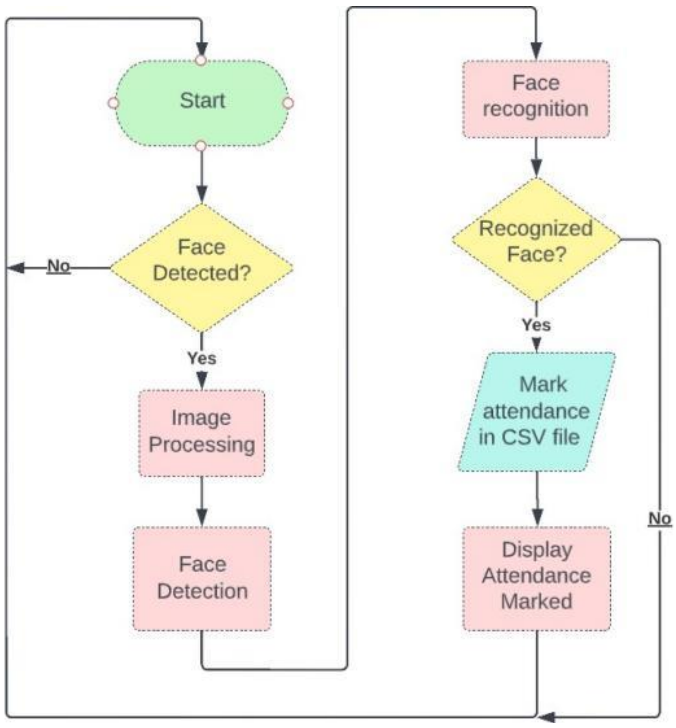
Problem Statement

As technology has advanced there is a lot of security breach and perpetrators in the digital world. Hence there is a need for a detection system where you can detect and eliminate the defaulters. To tackle this, we have proposed a project that involves face detection using a PID controller and linear SVM. This technology is expanding at a rapid pace, and it is used in a variety of applications, including device unlocking, banking, tourism, police enforcement, building security, etc and we are trying to increase the accuracy of the technology.

Proposed Analogy

The ML technique we are working on is a Support Vector Machine. It is a supervised classification algorithm where we draw a line between two different categories to differentiate between them i.e., we draw a line between two different categories to differentiate between them. Our idea is that the camera shall move with the recognized person's face to a certain extent Hence we will use an Arduino Uno to communicate with the servos. After mounting the camera onto two servo motors to keep the focus on a moving face and obtain better results. The servo connected to the Arduino provides a pan/tilt mechanism where the camera is connected to one of the servos. The values we get from the cartesian coordinates as such it let the servos move smoothly. So here we used a PID controller to keep the green dot in the center of the screen.

Algorithm



Literature Review

Pattern detection in images using classifiers is one of the most promising research topics in computer vision. There are a large number of practical applications for face detection and current work even suggests that any specialized detectors can be approximated using fast detection classifiers. In this project, an algorithm that will detect faces from the input image with less false detection rate using combined computer effects vision concepts. This algorithm uses the concept of skin color recognition, edge detection, and extracting different features from the face. The result is supported by the obtained statistics from the calculation of the parameters defining the parts of the face. He also implements the project high-performance Support Vector Machine concept used for image classification into face and no face classes. This classification is based on a set of training data and indicators brightness values, chrominance values, saturation values, elliptic values, and nose, eye, & mouth values map values [1].

Face recognition has been one of the most active areas of research in the last two years decades. Attempts are being made to understand how one recognizes another human face. It is widely accepted that face recognition can be based on structural information and non-structural/spatial details. In this study, he uses differential observation using Custom/docking characteristics of many built-in facial features and artificial neural networks. The proposed method aims to obtain a face feature by reducing facial features such as eyes, nose, mouth, and face depending on the importance of facial features. The face recognition system developed in this paper will inform the human face and evaluate the current percentage accuracy. Therefore, this work is for human face recognition and includes the percentage of facial expressions. The implementation of this function also offers many applications such as photos, biometric data in bank lockers, etc. [2].

Deep scientific use of computer technology applied in the fields of artificial intelligence and machine learning mainly focused on image processing and pattern recognition. Techniques like ours are widely used to recognize real objects including human faces etc. So, we can use these techniques to recognize a person from images. Using facial recognition modules from a huge collection of python libraries, we are able to train a model to recognize people while wearing masks. Since half of the facial features are lost when wearing masks, it is, therefore, crucial to develop a technique to recognize faces in this way. This specific face-detection technology is used in biometrics, video surveillance, etc. Therefore, it is of utmost importance to increase both security and efficiency while speeding up recognition [3].

A Face Detection Algorithm based on AdaBoost was studied, which is a cascaded multi-classifier face detection algorithm that can achieve real-time face detection in the system. It is an algorithm for automatically generating a cascade classifier in the training process, which can effectively prevent the phenomenon of overtraining. Adaptive Boosting is a machine learning technique used as Ensemble Method. The most common algorithm used with AdaBoost is single-level decision trees, that is, decision trees with only 1 split. Compared with the traditional AdaBoost face detection algorithm and the feature subspace algorithm, the detection speed of the algorithm presented in this paper is not only higher by almost 10 percent, but also the detection speed has been improved more than once. It is also very resistant to changes in lighting, position, etc., and is suitable for a real-time face detection system [4].

Facial recognition is a rapidly growing and interesting field that is gradually being used. A huge amount of facial recognition computations has been done a long time ago. For face detection, we use a HOG (Histogram of Oriented Gradient) based face detector, which provides more accurate results than other machine learning algorithms such as HAAR Cascade. In the recognition process, we use CLAHE (Contrast Limited Adaptive Histogram equalization) for pre-processing before using HOG, which is a standard feature extraction technique. HOG features are extracted for the test image as well as for the training images. And finally, we use SVM (support vector machine) for classification. SVM classifies HOG features. A pre-processing technique is used to remove noise, enhance contrast, and even out the lighting. The result of this paper shows accountability and productivity in better performance of face recognition [5].

Results and Discussions

We used python as our base language due to its versatility and its ability to give concise and readablecode. While complex algorithms and versatile workflows stand behind machine learning and AI, Python's simplicity allows developers to write reliable systems. For showing the working of the face-recognition code,we would mostly use this image and demonstrate how it is working as well.

Finding all the faces

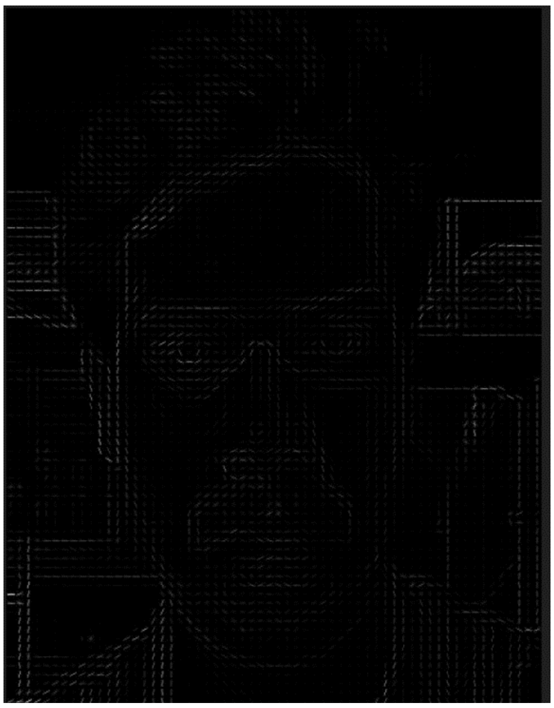
The first step is obviously to detect the face or simply face detection. We need to find each face in the photograph or video frame to tell them apart. For face detection, we are using a histogram of oriented gradients, or just HOG for short. We start by making the whole image black and white because we don't need color to find a face. It can be seen below that we have successfully filtered the image to black-and-white form.

Model Image

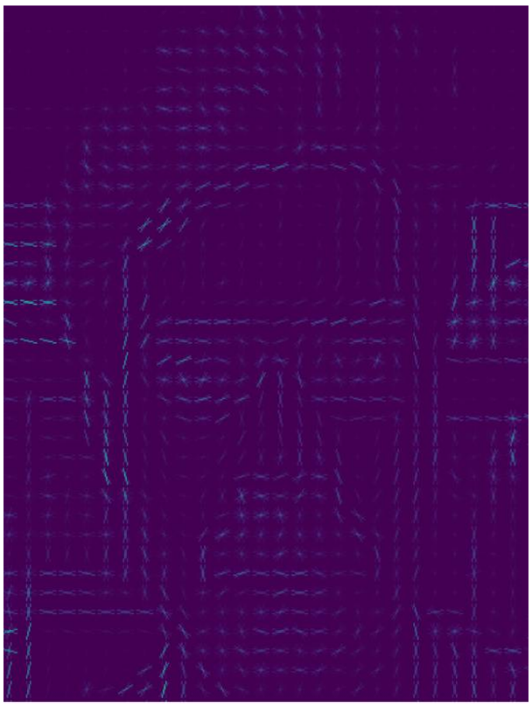
Black and White Image

We then look at every pixel and the pixel surrounding a particular pixel. After that, we try to find the direction of the pixel through which the image is getting darker. We would then make an arrow to represent this direction. When this same process is repeated for every pixel in the image, every pixel is replaced with an arrow. These arrows are called gradients. They show the flow from light to dark across the entire image. Pixels of the same person analyzed in different lighting conditions give different results or pixel values thus making them separate images to analyze. But by making them into gradients both the darkest and the lightest image will end up with the exact same pattern thus making our job much easier.

But applying it to every single pixel results in a loss of background data. It is just sufficient if we could see the basic flow of light along the pixels of the image. It gives us the basic pattern of the image. To do this we can break the image into a set number of squares containing a certain number of pixels. In the output image given below, it could be observed that we have obtained the gradient of the given face. One can observe it is oddly like outlining the facial feature of a face.



Gradient Image

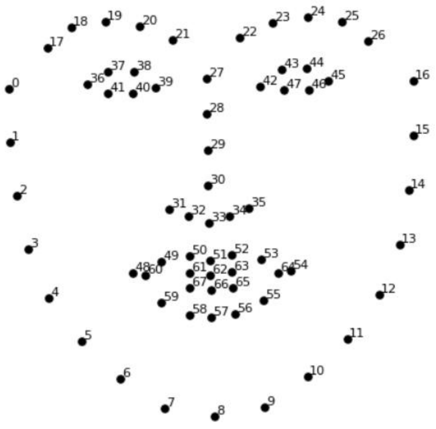


Scaled down Gradient Image

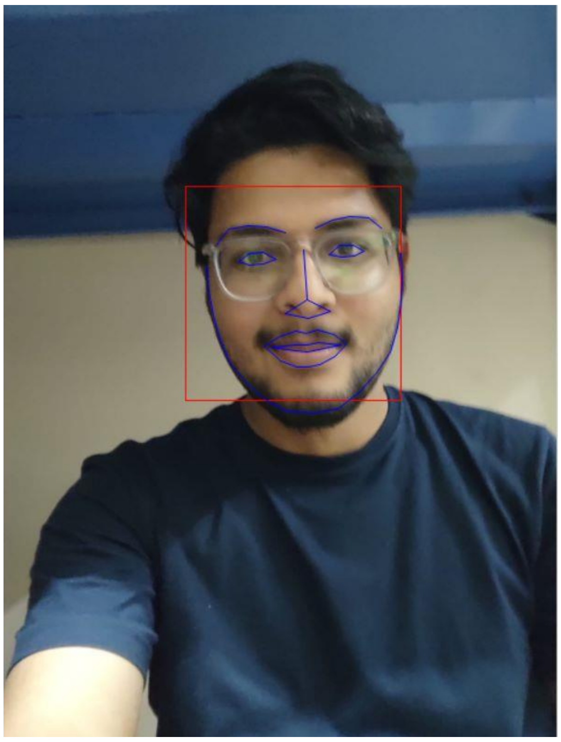
We found that it worked best when we took a square of 16*16 pixels each. In each of these squares, we shall count the number of gradients in each direction and replace them with the square in the image with the arrow direction that was in majority. We would now compare the obtained gradient of the image with the other gradient patterns that were extracted from a bunch of other images using this process.[6]

Posing and projecting faces

The above-proposed method would only work for straight faces that are looking at the camera. But if the photo is taken from a side profile the method would struggle to form a face with the gradients. It might give completely different results for the same person the photo was taken from an angle. To tackle this issue, we decided to wrap the picture so that the eyes and lips are always in the same place in the image.



The basic idea is we would find 68 points that exist on every face-the top of the chin, the outside of the eye, the inner edge of each eyebrow, etc. Then we 3 will train the machine to find these points on the face. Now that we have obtained eyes and mouths we will simply resize and rotate the image such that they do not distort and keep their parallel lines as well. Now no matter the direction in the face is captured, we can center and reposition the image such that our algorithm of HOGs works on it. Below we have again shown this with the same picture that we have created the gradient of.

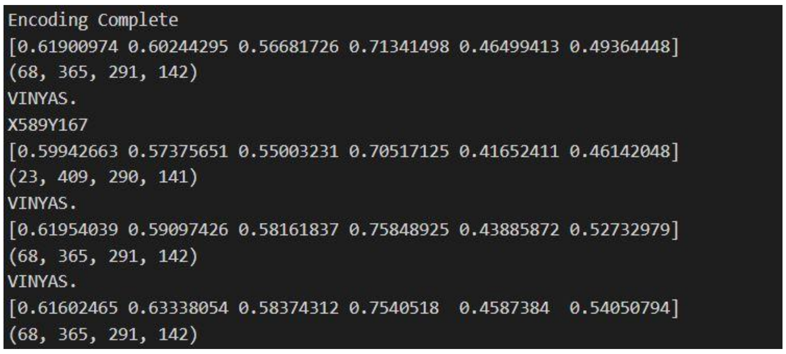


This image shows facial features outline.

After this we will do encoding of the face to recognize particular face from the database of various images.

Encoding faces

Now the real problem is how do we tell apart these images or how do we recognize them. A simple way is that we could directly compare the obtained gradient of an unknown face with a known one and it should just do a fine job. But that is not a proper approach to these problems. An app like Instagram with billions of users and photos cannot possibly loop through all of them to compare every newly uploaded image. This will take too long of time and we possibly cannot rely on it as we want them to be recognized in milliseconds. We don't need to compare the entire image with each other. We could extract only certain measurements and compare them between the known and unknown images and recognize the face with the closest measurement.



```
Encoding Complete
[0.61900974 0.60244295 0.56681726 0.71341498 0.46499413 0.49364448]
(68, 365, 291, 142)
VINYAS.
X589Y167
[0.59942663 0.57375651 0.55003231 0.70517125 0.41652411 0.46142048]
(23, 409, 290, 141)
VINYAS.
[0.61954039 0.59097426 0.58161837 0.75848925 0.43885872 0.52732979]
(68, 365, 291, 142)
VINYAS.
[0.61602465 0.63338054 0.58374312 0.7540518  0.4587384  0.54050794]
(68, 365, 291, 142)
```

Encoded values of the face

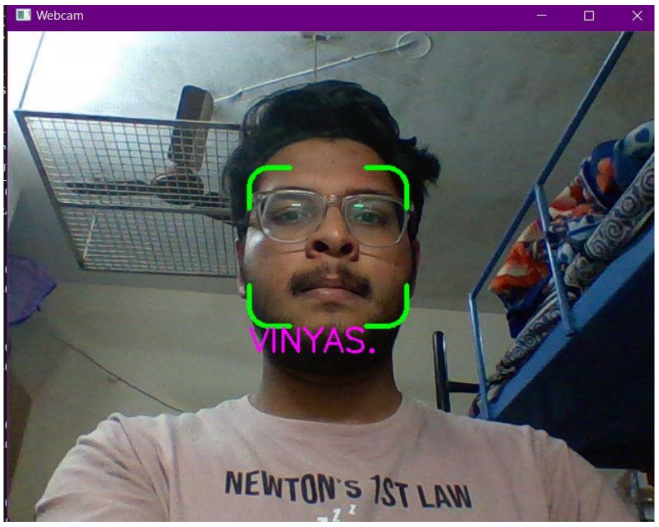
We would use deep learning and let the computer decide which measurements make the most sense to it. But instead of finding the locations of each object, we are going to train it to find 128 measurements of the face.

We do this by giving the machine three different images:

1. Image of the known person
2. Another image of the same person
3. Image of a totally different person

The algorithm generates the measurements of all three images. It then makes some adjustments such that the measurements of the first and second images are comparable while that of the second and the third are far apart from each other. We make the machine repeat this a million times with thousands of different faces to make it reliably generate these 128 measurements for each person. This process of training actually requires a high computation power. But once it has completed this process it could generate the measurements even for the faces it had never seen before. So, in this project, we have just used the pre-trained network to get the 128 measurements for our test image.[7]

Finding the Name of the person

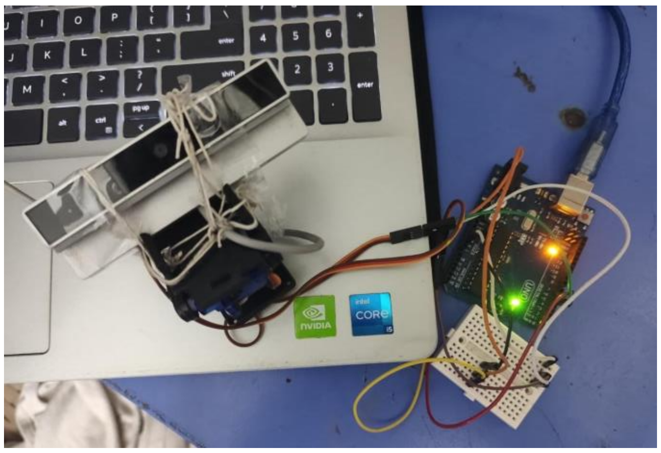


We now need to compare the obtained measurements with the measurement in our database and extract the name of the person from the database with which it has the closest match. We simply use a linear SVM classifier for this task. As can be seen in the picture below, it has successfully detected the person in the image and given the name of the said person.

In this image, we can see the name of the person is detected from the database of the images.

This total process is applied to a single image. We can do this with a camera as well. The output of a 24 FPS camera can be seen as 24 images per second are processed one after the other and the output is overlapped with each other which finally results in a live surveillance system. To show that this algorithm works we took a completely different picture of the model. This picture was taken under low light and at a different angle but as you can see it still detected his face perfectly.

Hardware Circuit

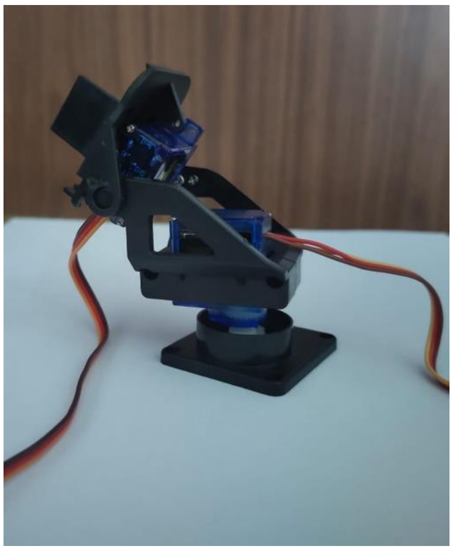


In our project, we have used two servo motors, Arduino UNO, and one USB-based Camera.

With the help of threads, we fixed the camera and mounted it over the servo motor. We have written code for the same in Arduino IDE.

Two servo motors are used in the project for the greater field of view of the camera for face detection which increases precision for detecting of faces.

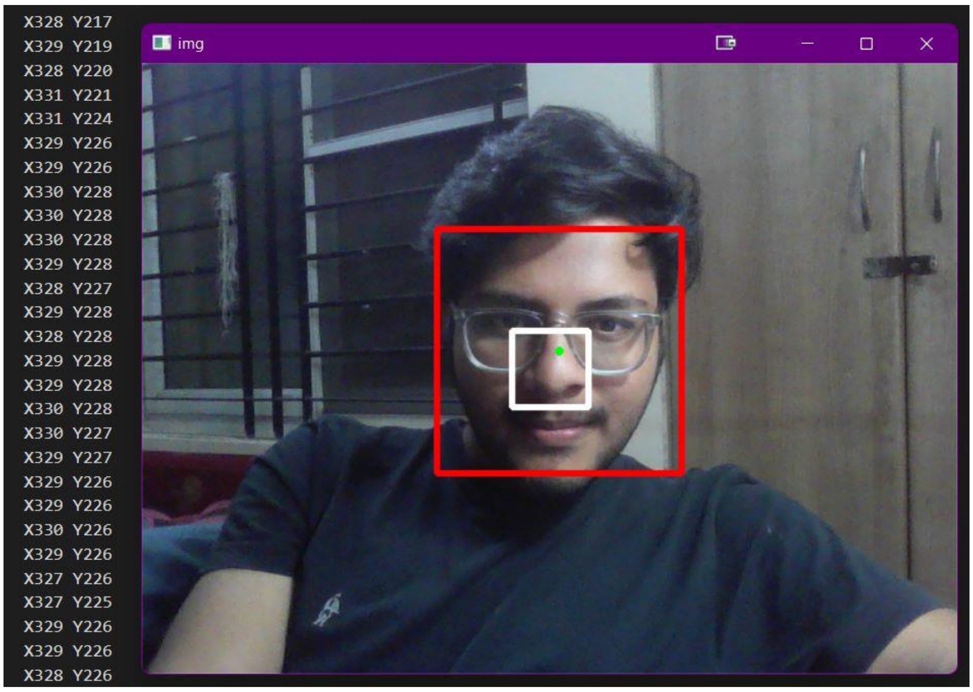
Moving the servos with the face



Our idea is that the camera shall move with the recognized person's face to a certain extent as it can have several applications on its own that we have discussed later. To do this we have used a simple serial in and serial out of data between our classifier and the servos. We used an Arduino Uno to communicate with the servos. The logic we have used is rather a very simple one. From OpenCV, we extract the cartesian coordinates of the detected and recognized image with height and width. With these values, we calculate the center of the image by simply using the equation: -

$(x + width)/2$ and $(y + height)/2$.

These coordinates are passed to Arduino UNO using the pyserial library when the face is detected. The square in the center of the frame in white describes the region within which the center of the face i.e the green dot must be. If it is outside the squared region when the face is moved, then the servo will align the camera to bring it inside the region. [8]



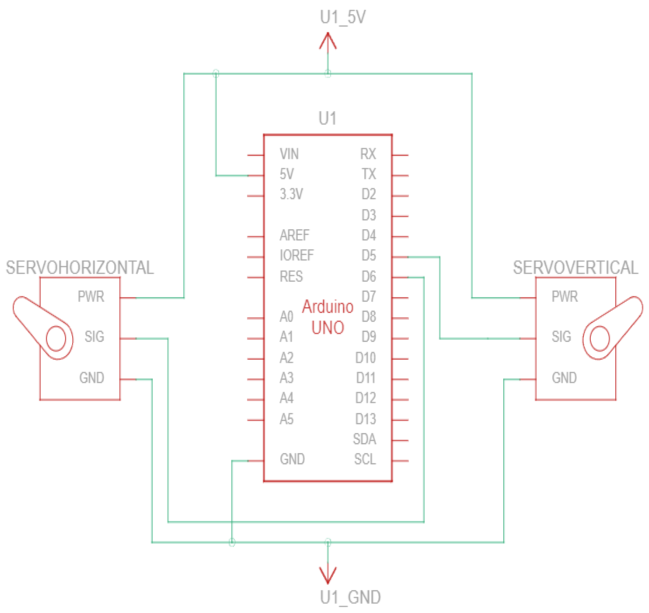
```
X328 Y217
X329 Y219
X328 Y220
X331 Y221
X331 Y224
X329 Y226
X329 Y226
X330 Y228
X330 Y228
X330 Y228
X329 Y228
X328 Y227
X329 Y228
X328 Y228
X329 Y228
X329 Y228
X330 Y228
X330 Y227
X329 Y227
X329 Y226
X329 Y226
X330 Y226
X329 Y226
X327 Y226
X327 Y225
X329 Y226
X329 Y226
X328 Y226
```

In the above image, we can see that with the help of a servo motor we are detecting face more accurately. If a person’s face is moved left or right then the servo motor will also align with the face movement. In the left panel, there are coordinates of the image are written which will change when the person’s face is not stable.

Schematic

Now since the coding part of both python and Arduino Uno is over let's see the basic schematic of our project and how we put it together.

As can be seen, by the schematic, we have connected the "Horizontal" or x-axis servo motor to pin 6 of the Arduino and the "Vertical" or y-axis servo motor to pin 5. Obviously, Arduino also has to be grounded. We could also supply a 5V power source if needed in case the camera that is mounted is a little heavy. Since we have used a normal webcam, it was not needed. The Arduino Uno is then connected to the PC with the USB port.



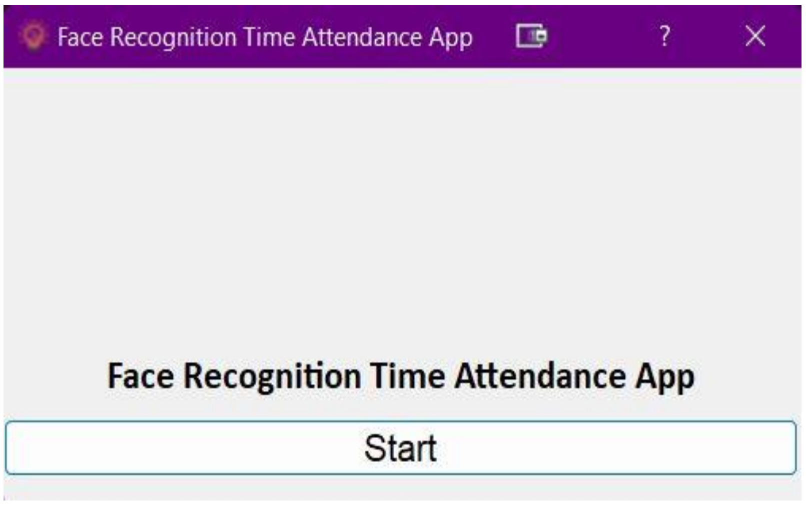
Through this port only we pass the information of coordinates generated from the code to the Arduino which then instructs the servo motors to align themselves according to it.

Applications

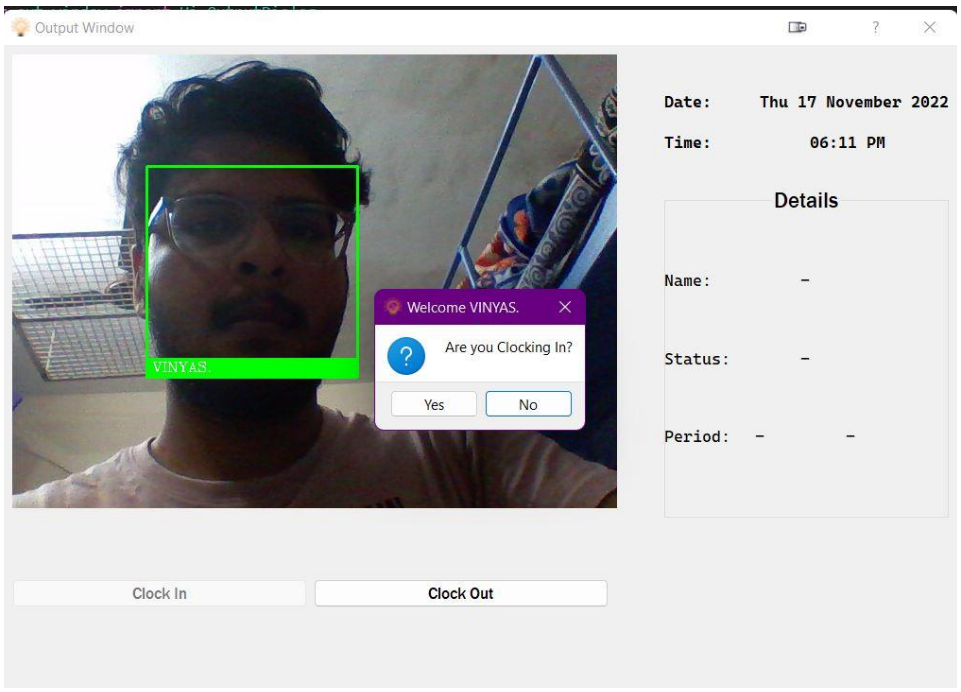
1. Attendance System

One of the main uses for which we initially began to work on the project is Smart Attendance System. As we already told how we would implement the face recognition system. We also mounted the camera on two servo motors that would rotate the camera left and right on the horizontal axis and move it up-down on the vertical axis. This would save time for the person as the camera would follow them and they would just need to keep walking. Their face would get recognized and their attendance or time of entry would get marked in a CSV file which we could then send to the server. Hence, we would be able to store this data for future use. In the pictures below we have shown our results which are written into a CSV file in python.

To mark attendance, we have made a face recognition time attendance app in which you have to show your face on camera two times. First time coming into the glass and the second time going from the class which also helps to calculate how much time person stays in class. This is the interface of the face recognition time attendance app.


Face Recognition Time Attendance App
?
X
Face Recognition Time Attendance App
Start

For example, Vinyas is coming into the class so he has to show his face in the camera and Clock in. The system will automatically detect the student's name.

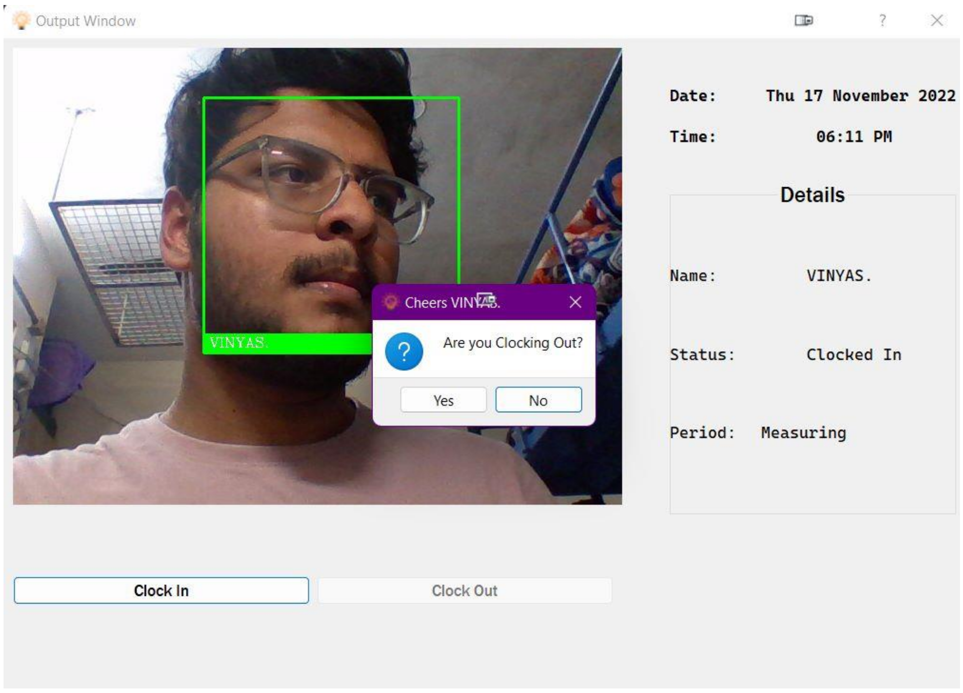


```
Date:      Thu 17 November 2022
Time:           06:11 PM

Details

Name:         -
Status:       -
Period:   -        -
```

After completing the class, Vinyas will show his face in the camera and Clock out. This helps us to calculate how much time a student stays in the class.

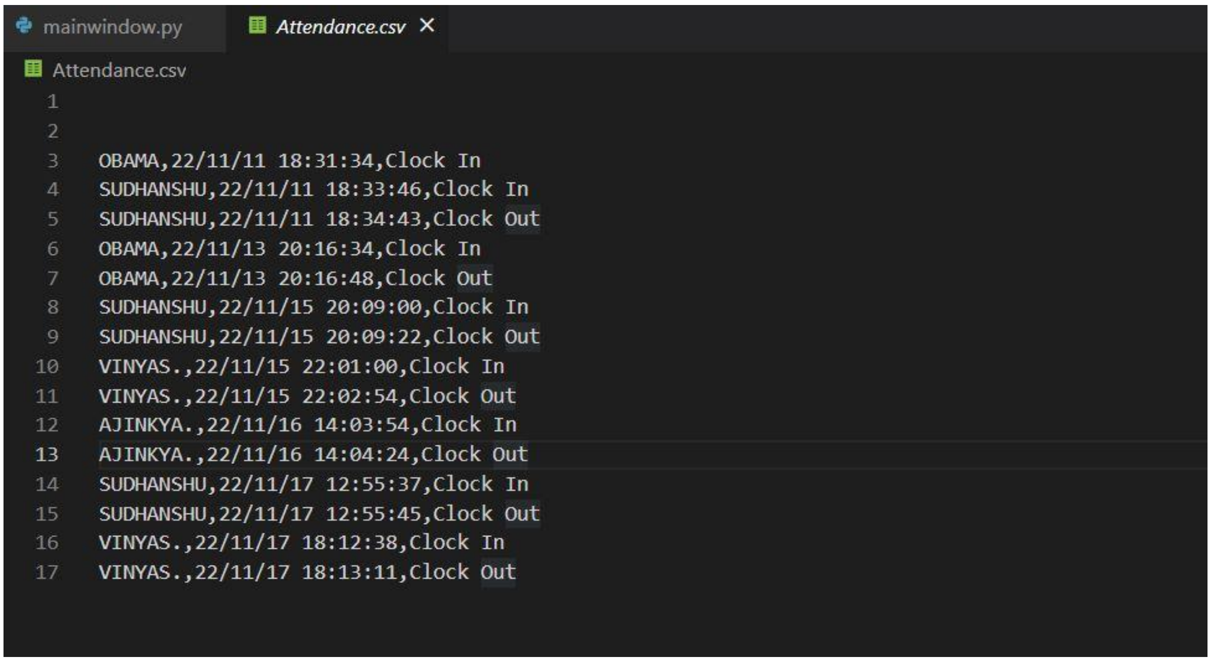


```
Date:      Thu 17 November 2022
Time:           06:11 PM

Details

Name:         VINYAS.
Status:       Clocked In
Period:   Measuring
```

After Clocking In and Clocking Out, the student's name will show in the attendance (.CSV) file with the timestamp that when he entered the class and when he left the class.
The below image shows that attendance is marked in .CSV file with time stamp



```


OBAMA,22/11/11 18:31:34,Clock In
SUDHANSHU,22/11/11 18:33:46,Clock In
SUDHANSHU,22/11/11 18:34:43,Clock Out
OBAMA,22/11/13 20:16:34,Clock In
OBAMA,22/11/13 20:16:48,Clock Out
SUDHANSHU,22/11/15 20:09:00,Clock In
SUDHANSHU,22/11/15 20:09:22,Clock Out
VINYAS.,22/11/15 22:01:00,Clock In
VINYAS.,22/11/15 22:02:54,Clock Out
AJINKYA.,22/11/16 14:03:54,Clock In
AJINKYA.,22/11/16 14:04:24,Clock Out
SUDHANSHU,22/11/17 12:55:37,Clock In
SUDHANSHU,22/11/17 12:55:45,Clock Out
VINYAS.,22/11/17 18:12:38,Clock In
VINYAS.,22/11/17 18:13:11,Clock Out
```

The data from the .CSV file can import into an excel sheet so that we will able to store the data for future use.

2.

We can also use this home safety device. During night time if anyone other than the family is picked up by the camera for extended periods of time it could ping the family either through a message or any kind of alarm system through the Arduino We can also use them to open the main door and garage of the house. Once all the members of the house are detected outside the house it could automatically turn the mains off for the house thus preventing loss of electricity.

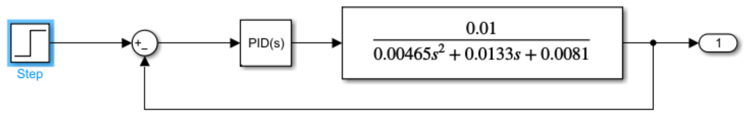
3.

Our system could also be used as a powerful security system to differentiate a particular person from a group of people and follow them to a certain extent. The system can then give or trigger any alarm or buzzer some kind of indication or led indication over the Arduino that it has located the said person. This would really help and make the task of police much easier to find or locate any missing person or a wanted criminal. Here is a sample picture I took with my roommates. We could also program the camera to take screenshots on the instances it thought it recognized someone similar to whom it was searching. These screenshots could be saved along with the time they were taken. Police can then go through the taken screenshots to get more information on the person and make an advancement in the case.

Possible Improvements

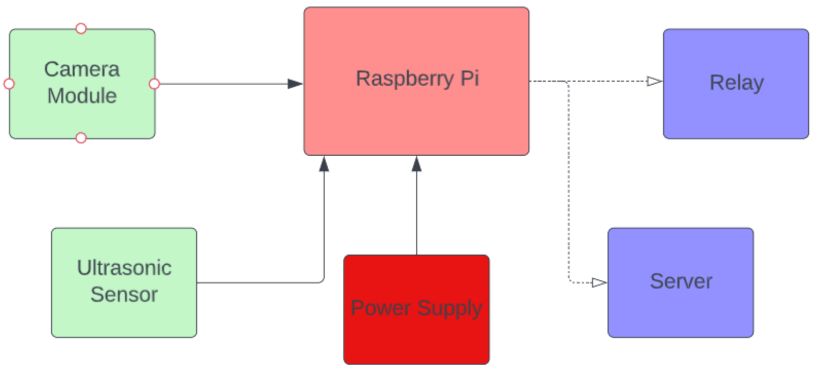
1.

We can use a PID controller here to increase sensitivity, response time, and stability. When the coordinates of the face are away from the center, the setpoint of the PID function will be set as zero and the input to the PID function is the difference between the center of the screen and the location of the green dot. Then the output of the PID function would be the servo angle by which the servo has to turn to bring the green dot toward the center of the screen. To get the most optimal values of Kp, Kt, and Kd we use Simulink to tune our system for a step input. We obtain the transfer function of the servo motor from it [9].



2.

We used a single camera setup for our project which is not very efficient in finding someone. We could actually extend this over multiple cameras. As shown we have used a laptop to power and give instructions to the servos and the Arduino. But it is impossible to link each camera with a laptop or a computer. So instead of using a laptop as the microcontroller, we could switch to a raspberry-pi to use it as a microcontroller instead. This would make each camera system compact, easily portable, and convenient to install even in small and cramped places. We are not using any complex calculations so a raspberry pi can easily handle the load. This raspberry pi would behave as a node and can be accessed by the main computer that will act as a server. Data received from these nodes would be remotely saved on cloud platforms like Azure or WCS. This decreases the risk of information leakage as well as makes the servers very fast and efficient to use. Also since the data are in the cloud it could be accessed by other computers which are not servers as well. A raspberry pi-4 could be used for the task.



3.

Another improvement we could have done is that we could have used an ultrasonic sensor for human presence detection. This would prevent the camera from unnecessary data processing. Because of the ultrasonic sensor, we would not have to keep the camera running. In short words, the camera would only process and run the code when the ultrasonic sensor detects any human presence. We can use the RS232 Serial MB1603 ultrasonic sensor for this build. [10]

We still have to work on the face recognition algorithm and process as it sometimes does not give very accurate results. One of the solutions to this is giving multiple different pictures of the same person but we don't always have this convenience with us.

Conclusion

In this paper, we have discussed the algorithm for facial detection and recognition. Using this algorithm we have proposed an idea of using a smart recognition and surveillance system based on facial recognition models which could have multiple uses in different ways such as an attendance system for schools or offices as a security device for home or a powerful searching or spying tool for the police and military. The camera is also capable of moving on servo motors in the direction of the person with the recognized face walking, both vertically and horizontally thus covering a large point of view. Improvements that could be made to the system to make it work even more efficiently have also been discussed.

References

[1]. Shah, Parin M., "Face Detection from Images Using Support Vector Machine" (2012).

[2]. Savio, M & Deepa, T & Bonasu, Anudeep & Anurag, Talluru. (2021). Image Processing For Face Recognition Using HAAR, HOG, and SVM Algorithms. Journal of Physics: Conference Series. 1964. 062023. 10.1088/1742-6596/1964/6/062023.

[3]. J. Vadlapati, S. Senthil Velan and E. Varghese, "Facial Recognition using the OpenCV Libraries of Python for the Pictures of Human Faces Wearing Face Masks during the COVID-19 Pandemic," *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2021, pp. 1-5, doi: 10.1109/ICCCNT51525.2021.9579712.

[4]. L. Lang and W. Gu, "Study of Face Detection Algorithm for Real-time Face Detection System," 2009 Second International Symposium on Electronic Commerce and Security, 2009, pp. 129-132, doi: 10.1109/ISECS.2009.237.

[5]. Nath, Raktim & Kakoty, Kaberi & Bora, Dibya & Welipitiya, Udari. (2021). Face Detection and Recognition Using Machine Learning. 43. 194-197.

[6] Asatryan, David. (2019). Gradient-based technique for image structural analysis and applications. Computer Optics. 43. 245-250. 10.18287/2412-6179-2019-43-2-245-250.

[7]. Hasan, M.K.; Ahsan, M.S.; Abdullah-Al-Mamun; Newaz, S.H.S.; Lee, G.M. Human Face Detection Techniques: A Comprehensive Review and Future Research

[8]. Kumar, Sudhakar & Das, Manas & Kushalkar, Rajesh & Venkat, Nirmala & Gourshete, Chandrashekhar & Moudgalya, Kannan. (2021). Microcontroller programming with Arduino and Python.

[9]. di Pasquo, Guido. (2021). SG90 Servo Characterization. 10.13140/RG.2.2.15715.89127.

[10]. Mutinda Mutava Gabriel, Kamweru Paul Kuria, 2020, Arduino Uno, Ultrasonic Sensor HC-SR04 Motion Detector with Display of Distance in the LCD, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 09, Issue 05 (May 2020).

APPENDIX:

Facial recognition using arduino and servo motor code

```
Facial recognition
import cv2
import numpy as np
import face_recognition
import os
from datetime import datetime
import serial,time
face_cascade= cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
def draw_border(img, pt1, pt2, color, thickness, r, d):
    x1,y1 = pt1
    x2,y2 = pt2
    # Top left
    cv2.line(img, (x1 + r, y1), (x1 + r + d, y1), color, thickness)
    cv2.line(img, (x1, y1 + r), (x1, y1 + r + d), color, thickness)
    cv2.ellipse(img, (x1 + r, y1 + r), (r, r), 180, 0, 90, color, thickness)
    # Top right
    cv2.line(img, (x2 - r, y1), (x2 - r - d, y1), color, thickness)
    cv2.line(img, (x2, y1 + r), (x2, y1 + r + d), color, thickness)
    cv2.ellipse(img, (x2 - r, y1 + r), (r, r), 270, 0, 90, color, thickness)
    # Bottom left
    cv2.line(img, (x1 + r, y2), (x1 + r + d, y2), color, thickness)
    cv2.line(img, (x1, y2 - r), (x1, y2 - r - d), color, thickness)
    cv2.ellipse(img, (x1 + r, y2 - r), (r, r), 90, 0, 90, color, thickness)
    # Bottom right
    cv2.line(img, (x2 - r, y2), (x2 - r - d, y2), color, thickness)
    cv2.line(img, (x2, y2 - r), (x2, y2 - r - d), color, thickness)
    cv2.ellipse(img, (x2 - r, y2 - r), (r, r), 0, 0, 90, color, thickness)
path = "imagesattendance"
images = []
classnames=[]
mylist=os.listdir(path)
print(mylist)
for cl in mylist:
    curimg=cv2.imread(f'{path}/{cl}')
    images.append(curimg)
    classnames.append(os.path.splitext(cl)[0])
print(classnames)
ArduinoSerial=serial.Serial('com5',9600,timeout=0.1)
time.sleep(1)
def findEncodings(images):
    encodelist=[]
    for img in images:
        img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
        encode=face_recognition.face_encodings(img)[0]
        encodelist.append(encode)
    return encodelist
def markAttendance(name):
    with open('atten.csv','r+') as f:
        myDataList=f.readlines()
        nameList=[]
        for line in myDataList:
            entry=line.split(',')
            nameList.append(entry[0])
        if name not in nameList:
            now=datetime.now()
            dtString=now.strftime('%H:%M:%S')
            f.writelines(f'\n{name},{dtString}')
encodelistknown=findEncodings(images)
print('Encoding Complete')
cap=cv2.VideoCapture(0)
while True:
        success,img=cap.read()
        imgS = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
        facesCurFrame=face_recognition.face_locations(imgS)
        encodesCurFrame = face_recognition.face_encodings(imgS,facesCurFrame)
        for encodeFace,faceLoc in zip(encodesCurFrame,facesCurFrame):
            matches = face_recognition.compare_faces(encodelistknown,encodeFace)
            faceDis= face_recognition.face_distance(encodelistknown,encodeFace)
            print(faceDis)
            matchIndex=np.argmin(faceDis)
            print(faceLoc)
            if  matches[matchIndex]:
                name=classnames[matchIndex].upper()
                print(name)
                y1,x2,y2,x1=faceLoc
                draw_border(img,(x1,y1),(x2,y2),(0,255,0),4,20,20)
                cv2.putText(img,name,(x1,y2+25),cv2.FONT_HERSHEY_SIMPLEX,1.1,(255,
0,255),2)
                markAttendance(name)
                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:
                #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'))
            else:
                y1,x2,y2,x1=faceLoc
                draw_border(img,(x1,y1),(x2,y2),(0,255,0),4,20,20)
        cv2.imshow('Webcam',img)
        cv2.waitKey(1)
```

Attendance System application code:

```
import sys
from PyQt5.uic import loadUi
from PyQt5 import QtWidgets
from PyQt5.QtCore import pyqtSlot
from PyQt5.QtWidgets import QApplication, QDialog
import resource
# from model import Model
from out_window import Ui_OutputDialog


class Ui_Dialog(QDialog):
    def __init__(self):
        super(Ui_Dialog, self).__init__()
        loadUi("mainwindow.ui", self)

        self.runButton.clicked.connect(self.runSlot)

        self._new_window = None
        self.Videocapture_ = None

    def refreshAll(self):

        self.Videocapture_ = "0"

    @pyqtSlot()
    def runSlot(self):

        print("Clicked Run")
        self.refreshAll()
        print(self.Videocapture_)
        ui.hide()  # hide the main window
        self.outputWindow_()  # Create and open new output window

    def outputWindow_(self):
        """
```

```python
        Created new window for visual output of the video in GUI
        """
        self._new_window = Ui_OutputDialog()
        self._new_window.show()
        self._new_window.startVideo(self.Videocapture_)
        print("Video Played")
if __name__ == "__main__":
    app = QApplication(sys.argv)
    ui = Ui_Dialog()
    ui.show()
    sys.exit(app.exec_())
```