INDIVIDUAL LAB REPORT 5
Progress Review
MRSD Project

Name: Keerthana P G
Andrew ID: kgopalak

Team D
Members:
Ritwik Das
Luka Eerens
Keerthana P G

22nd November 2017

Teamwork

Ritwik trained the network, hypothesised on why our loss was not converging and corrected them. One problem was low accuracy due to pre-trained Resnet which stuck the weights at a very low minima that it was hard to adapt it to our task of emotion recognition. This was solved by unfreezing and retraining the Resnet. Luka designed and printed the 3D part on which the camera is mounted. He also optimised the preprocessing routine and identified and fixed bugs in it. I built the face tracker using stepper motor, and tested it along with Ritwik. I also wrote a working code to implement log mel filter banks for audio processing. We also spent some time working on Systems Engineering & Management Assignment 3.

Individual Responsibility

I was responsible for making the face tracker mechanism work. I also implemented log mel filter banks for audio processing.

Log Mel Filter Banks

Why we are doing this:

Log mel filter banks[1] is a method to extract the audio spectrum for different frequencies and estimate energy incidence at each frequency. This is a form of audio processing that changes emphasis and extracts key features from human voice.

Procedure:

1. Pre-emphasis: Pre-emphasis is applied to amplify high frequencies with an operation $y(t)=x(t)-\alpha x(t-1)$
2. Framing: The signal is divided into overlapping small frames where frequency is assumed to be almost invariant. The frames overlap since fourier transform doesn't work well at edges.
3. Applied window function to curve the edges, applied fourier transform and calculated power spectrum.
4. Applied log mel filters to extract frequency bands: Mel filters are frequencies arranged in such a way that filters are thinner in lower frequency bands(near hearing range) and wider at higher frequencies(where we don't care about changes as much). Once we know the energies at different frequency banks, we take their logarithmic values.

Stage of Implementation

We found an implementation at [2]. We cleaned it, removed unused modules and unnecessary transformations and made it conducive to be integrated to our code. Currently, the filter banks extracts energy spectrum over three frequency bands. They've not been put into action with the main code as we are still figuring out the issues with accuracy.

Face tracker mechanism design

Procedure:

1. Established 2-way serial communication between Arduino and python on PC.
2. Set up over-current protection
3. Set up the circuit and configured the driver for stepper motor
4. Assembled the hardware and mounting assembly
5. Integrated the stepper motor control and face detection algorithm
6. Tested and calibrated parameters

Challenges:

1. The serial communication is 2 times or more slower than face detection, and to detect face more than 75% of the time(as mandated by requirements), we had to do multi core processing for face detection and arduino communication. These parallel processes share memory with a queue where the face tracker flushes the queue and inputs the latest value to it, and the arduino reads from the queue. This way, both processes don't have to wait for each other and works with the latest available data at their fastest possible speeds.
2. Assembling the hardware was a challenge. Off axis moments would highly set off the mechanism. Contact friction at the mounting assembly provided a resistance to the motor torque. Due to high vibration, parts having small clearances produced a lot of noise, we simply glued them together to fix this. At one instant heavy weights were required to establish contact and motion at the mounting assembly. We had to perform several iterations to get the hardware to work correctly.
3. Faces aren’t detected at low light and at large distance from the camera(>1.75m). We realised that the FVE requirements don’t specify lighting conditions. Also, since the person designing the bot and the person writing the FVE didn't communicate properly, the FVE requirements were set at unrealistic levels. For example, the test states that the human moves at a distance of 2m from the camera. It is not realistic that the user speaks to the small bot at this large distance, where the resolution is also low. Additionally the speed of movement mentioned is too high. We have to update FVE requirements.

Additional Responsibility

I assisted Ritwik in acquiring more facial expression datasets by sending requests to researchers.

Future Plans

Individual:

1. Update FVE requirements.
2. Complete testing of face tracking mechanism.
3. Package the mechanism in a compact and robust way.

4. Prepare the PCB for FVE
5. Assist Ritwik in training the network
6. Contribute to the UI for outputting emotion chart.

Team:

1. We hope to have met the accuracy requirement for the network
2. We hope to have met the requirements for the face tracker.

References

1. http://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html
2. https://github.com/jameslyons/python_speech_features

Appendix 1: Codes

gammacorrectedfacetrack.py

```
import serial
import syslog
import time
import cv2
import dlib
import serial
from threading import Thread
from queue import Queue
import numpy as np

detector = dlib.get_frontal_face_detector()
q = Queue()

#The following line is for serial over GPIO
port = '/dev/ttyACM0' # note I'm using Mac OS-X

ard = serial.Serial(port,9600,timeout=5)
time.sleep(2) # wait for Arduino

val1=0

def gamma_correction(img, correction):
    img = img/255.0
    img = cv2.pow(img, correction)
    return np.uint8(img*255)

def sendtoard(val1):
    while (1):
    # Serial write section

        ard.flush()
        val2 = str(q.get())
        print ("Python value sent: ")
        print (val2)
        val2=val2.encode('utf-8')
        ard.write(val2)
        time.sleep(1) # I shortened this to match the new value in your Arduino code

        # Serial read section
        msg = ard.read(ard.inWaiting()) # read all characters in buffer
        print ("Message from arduino: ")
        print (msg)

    else:
        print ("Exiting")
```

```
    exit()

def show_webcam(mirror=False):
    cam = cv2.VideoCapture(1)
    while True:
        ret_val, img = cam.read()
        if mirror:
            img = cv2.flip(img, 1)
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            gray = gamma_correction(gray,0.25)
            try:
                dets = detector(gray)

                #print("Number of faces detected: {}".format(len(dets)))

                for i, d in enumerate(dets):
                    #print("Detection {}: Left: {} Top: {} Right: {} Bottom: {}".format(
                      # i, d.left(), d.top(), d.right(), d.bottom()))
                    cv2.rectangle(gray, (d.left(), d.top()), ( d.right(), d.bottom()), (255,0,0), 2)
                    centroid=((d.left()+d.right())/2, (d.top()+ d.bottom())/2)
                    val1=centroid[0]-320
                    with q.mutex:
                         q.queue.clear()
                    q.put(val1)
                    print(val1)

                    #the frame seems to have 620 horizontal pixels
                    #the val tells you how many pixels from the centre he centroid of face is ocated.
                    #-ve means image centroid is at the right f face centroid
                    #+ve means viceversa
                    #magnitude of val indicates how shifted it is

                cv2.imshow('my webcam', gray)

                if cv2.waitKey(1) == 27:
                    break  # esc to quit
            except:
                continue
    cv2.destroyAllWindows()

if __name__ == "__main__":
    t1 = Thread(target=sendtoard, args=(val1,))
    t2 = Thread(target = show_webcam, args=(True,))
    t1.setDaemon(True)
    t2.setDaemon(True)
    t1.start()
    t2.start()
    while True:
```

```
        pass
```

Steppercontrol.py

```
// Serial test script
int STEP = 8;
int DIR = 7;
void setup()
{

  Serial.begin(9600);  // initialize serial communications at 9600 bps
  pinMode(STEP,OUTPUT);
  pinMode(DIR,OUTPUT);

}

void StepperMotor(float speed, float angle)
{
  angle=angle/340*45*0.7;
  angle=int(angle/1.8);
  for(int i = 0; i < angle; i++)
  {
    digitalWrite(STEP, HIGH);
    delayMicroseconds(speed);
    digitalWrite(STEP, LOW);
    delayMicroseconds(speed);
  }
  //delay(1000);
}


void loop()
{
  while(!Serial.available()) {}
  // serial read section
  String readString;
  while (Serial.available())
  {
    delay(30);
    if (Serial.available() >0)
    {
      char c = Serial.read();  //gets one byte from serial buffer
      readString += c; //makes the string readString
    }
  }

  if (readString.length() >0)
  {
```

```
  //Serial.print("Arduino received: ");
  //Serial.println(readString); //see what was received
 }

 //delay(500);

 //; serial write section
 float val=readString.toFloat();

 //char ard_sends = '1';
 //Serial.print("Arduino sends: ");
 //Serial.println(val);
 //Serial.print("\n");
 Serial.flush();

 float Speed=700;//give a value between 400 to 1000
 if(val>80 || val<-80){
 if(val<0){
   digitalWrite(DIR, LOW);
 }
 else{
  digitalWrite(DIR, HIGH);
 }
 StepperMotor(Speed, abs(val));
 }
}
```