TEXAS TECH UNIVERSITY
Whitacre College
of Engineering

CS 5331: Foundation of Cyber-Physical Systems

Project: CLASS ATTENDANCE SYSTEM USING FACIAL RECOGNITION

Jake Gonzalez

Ahmed Sunny

Afamefuna Umejiaku

CLASS ATTENDANCE SYSTEM USING FACIAL RECOGNITION

Introduction:

The aim of this project is to design and implement a classroom attendance recording system using a Raspberry Pi computer and camera module. The system utilizes facial recognition in order to recognize students as they walk into the classroom, and crate a record of when students arrive marking them as present. Then the arrivals are compared with the roster list of the class and if the student did not arrive, the student is marked absent.

A Cyber Physical System is typically designed as a network of interacting elements with physical input and output instead of as standalone devices and the notion is closely tied to concepts of robotics and sensor networks with intelligence mechanisms proper of computational intelligence leading the pathway. In this project we implemented such a networked system using a Raspberry Pi to take input from a camera and broadcast the video feed over IP. The video feed is used by the facial recognition program in order to determine which students arrive to the class. An output file is created by the program to denote which students were present for the class, and what time they arrived.

Motivation:

For this project we were asked to build a system using a microcontroller (Arduino/ Raspberry Pi) and sensors. The main focus of this project was to build a system through which we can replicate the mechanisms of embedded system. The project's goal was to make us realize the distinct engineering techniques which are inevitable to design and analyze any particular system.

We wanted to come up with a system that can deliver a cost-effective solution to a real world problem. There were many existing issues which we wanted to choose from such as Arduino based GSM home security, smart receptionist with a smart lock system, soccer robot, solar tracker system and so on. But we also had to consider time and cost constraint. We had to make several decisions prior deciding our project. Such as:

1. Does the project fulfill all the requirements?
2. Would we be able to deliver the project in a timely manner? To answer this question we had to analyze the following issues:
    a. Do we have required skills to complete a particular task?
    b. What are the expenses related to the physical equipment?
    c. How long it would take to receive the equipment?
3. Is our solution cost effective?

Based on all these fundamental criteria we decided to build a face recognition system based on raspberry pi. One of member of the group grew up in a South Asian country. There were approximately 120 students in a single classroom. The teacher would call the name of each student, which consumed 20 minutes of the 1-hour class session. In some cases the process was repetitive for each class. This is quite common throughout all the south Asian schools and colleges. We could imagine the amount of precious learning time lost by the students in that region.

Another group member grew up in Texas. He experienced a better system than students in South Asia. Each classroom had a RFID card reader, which kept track of the students attending the class. However, he also faced significant delays where he had to wait in long line of classmates prior entering the classroom. Compared to the to the previous manual roll call system, the RFID card scanner was better but certainly nowhere near to the optimum solution as it took the system approximately 10-15 seconds to process each student. With a class containing 75 to 100 students, this system also took away precious learning time.

Once we came across this idea we did some market research on similar face recognition systems available on the market. The price range for the cheapest camera starts from $90 and goes up to $1600. Where as we have only spent $42 in coming up with a consistent face recognition system using Raspberry Pi with the possibility of the price dropping to as low as $22 when we choose to implement the system with a smaller Raspberry Pi system.

Project Description:

For the project we utilized a Raspberry Pi model 3B+ with the Raspberry Pi camera module. This was used as the input to the system, as the video feed is captured, it is broadcasted over the local network. The video feed is connected by local IP to the facial recognition software, which runs on a nearby laptop connected to the same. The output of the software is a file containing the information of which students were present for the class and what time they arrived, along with the student that were absent for the class.

The software component of the project consisted of a graphical user interface (Figure 1), which allows lecturers to input the name of the course and the number of minutes to run the facial recognition part of the application to determine student arrival times.

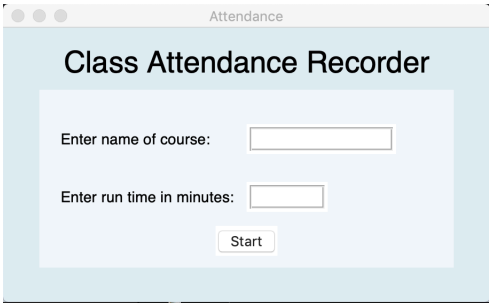

Figure 1: Graphical user interface to assist the lecturer

The program recognizes faces (Figure 2) that are stored as .jpg images in the same folder (Figure 3) of the python program. This makes adding or removing students to be recognized very quick and simple.

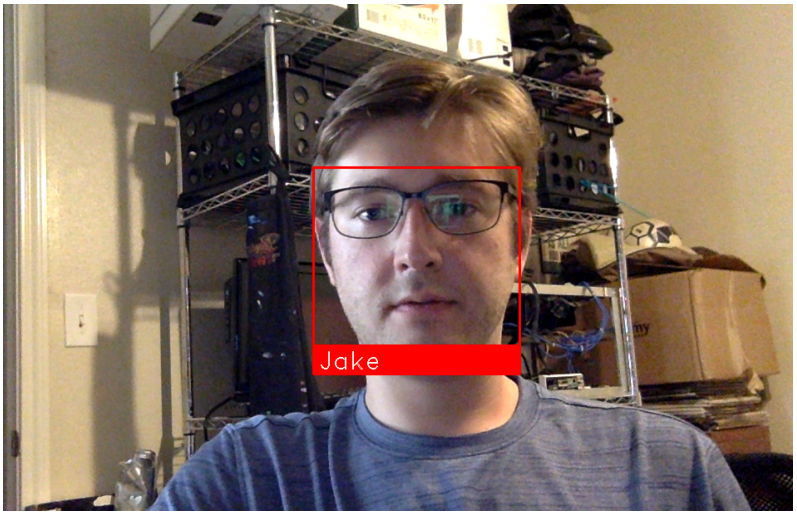

Figure 2: Real Time Face recognition using Raspberry Pi

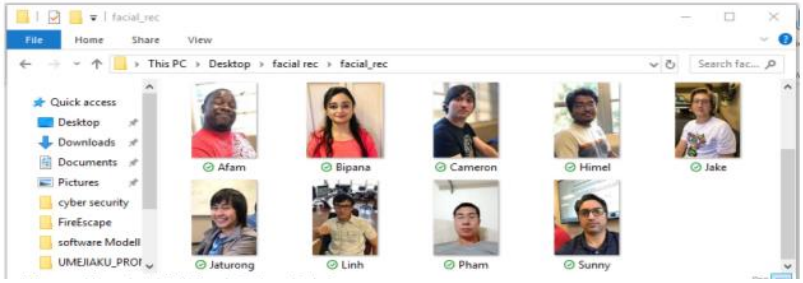

Figure 3: Folder Containing Student faces

After the time limit is reached the application outputs a unique file (Figure 4) with the course name and the date the program was ran. The output file states what time the students arrived for the class, and states which students did not arrive for the class.

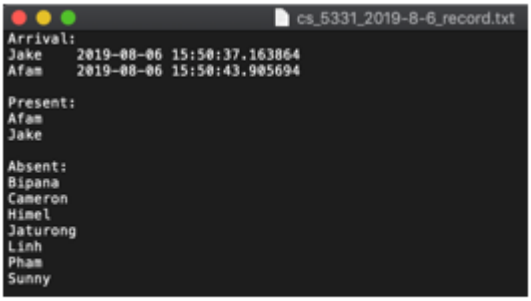

Figure 4: Output File indicative of timelines

Challenges:

Our group faces some challenges when trying to implement our project. Originally we wanted to implement the entire project on the Raspberry Pi so that a laptop would not be needed. However, when attempting to install all dependencies of the Open Computer Vision python library, and Facial Recognition python library we ran into trouble. After many failed attempts to install OpenCV, we finally had success after a 4 hour install process, but were very disappointed to realize that the version installed was not up to date with Python 3. To resolve this issue and still utilize the Raspberry Pi we decided to use the Pi as a video feed that we could connect to over the local network.

State Diagram for the Class Attendance System:

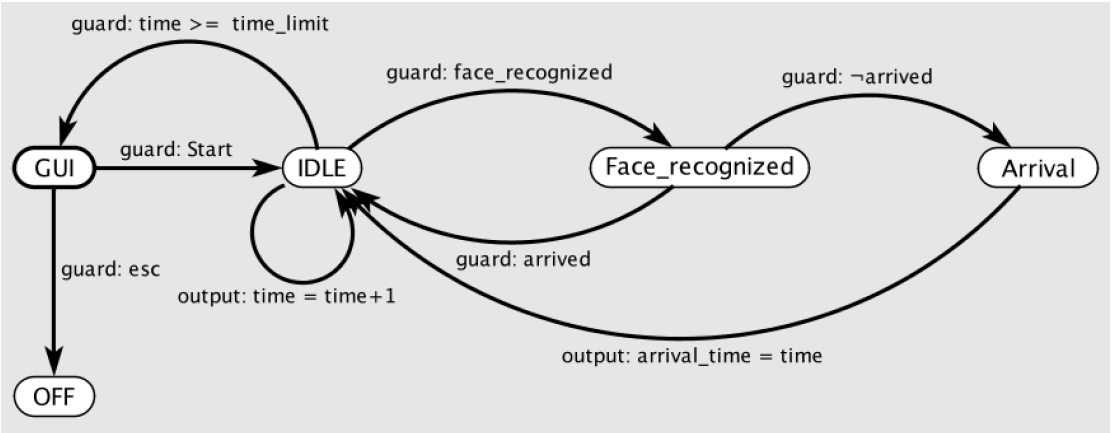

Figure 5: State Diagram for the Class Attendance System with Raspberry Pi

Future Work:

We are currently working on implementing our attendance system in C in order to compile the program to an executable capable of running across various platforms. Having examined the cost of our design compared to other systems available on the market, we are confident that our system is amongst the least expensive while still generating reliable results.

Conclusion:

We have explained the motivation behind creating the facial recognition system in order to fully utilize lecture time without sacrificing classroom attendance keeping. Using a system like the one we have implemented, institutions would be able to keep an accurate record of students' attendance while wasting no lecture time. In this report we have also compared the cost of hardware components we used in our implementation with the cost of other systems available on the market. Since our system can be implemented with as little cost as $22, it could potentially be very attractive to consumers. However, the main advantage of our system would be the reclaimed lecture time for institutions. By eliminating the timely process of manual role calling and automating the process, institutions can keep accurate record of student attendance without sacrificing precious lecture time.

References:

https://github.com/opencv/opencv
https://en.wikipedia.org/wiki/Cyber-physical_system
https://www.hackster.io/mjrobot/real-time-face-recognition-an-end-to-end-project-a10826

Appendix

```
import cv2
import face_recognition
import numpy as np
import datetime
from datetime import timedelta
from tkinter import *
import os

class Face_rec:
    def __init__(self):
        self.main_window = Tk()
        self.main_window.title("Attendance")
        self.canvas = Canvas(self.main_window, height=250, width=450)
        self.canvas.pack()
        self.background_frame = Frame(self.main_window, bg = "#d9e6f2")
        self.background_frame.place(relx=0, rely=0, relwidth=1, relheight=1)

        self.title_frame = Frame(self.background_frame, bg = "#d9e6f2")
        self.title_frame.place(relx=.5, rely=.125, relwidth=.85, relheight=.125, anchor = 'c')
        self.title_label = Label(self.title_frame, text = 'Class Attendance Recorder', justify='center',font=("Helvetica",28),
bg='#d9e6f2')
        self.title_label.place(relx=.5, rely=.5, anchor='c')

        self.main_frame = Frame(self.background_frame,  bg = "#ecf2f9", )
        self.main_frame.place(relx = .5, rely= .55, relwidth = .85, relheight = .65, anchor = 'c')

        self.time_label = Label(self.main_frame, text = 'Enter run time in minutes:', bd=0, justify='left',font=("Helvetica",14),
bg="#ecf2f9")
        self.time_label.place(relx = .05, rely = .6, anchor = 'w')

        self.time_entry = Entry(self.main_frame, bd=2, width=8,
justify='center',font=("Helvetica",14),relief='groove',highlightcolor="black")
        self.time_entry.place(relx = .5, rely = .6, anchor = 'w')

        self.course_label = Label(self.main_frame, text = 'Enter name of course:', bd = 0, justify =
'left',font=("Helvetica",14),bg="#ecf2f9")
        self.course_label.place(relx = .05, rely = .275, anchor = 'w')

        self.course_entry = Entry(self.main_frame, bd=2, width=16,
justify='center',font=("Helvetica",14),relief='groove',highlightcolor="black")
        self.course_entry.place(relx = .5, rely = .275, anchor = 'w')

        self.start_button = Button(self.main_frame, text = 'Start', command = self.face_rec)
        self.start_button.place(relx = .5, rely = .85, anchor = 'c')

        mainloop()



    def face_rec(self):
```

```
# set desired runtime in minutes
min_run = int(self.time_entry.get())
start_time = datetime.datetime.now()
end_time = start_time + timedelta(minutes = min_run)

# video capture 0 will use default camera
video_capture = cv2.VideoCapture(0)

# run ./start_stream in ~/bin/ on raspberry ip to start video stream and get valid hostname
#video_capture = cv2.VideoCapture('http://192.168.1.10:8080/?action=stream')

#video_capture = cv2.VideoCapture('http://129.118.162.82:8080/?action=stream')

# initialize variables
face_locations = []
face_encodings = []
face_names = []
present_names = []
absent_names = []
known_face_encodings = []
known_face_names = []

# load all .jpg images from directory to recognize face and display name
for file in os.listdir():
    if file.endswith(".jpg"):
        image = face_recognition.load_image_file(file)
        face_encoding = face_recognition.face_encodings(image)[0]
        known_face_encodings.append(face_encoding)
        known_face_names.append(file[:-4])

# used to process every second frame
process_this_frame = True

# get current date
date = datetime.datetime.now()

# create a string for date with year month and day
year = str(date.year)
month = str(date.month)
day = str(date.day)
date1 = (year +'-'+ month +'-'+ day)

# create a file unique to the day with course name
course_name = self.course_entry.get()
record_file = (course_name+'_'+date1+'_record.txt')
f = open("records/"+record_file, 'w+')
f.write('Arrival:\n')
f.close()

while True:
    # single frame is retrieved
    ret, frame = video_capture.read()

    # resize frame of video to .25 size for faster face recognition processing
    small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

    # convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)
    rgb_small_frame = small_frame[:, :, ::-1]

    # process every second frame
    if process_this_frame:
```

```
    # find all the faces and face encodings in the current frame of video
    face_locations = face_recognition.face_locations(rgb_small_frame)
    face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)

    face_names = []
    for face_encoding in face_encodings:
        # see if the face is a match for the known face(s)
        matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
        name = "Unknown"

    # if a match was found in known_face_encodings, use the known face with the smallest distance to the new face
        face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
        best_match_index = np.argmin(face_distances)
        if matches[best_match_index]:
            name = known_face_names[best_match_index]

            # check each line in report file for matched name
            with open("records/"+record_file) as file:
                datafile = file.readlines()
            found = False
            for line in datafile:
                if name in line:
                    found = True

            # if name not yet in file
            if not found:
                # append name to present_names[]
                present_names.append(name)

                # write name and time of arrival to file
                f = open("records/"+record_file, 'a')
                currentDT = datetime.datetime.now()
                f.write(name + '\t' + str(currentDT) + '\n')
                f.close()

        face_names.append(name)

# alternate frames to process (only needed when a high framerate is used)
#process_this_frame = not process_this_frame

# display results
for (top, right, bottom, left), name in zip(face_locations, face_names):
    # scale back up face locations since the frame we detected in was scaled to .25 size
    top *= 4
    right *= 4
    bottom *= 4
    left *= 4

    # draw a box around the face
    cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

    # draw a label with a name below the face
    cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)
    font = cv2.FONT_HERSHEY_DUPLEX
    cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)

# display the resulting image
cv2.imshow('Video', frame)

# hit 'q' on the keyboard to quit
if cv2.waitKey(1) & 0xFF == ord('q'):
```

```
            break

        curr_time = datetime.datetime.now()
        elap_time = curr_time - start_time

        print(elap_time)
        if curr_time >= end_time:
            break

    # open record file
    f = open("records/"+record_file, 'a')

    # sort and write present names to file
    present_names.sort()
    f.write('\nPresent:\n')
    for name in present_names:
        f.write(name+'\n')

    # find absend names
    for name1 in known_face_names:
        found = False
        for name2 in present_names:
            if name1 == name2:
                found = True
        if not found:
            absent_names.append(name1)

    # sort and write absent names to file
    absent_names.sort()
    f.write('\nAbsent:\n')
    for name in absent_names:
        f.write(name+'\n')

    f.close()

    # release handle camera
    video_capture.release()
    cv2.destroyAllWindows()


Face_rec()
```