

**School of Computing**

**SRM IST, Kattankulathur – 603 203**

**Course Code: 18CSC206J**

**Course Name: Software Engineering and Project Management**

|  |  |
| --- | --- |
| **Experiment No** | 13 |
| **Title of Experiment** | Provide the details of Architecture Design/Framework/Implementation |
| **Name of the candidate** | VEMULA AKSHAY DUTH |
| **Team Members** | BHARGAV VARMA,MAHIN SHARON |
| **Register Numbers** | RA2011003011113,RA2011003011106,RA2011003011101 |
| **Date of Experiment** | 19/06/2022 |

**Mark Split Up**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Description** | **Maximum Mark** | **Mark Obtained** |
| 1 | Exercise | 5 |  |
| 2 | Viva | 5 |  |
| **Total** | | **10** |  |

**Staff Signature with date**

**Aim**

To provide the details of architectural design/framework/implementation

**Team Members:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S No** | **Register No** | **Name** | **Role** |
| **1** | **RA2011003011113** | **VEMULA AKSHAY DUTH** | **Rep/Member** |
| **2** | **RA20110030111106** | **BHARGAV VARMA** | **Member** |
| **3** | **RA20110030111101** | **MAHIN SHARON** | **Member** |

# IMPLEMENTATION

The implementation details of each the modules can be explained as follows:

**1 Video Acquisition**

OpenCV provides extensive support for acquiring and processing live videos. It is also possible

to choose whether the video has to be captured from the in-built webcam or an external camera

by setting the right parameters. As mentioned earlier, OpenCV does not specify any minimum

requirement on the camera, however OpenCV by default expects a particular resolution of the

video that is being recorded, if the resolutions do not match, then an error is thrown. This error

can be countered, by over riding the default value, which can be achieved, by manually

specifying the resolution of the video being recorded.

**2 Dividing into frames**

Once the video has been acquired, the next step is to divide it into a series of frames/images. This

was initially done as a 2 step process. The first step is to grab a frame from the camera or a video

file, in our case since the video is not stored, the frame is grabbed from the camera and once this

is achieved, the next step is to retrieve the grabbed frame. While retrieving , the image/frame is

first decompressed and then retrieved . However, the two step process took a lot of processing

time as the grabbed frame had to be stored temporarily. To overcome this problem, we came up

with a single step process, where a single function grabs a frame and returns it by decompressing.

**3 Face detection**

Once the frames are successfully extracted the next step is to detect the face in each of these

frames. This is achieved by making use of the Haarcascade file for face detection. The

Haarcascade file contains a number of features of the face , such as height , width and thresholds

of face colors., it is constructed by using a number of positive and negative samples. For face

detection, we first load the cascade file. Then pass the acquired frame to an edge detection

function, which detects all the possible objects of different sizes in the frame. To reduce the

amount of processing, instead of detecting objects of all possible sizes, since the face of the

automobile driver occupies a large part of the image, we can specify the edge detector to detect

only objects of a particular size, this size is decided based on the Haarcascade file, wherein each

Haarcascade file will be designed for a particular size. Now, the output the edge detector is stored

in an array. Now, the output of the edge detector is then compared with the cascade file to

identify the face in the frame. Since the cascade consists of both positive and negative samples, it

is required to specify the number of failures on which an object detected should be classified as a

negative sample. In our system, we set this value to 3, which helped in achieving both accuracy as

well as less processing time. The output of this module is a frame with face detected in it.

**4 Eye detection**

After detecting the face, the next step is to detect the eyes, this can be achieved by making use of

the same technique used for face detection. However, to reduce the amount of processing, we

mark the region of interest before trying to detect eyes. The region of interest is set by taking into

account the following:

Computer Science & Information Technology (CS & IT) 50

• The eyes are present only in the upper part of the face detected.

• The eyes are present a few pixels lower from the top edge of the face.

Once the region of interest is marked, the edge detection technique is applied only on the region

of interest, thus reducing the amount of processing significantly. Now, we make use of the same

technique as face detection for detecting the eyes by making use of Haarcascade Xml file for eyes

detection. But, the output obtained was not very efficient, there were more than two objects

classified as positive samples, indicating more than two eyes. To overcome this problem, the

following steps are taken:

• Out of the detected objects, the object which has the highest surface area is obtained. This is

considered as the first positive sample.

• Out of the remaining objects , the object with the highest surface area is determined. This is

considered as the second positive sample.

• A check is made to make sure that the two positive samples are not the same.

• Now, we check if the two positive samples have a minimum of 30 pixels from either of the

edges.

• Next , we check if the two positive samples have a minimum of 20 pixels apart from each

other.

After passing the above tests, we conclude that the two objects i.e positive sample 1 and positive

sample 2 , are the eyes of the automobile driver.

**5 Drowsiness detection**

Once the eyes are detected, the next step is to determine if the eyes are in closed or open state.

This is achieved by extracting the pixel values from the eye region. After extracting , we check if

these pixel values are white, if they are white then it infers that the eyes are in the open state, if

the pixel values are not white then it infers that the eyes are in the closed state.

This is done for each and every frame extracted. If the eyes are detected to be closed for two

seconds or a certain number of consecutive frames depending on the frame rate, then the

automobile driver is detected to be drowsy. If the eyes are detected to be closed in non

consecutive frames, then We declare it as a blink.

If drowsiness is detected, a text message is displayed along with triggering an audio alarm. But, it

was observed that the system was not able to run for an extended period of time , because the

conversion of the acquired video from RGB to grayscale was occupying too much memory. To

overcome this problem, instead of converting the video to grayscale, the RGB video only was

used for processing. This conversion resulted in the following advantages,

• Better differentiation between colors, as it uses multichannel colors.

• Consumes very less memory.

• Capable of achieving blink detection, even when the automobile driver is wearing

spectacles.

51 Computer Science & Information Technology (CS & IT)

Hence there were two versions of the system that was implemented; the version 1.0 involves the

conversion of the image to grayscale. Currentlyversion 2.0 makes use of the RGB video for

processing.

**CODE:**

import numpy as np

import cv2

import pickle

import pyttsx3

import dlib

from scipy.spatial import distance

engine = pyttsx3.init('sapi5')

def speak(audio):

        engine.say(audio)

        engine.runAndWait()

def calculate\_EAR(eye):

    A = distance.euclidean(eye[1], eye[5])

    B = distance.euclidean(eye[2], eye[4])

    C = distance.euclidean(eye[0], eye[3])

    ear\_aspect\_ratio = (A+B)/(2.0\*C)

    return ear\_aspect\_ratio

cap = cv2.VideoCapture(0)

hog\_face\_detector = dlib.get\_frontal\_face\_detector()

dlib\_facelandmark = dlib.shape\_predictor("shape\_predictor\_68\_face\_landmarks.dat")

face\_cascade = cv2.CascadeClassifier('cascades/haarcascade\_frontalface\_alt2.xml')

recognizer = cv2.face.LBPHFaceRecognizer\_create()

recognizer.read('trainner.yml')

label ={"person name": 1}

with open('label.pickle', 'rb') as f:

    label = pickle.load(f)

    label = {v:k for k,v in label.items()}

    cap = cv2.VideoCapture(0)

count = 1

while  True:

    rect, frame = cap.read()

    gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

    faces = face\_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

    for (x,y,w,h) in faces:

        roi\_gray = gray[y:y+h, x:x+w]

        roi\_color = frame[y:y+h, x:x+w]

        id\_, conf = recognizer.predict(roi\_gray)

        if conf >= 60:

            font = cv2.FONT\_HERSHEY\_SIMPLEX

            name = label[id\_]

            color = (0,0,255)

            stroke = 2

            if count ==1:

                speak('HELLO ' + name)

                count +=1

            gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

            faces = hog\_face\_detector(gray)

            for face in faces:

                face\_landmarks = dlib\_facelandmark(gray, face)

                leftEye = []

                rightEye = []

                for n in range(36,42):

                    x = face\_landmarks.part(n).x

                    y = face\_landmarks.part(n).y

                    leftEye.append((x,y))

                    next\_point = n+1

                    if n == 41:

                        next\_point = 36

                    x2 = face\_landmarks.part(next\_point).x

                    y2 = face\_landmarks.part(next\_point).y

                    cv2.line(frame,(x,y),(x2,y2),(0,255,0),1)

                for n in range(42,48):

                    x = face\_landmarks.part(n).x

                    y = face\_landmarks.part(n).y

                    rightEye.append((x,y))

                    next\_point = n+1

                    if n == 47:

                        next\_point = 42

                    x2 = face\_landmarks.part(next\_point).x

                    y2 = face\_landmarks.part(next\_point).y

                    cv2.line(frame,(x,y),(x2,y2),(0,255,0),1)

                left\_ear = calculate\_EAR(leftEye)

                right\_ear = calculate\_EAR(rightEye)

                EAR = (left\_ear+right\_ear)/2

                EAR = round(EAR,2)

                if EAR < 0.25:

                    speak('DROWSY')

                    speak('Are you Sleepy?')

        color = (0, 255, 0)

        stroke = 2

        end\_chord\_x = x + w

        end\_chord\_y = x + h

        cv2.imshow('frame', frame)

    key = cv2.waitKey(1)

    if key == 27:

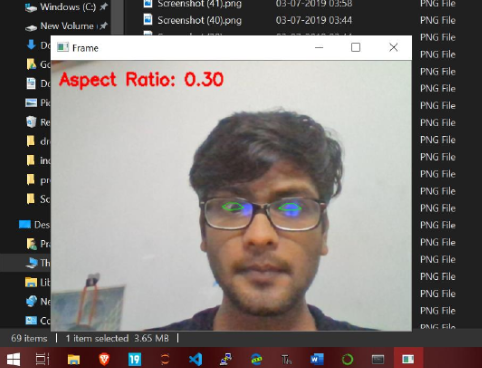
        break

cap.release()

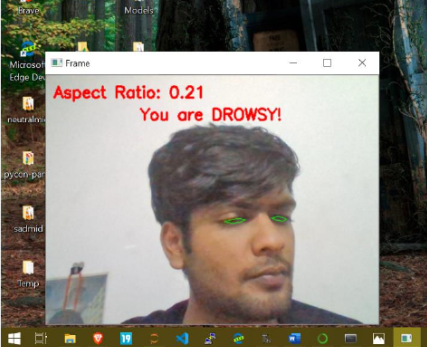
cv2.destroyAllWindows()

**SOME SCREEN SHOTS OF OUR PROJECT**

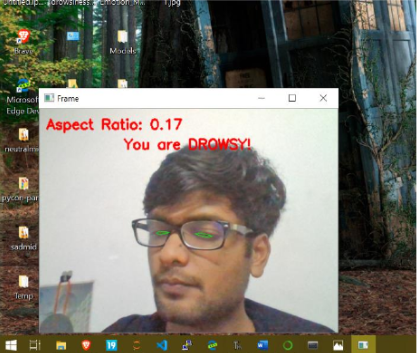
**NON DROWSY PERSON**

****

**DROWSY PERSON**

****

**DROWSYPERSON WEARING SPECTS**

****

Result:

Thus, the details of architectural design/framework/implementation along with the screenshots were provided for our project