SMART PUP

Submitted by:

Abhirup Chakravarty – 17BCE7055

Rahul Ramkumar – 17BCN7026

Timir Patel - 17BCE7021

Amal Rajan - 17BCE7017

ECS180400

***Under the guidance of* Dr. Sudha Ellison Mathe**

VIT is a premier institute in the state of Andhra Pradesh, in the capital city – Amaravati.

ABSTRACT

The modern times show a surge in the growing trends of Machine Learning, artificial cognisance and bitterly enough, loneliness. It seems that the more connected the human society is growing to become, the more distant, distraught from the basic forms of human pleasures it is becoming. In several cases of psychological approach, one of the prime methodology is of using a therapy dog [1] that provides accompaniment to the owner. Given that most people are apathetic to having a pet, ensued by the physical responsibilities that come along-with, we are seeking to build a basic emotion recognisant robot based on that of a dog's behaviour. This can be utilised in many a way, but presently our aim is to make it recognise emotions, detect facial patterns (expressions), detect movement in its FOV (field of view) and react to the situation via a display, sound implemented by speakers, wireless or otherwise, and locomotion.

In this report, there is presented the results of a project targeted to build a robotic system aiming at emotional recognisance via analysis of facial recognition, as well as textual analysis of speech provided to it as an input via the user; as also, the manifestation of suitable reactions pertaining to such emotions via the channels of audio, video (still) and locomotion. The driving ideal is the construction of a robotic system aptly capable of providing life-like interactions based upon audio-visual inputs.

INDEX

INTRODUCTION

In this section, let us get comfortably familiar regarding all the jargon that are to be iteratively found throughout this project.

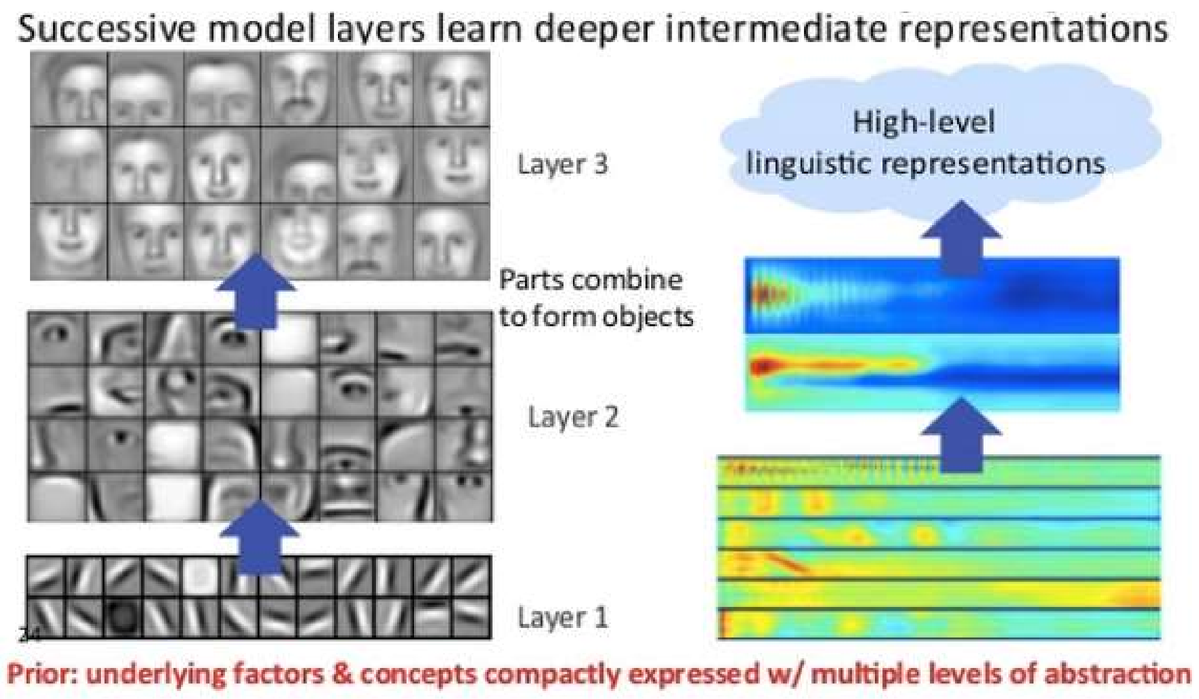
Emotional Recognisance: Emotion recognition is the process of identifying human emotion, most typically from facial expressions as well as from verbal expressions. This is both something that humans do automatically but computational methodologies have also been developed. [2]

Open CV: The Open Source Computer Vision Library is released under a BSD license and hence it’s free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. 3 was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. [3]

Deep Neural Network (DNN): Deep-learning networks are distinguished from the more commonplace single-hidden-layer neural networks by their depth; that is, the number of node layers through which data must pass in a multistep process of pattern recognition.

Earlier versions of neural networks such as the first perceptrons were shallow, composed of one input and one output layer, and at most one hidden layer in between. More than three layers (including input and output) qualifies as “deep” learning. So *deep* is not just a buzzword to make algorithms seem like they read Sartre and listen to bands you haven’t heard of yet. It is a strictly defined term that means more than one hidden layer.

In deep-learning networks, each layer of nodes trains on a distinct set of features based on the previous layer’s output. The further you advance into the neural net, the more complex the features your nodes can recognize, since they aggregate and recombine features from the previous layer. [4]


Successive model layers learn deeper intermediate representations
Layer 3
High-level
linguistic representations
Parts combine
to form objects
Layer 2
Layer 1
24
Prior: underlying factors & concepts compactly expressed w/ multiple levels of abstraction

And thus, given that we have established the basis of our project, let us continue further to elucidate clearly, the workings of the model.

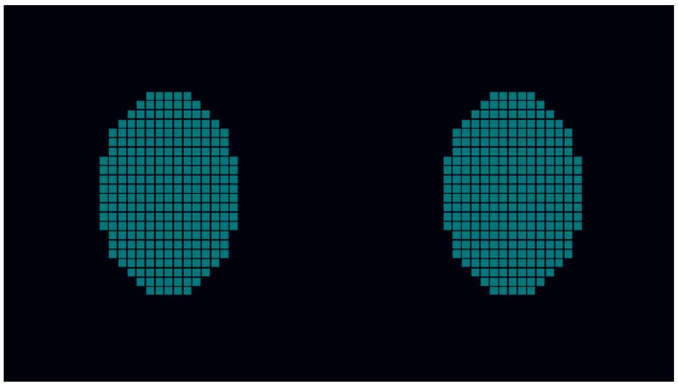
METHODOLOGY

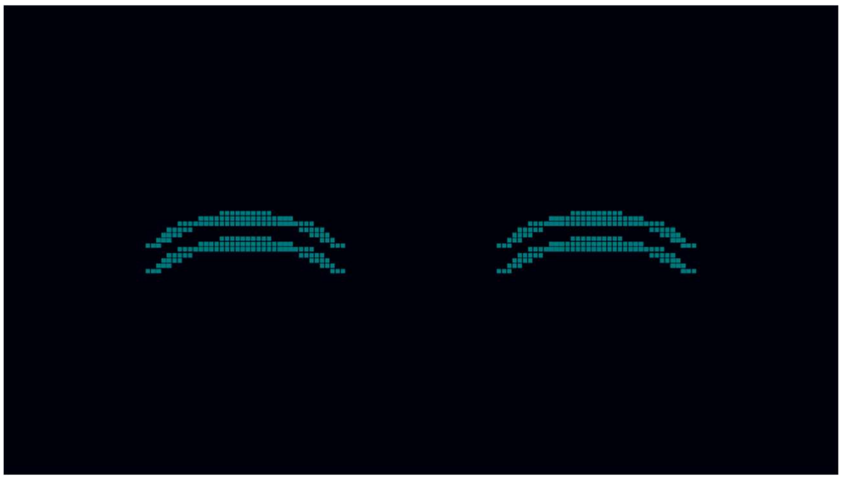
In recent years, emotion detection in text has become more popular due to its vast potential applications in marketing, political science, psychology, human-computer interaction, artificial intelligence, etc. Access to a huge amount of textual data, especially opinionated and self-expression text also played a special role to bring attention to this field. In this paper, we review the work that has been done in identifying emotion expressions in text and argue that although many techniques, methodologies, and models have been created to detect emotion in text, there are various reasons that make these methods insufficient. Although, there is an essential need to improve the design and architecture of current systems, factors such as the complexity of human emotions, and the use of implicit and metaphorical language in expressing it, lead us to think that just re-purposing standard methodologies will not be enough to capture these complexities, and it is important to pay attention to the linguistic intricacies of emotion expression. [5]

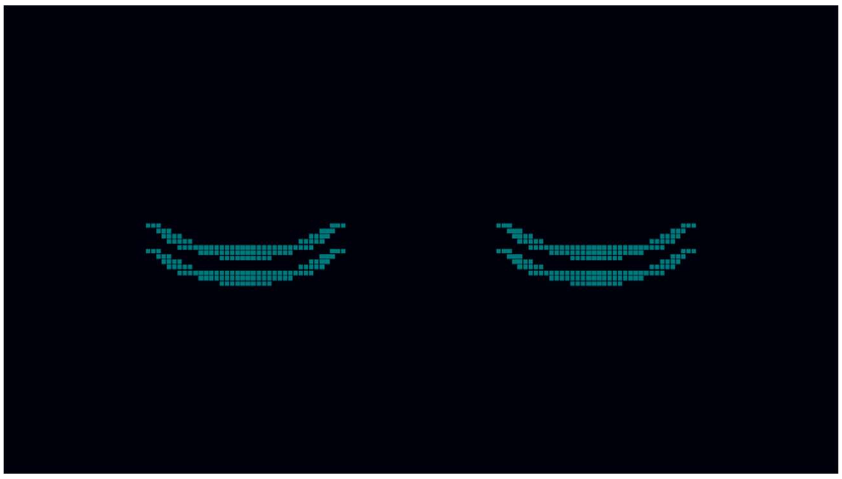
Facial expressions play an important role in recognition of emotions and are used in the process of non-verbal communication, as well as to identify people. They are very important in daily emotional communication, just next to the tone of voice. They are also an indicator of feelings, allowing a man to express an emotional state. People, can immediately recognize an emotional state of a person. As a consequence, information on the facial expressions are often used in automatic systems of emotion recognition. The aim of the research, presented in this article, is to recognize seven basic emotional states: neutral, joy, surprise, anger, sadness, fear and disgust based on facial expressions. [6]

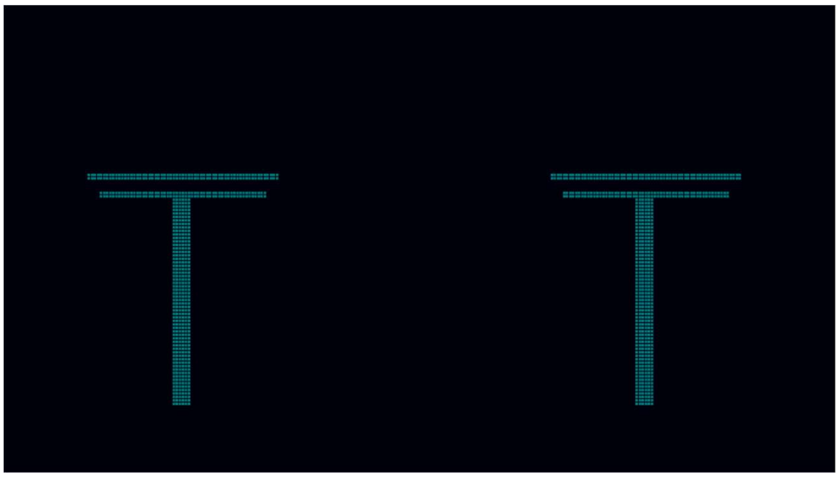
Our plan is to implement speech-based recognition in which we use Google APIs [7] for speech detection and conversion to a string of literal characters and use a trained classifier model using Convolutional Neural Networks (CVN) [8] to analyse the text and based on relative confidence scores, provide the highest scoring emotion as the underlying emotion.

The emotion is then sent forward to the other modules using a Raspberry Pi 3.0 (B) model, which is then read by the display, locomotor drivers, and speakers.


Happy


Excited


Content


Sad

The above are a few examples fed to the display module, associated with the emotions.

The locomotor drivers move about in a way mimicking the movements of a dog. Example, suppose the module is two-wheeler in nature, with a support wheel in the front. If it exhibits the excited emotion, say the left motor spins at 60% Duty Cycle and the right at 85%, providing a loop-like motion, similar to how dogs run around in circles when they are very excited.

For the audio, we can simply upload audio clips to the Pi to run a particular file associated to each emotion.

As an alternative, we also implement facial emotion recognition algorithms, which are implementations of Open CV libraries. [3]

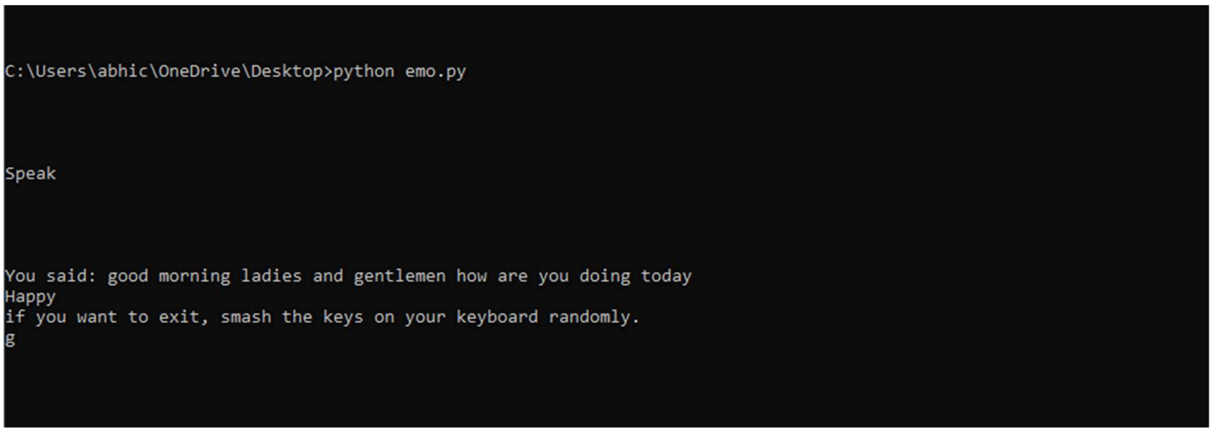
To make it further doglike, we also include a module which makes the bot, whilst standing still, to bark if it detects an motion in the background, similar to how dogs with react to any new motion in their FOV.

RESULTS

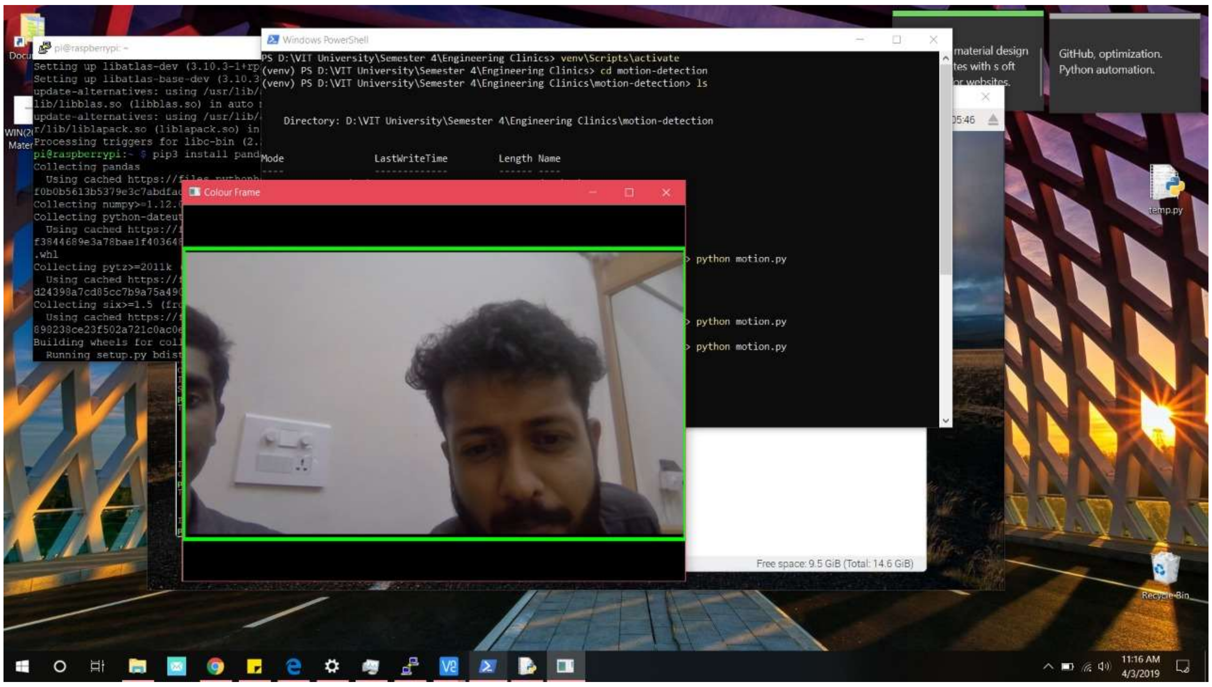
To establish an accurate dataset, we might say that we have been successful in getting the proper results more or less, with an error rate of 17.49% (approx.). The peripheral modules work perfectly as long as the data connectivity is properly available. For, since it implements connectivity of the apps over a WLAN, whilst accessing Google APIs, if the connection is poor, the bot suffers from a bottleneck.

The moving blob detection algorithm implemented using Open CV for detecting motion in the bot's field of view (FOV), is accurate if the area is properly let, as if that's not the case, the moving shades are also classified as separate moving blobs and thus results in the dog barking.

The locomotion is smooth and without any issues that we have documented in our extensive testing.

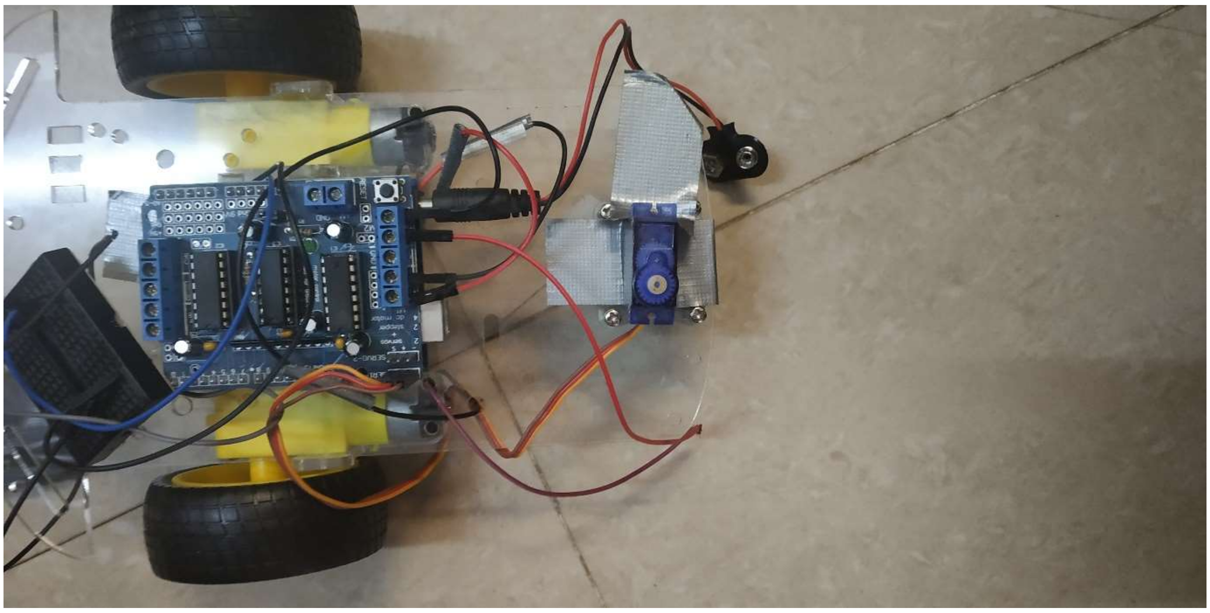


```
C:\Users\abhic\OneDrive\Desktop>python emo.py


Speak


You said: good morning ladies and gentlemen how are you doing today
Happy
if you want to exit, smash the keys on your keyboard randomly.
g
```



```
Setting up libatlas-dev (3.10.3-1+rp
Setting up libatlas-base-dev (3.10.3
update-alternatives: using /usr/lib/
lib/libblas.so (libblas.so) in auto
update-alternatives: using /usr/lib/
r/lib/liblapack.so (liblapack.so) in
Processing triggers for libc-bin (2.
pi@raspberrypi:~ $ pip3 install pand
Collecting pandas
  Using cached https://files.pythonh
f0b0b5613b5379e3c7abdfa
Collecting numpy>=1.12.
Collecting python-dateut
  Using cached https://
f3844689e3a78bae1f40364
.whl
Collecting pytz>=2011k
  Using cached https://
d24398a7cd85cc7b9a75a49
Collecting six>=1.5 (fr
  Using cached https://
898238ce23f502a721c0ac0
Building wheels for col
  Running setup.py bdist
```

```
PS D:\VIT University\Semester 4\Engineering Clinics> venv\Scripts\activate
(venv) PS D:\VIT University\Semester 4\Engineering Clinics> cd motion-detection
(venv) PS D:\VIT University\Semester 4\Engineering Clinics\motion-detection> ls


    Directory: D:\VIT University\Semester 4\Engineering Clinics\motion-detection


Mode                LastWriteTime         Length Name
----                -------------         ------ ----

> python motion.py


> python motion.py

> python motion.py
```



Video link:

https://drive.google.com/drive/folders/1Nn72UB6tpeeQrVmiQFSwTJw3Xw97bACG?usp=sharing

CONCLUSION

After assessing all the factors, we may say that the project was a success in building a smart emotion recognising module that, although recognising data connectivity, recognises the underlying emotions of a human's countenance and speech well enough.

FUTURESCOPE

1. With further development of technology, especially the storage space, we may include all the libraries offline and in store, thus removing the dependency of data connection to the internet.

2. The locomotion can be made more effective using chain and track instead of wheels. And if the end-goal is to make it more dog-like, not just efficient, using robotic limbs may be suggested.

3. More work can be put into the moving blob detection to make it further accurate and more dependent.

REFERENCES

[1] Handbook on Animal-Assisted Therapy, 5th Edition - Academic Press, 1st July 2019

[2] https://en.wikipedia.org/wiki/Emotion_recognition

[3] https://3.org/

[4] https://skymind.ai/wiki/neural-network#concept

[5] :Armin Seyeditabari, Narges Tabari, Wlodek Zadrozn - Emotion Detection in 5: a Review, arXiv:1806.00674 [cs.CL]

[6] Pawel Tarnowski, Marcin Kolodziej, Andrezej Majkowski, Remigiusz J. Rak - Emotion recognition using facial expressions - International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland

[7] https://cloud.google.com/speech-to-5/docs/

[8] https://github.com/ParallelDots/ParallelDots-Python-API

APPENDIX: CODE

Blob Movement Detection:

```
# Import OpenCV for processing of images

# Import Time for window time logging

# Import DateTime for saving time stamp of motion detection events

# Import Pandas for dataframe and CSV creation
```

```
import cv2
import time
import pandas
from datetime import datetime
from threading import Thread
from playsound import playsound



def start_audio():
	playsound("dog_bark.mp3")


def delay():
	global AUDIO_ON


	time.sleep(4)
	AUDIO_ON = False


# reference background frame against which to compare the presence of object/motion
first_frame = None
AUDIO_ON = False


# Capture video feed from webcam (0), use video filename here for pre-recorded video
video = cv2.VideoCapture(0)



# stores the presence/absence of object in the present frame. -1 for absent and 1 for present
statusList = [-1, -1]
times = []  # stores timestamps of the entry and exit of object
# Pandas dataframe for exporting timestamps to CSV file
df = pandas.DataFrame(columns=["Start", "End"])
```

```
# the following loop continuously captures and displays the video feed until user prompts an exit by pressing Q

while True:

    # the read function gives two outputs. The check is a boolean function that returns if the video is being read

    check, frame = video.read()

    status = -1  # initialise status variable. This stores the presence/absence of object in the current frame

    # Grayscale conversion of the frame

    grayImg = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Gaussian blur to smoothen the image and remove noise.

    # The touple is the Kernel size and the 0 is the Std Deviation of the blur function

    grayImg = cv2.GaussianBlur(grayImg, (21, 21), 0)


    if first_frame is None:

        first_frame = grayImg  # collect the reference frame as the first video feed frame

        continue


    # calculates the absolute difference between current frame and reference frame

    deltaFrame = cv2.absdiff(first_frame, grayImg)


    # convert image from grayscale to binary. This increases the demarcation between object and background by using a threshold function that

    # converts everything above threshold to white

    threshFrame = cv2.threshold(deltaFrame, 30, 255, cv2.THRESH_BINARY)[1]


    # dilating the threshold removes the sharp edges at the object/background boundary and makes it smooth.

    # More the iterations, smoother the image. Too smooth and you lose valuable data

    threshFrame = cv2.dilate(threshFrame, None, iterations=2)


    # Contour Function

    # The contour function helps identify the closed object areas within the background.

    # After thresholding, the frame has closed shapes of the objects against the background
    # The contour function identifies and creates a list (cnts) of all these contours in the frame
    # The RETR_EXTERNAL ensures that you only get the outermost contour details and all child contours inside it are ignored
    # The CHAIN_APPROX_SIMPLE is the approximation method used for locating the contours. The simple one is used here for our trivial purpose
    # Simple approximation removes all the redundant points in the description of the contour line
    cnts, _ = cv2.findContours(
        threshFrame.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)


    for contour in cnts:
        if cv2.contourArea(contour) < 10000:
            # excluding too small contours. Set 10000 (100x100 pixels) for objects close to camera
            continue
        status = 1
        if AUDIO_ON == False:
            Thread(target=start_audio).start()
            AUDIO_ON = True
            Thread(target=delay).start()
        # obtain the corresponding bounding rectangle of our detected contour
        (x, y, w, h) = cv2.boundingRect(contour)

        # superimpose a rectangle on the identified contour in our original colour image
        # (x,y) is the top left corner, (x+w, y+h) is the bottom right corner
        # (0,255,0) is colour green and 3 is the thickness of the rectangle edges
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 3)

        # do the above for all contours greater than our set size

    # add the present status to our list
    statusList.append(status)

    # Detecting the entry and exit of objects
```

```
        # Every entry/exit is identified by a sign change of the last two elements in our list, hence
product is -1
        if (statusList[-1]*statusList[-2]) == -1:
                times.append(datetime.now())


        # unitTesting
        #cv2.imshow("Capturing", grayImg)
        #cv2.imshow("DeltaFrame", deltaFrame)
        #cv2.imshow("Threshold Frame", threshFrame)
        # print(status)


        # displays the continous feed with the green frame for any foreign object in frame
        cv2.imshow("Colour Frame", frame)


        # picks up the key press Q and exits when pressed
        key = cv2.waitKey(1)
        if key == ord('q'):
                # if foreign object is in frame at the time of exiting, it stores the timestamp
                if status == 1:
                        times.append(datetime.now())
                break



# print(statusList)
# print(times)


# take every 2 timestamps in the list and store them as startTime and endTime in the Pandas dataframe
for i in range(0, len(times), 2):
        df = df.append({"Start": times[i], "End": times[i+1]}, ignore_index=True)


# Export to csv
df.to_csv("Times.csv")
```

```
# Closes all windows
cv2.destroyAllWindows()

# Releases video file/webcam
video.release()
```

Emotion Detection:

```
//char receivedChar;
//boolean newData = false;
//
//void setup() {
// Serial.begin(9600);
// Serial.println("<Arduino is ready>");
//}
//
//void loop() {
// recvOneChar();
// showNewData();
//}
//
//void recvOneChar() {
// if (Serial.available() > 0) {
// receivedChar = Serial.read();
// newData = true;
// }
//}
//
//void showNewData() {
// if (newData == true) {
// Serial.print("This just in ... ");
// Serial.println(receivedChar);
// newData = false;
// }
//}

#include <AFMotor.h>
#include<Servo.h>
#include<SoftwareSerial.h>

#define TRIG_PIN 6
#define ECHO_PIN 7

AF_DCMotor leftMotor(1, MOTOR12_8KHZ);
AF_DCMotor rightMotor(2, MOTOR12_8KHZ);
Servo neckControllerServoMotor;

int curDist = 0;
int dist1 = 0;
int dist2 = 0;

boolean newData = false;
char receivedChar;

void setup() {
  neckControllerServoMotor.attach(10);
  neckControllerServoMotor.write(90);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  leftMotor.setSpeed(150);
  rightMotor.setSpeed(150);
  neckControllerServoMotor.write(90);
  // moveforward();
  Serial.begin(9600);
}

void recvOneChar() {
 if (Serial.available() > 0) {
 receivedChar = Serial.read();
 newData = true;
 }
}

void movePup() {
 if (newData == true) {
   Serial.print("This just in ... ");
   Serial.println(receivedChar);

   if(receivedChar == 'a') {


   }
   if(receivedChar == 'b') {
    angry();
   }
   if(receivedChar == 'c') {
    sad();
   }
   if(receivedChar == 'd') {


   }
   if(receivedChar == 'e') {
   }
   if(receivedChar == 'f') {


   }
   if(receivedChar == 'g') {
    happy();
   }
   recievedChar = 'h';
   newData = false;
 }
}

void loop()
{
  Serial.println("code starts");
  recvOneChar();
  movePup();
}
void moveforward()
{
    Serial.println("moving forward");
    leftMotor.run(FORWARD);
    rightMotor.run(FORWARD);
    delay(1000);
}
void checkObstacle()
{
  int curDist1;
  curDist = readping();
  if (curDist < 20)
  {
    Serial.println("obstacle sensed");
    moveStop();
    neckControllerServoMotor.write(180);            //180 degrees meaning sensor is on left
    dist1 = readping();
    delay(2000);
    neckControllerServoMotor.write(0);

    dist2 = readping(); //0 degrees meaning sensor is on right
    delay(2000);
    if (dist1 < dist2)
    {
     leftTurn();
    }
    if (dist2 < dist1)
    {
     rightTurn();
    }
    if (dist1 == dist2)
    {
     leftTurn();
    }
    neckControllerServoMotor.write(90);
    moveforward();
  }
}
void rightTurn()
{
  Serial.println("right");
  leftMotor.run(FORWARD);
  rightMotor.run(BACKWARD);
  delay(400);                //how much u want to turn
  leftMotor.run(FORWARD);
  rightMotor.run(FORWARD);
}

void leftTurn()
{
  Serial.println("left");
  leftMotor.run(BACKWARD);
  rightMotor.run(FORWARD);
  delay(400);                  //how much u want to turn
  leftMotor.run(FORWARD);
  rightMotor.run(FORWARD);
}
int readping()
{
  int duration, distance; // start the scan
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2); // delays are required for a succesful sensor operation.
  digitalWrite(TRIG_PIN, HIGH);

  delayMicroseconds(10); //this delay is required as well!
  digitalWrite(TRIG_PIN, LOW);
  duration = pulseIn(ECHO_PIN, HIGH);
  distance = (duration / 2) / 29.1; // convert the distance to centimeters.
  Serial.println("DISTANCE IS");
  Serial.println(distance);
  return distance;
}

void moveStop()
{
  leftMotor.run(RELEASE);
  rightMotor.run(RELEASE);
}

void happy()
{
  Serial.println("left");
  leftMotor.run(BACKWARD);
  rightMotor.run(FORWARD);
  delay(1600);
  Serial.println("right");
  leftMotor.run(FORWARD);
  rightMotor.run(BACKWARD);
  delay(1600);
  moveStop();
}
void sad()
{
  Serial.println("moving forward");
  leftMotor.run(FORWARD);
  rightMotor.run(FORWARD);
  delay(800);
  Serial.println("moving backward");
  leftMotor.run(BACKWARD);
  rightMotor.run(BACKWARD);
  delay(800);
  moveStop();
}
void angry()
{
  Serial.println("moving forward");
  leftMotor.run(FORWARD);
  rightMotor.run(FORWARD);
  delay(1600);
  Serial.println("right");
  leftMotor.run(FORWARD);
  rightMotor.run(BACKWARD);
  delay(800);
  Serial.println("moving forward");
  leftMotor.run(FORWARD);
  rightMotor.run(FORWARD);
  delay(1600);
  Serial.println("right");
  leftMotor.run(FORWARD);
  rightMotor.run(BACKWARD);
  delay(800);
  moveStop();
}
```