Smart Internz

Strain Analysis based on EyeBlinking

Team Members:

S. Hemanth Sai Nivas (20481A5449) (Team lead)

A. Khadar Jilani (20481A5401)

A. Danny (20481A5403)

K. Pushpa Sai Kavya(20481A5429)

1.INTRODUCTION

Strain analysis based on eye blinking" is a machine learning project that uses the natural blinking patterns of the eye to detect and analyse strain and fatigue levels in individuals. The project works by tracking the frequency and duration of eye blinks, and then using machine learning algorithms to analyse the blink patterns and determine the strain and fatigue levels of the individual. The project has the potential to be used in a variety of settings, including schools, workplaces, and healthcare facilities.

The project aims to create a system that can monitor a person's eye blinking behaviour withoutinterfering with their activities. The system will analyze the blink patterns to provide insights into the person's level of strain or fatigue. The system is based on the assumption that eye blinking patterns can reflect a person's cognitive state and stress levels.

The usage of computers, mobiles, and other display related electronic has been increasing along with the time. Many jobs had turned from paper work to the computer work which puts the human for long time gazing at a computer screen. This results as one of the primary causes of vison problems. Insufficient eye movement, looking at computer screen for hours can cause vision problems. The distance between the screen and the user's head typically remains constant, resulting in infrequent use of the eye muscles for adjustment. As a consequence, these muscles can become weaker over time. Additionally, the reduced frequency of eye blinking can lead to excessive dryness of the eye surface (cornea and sclera), posing potential harm to the eyes. Chronic dry eyes can eventually result in corneal scarring and even vision loss.

However, adopting preventive measures like taking regular breaks, doing eye exercises, and engaging in relaxation activities can effectively avoid many of these eye disorders. Unfortunately, most people are hesitant to change their workplace habits until they experience the initial signs of health issues. To raise awareness about this problem and aid in prevention, we propose using a simple webcam mounted on the monitor to capture video of the user at their workplace and analyze their eye blinking patterns.

In the European Community, over 40% of the current working population relies on computers for their daily tasks. However, this computer use is associated with static work, prolonged sitting, and vision issues. For instance, about 70% of computer workers globally experience vision problems, leading to what is known as Computer Vision Syndrome. As the number of computer-related jobs is projected to rise substantially in the next decade, so too is the expected increase in workplace-related illnesses.

1.1 Purpose: -

The objective of the "Strain Analysis based on Eye Blinking" project is to create a machine learning system capable of evaluating and analyzing an individual's mental and physical strain levels through the analysis of their eye blinking patterns. The project aims to accomplish the following goals:

1. Non-Intrusive Strain Monitoring: The project seeks to offer a less intrusive method for monitoring an individual's stress or strain levels by analyzing
2. their eye blinking patterns. Unlike traditional methods that may disrupt natural behavior, this approach provides a more seamless and non-intrusive way to understand strain levels.

2. Real-Time Assessment: The system aims to provide real-time evaluation of strain levels. This feature allows for timely interventions and support, particularly in contexts where immediate feedback can be valuable, such as workplaces, educational settings, or medical procedures.

3. Objective Strain Measurement: Leveraging machine learning techniques, the project aims to objectively quantify strain levels based on eye blinking patterns.

4. Insights for Well-Being Improvement: Analyzing strain patterns can offer valuable insights into an individual's well-being. By identifying triggers and understanding how strain levels change over time, the project can assist in implementing strategies to improve overall well-being and manage stress more effectively.

.

2. LITERATURE SURVEY

2.1 Existing problem: -

Accurately capturing eye blinking data can be challenging. Traditional methods like manual observation or video recording might not provide the required precision for strain analysis. Advanced technologies such as eye-tracking systems or specialized sensors may be needed to obtain more accurate data.

Eye blinking patterns can vary significantly between individuals and even in the same individual under different conditions. This variability can complicate the analysis and interpretation of strain-related data. External factors, such as lighting conditions, screen brightness, and environmental stress, can influence blinking behavior. Understanding and controlling these factors are crucial for accurate strain analysis.

Strain analysis of eye blinking might work well in controlled laboratory settings, but its application in real-world scenarios, such as during extended computer usage or driving, might introduce additional challenges.

2.2 Proposed solution: -

The eye blink is a fast closing and reopening of a human eye. Each individual has a little bit different pattern of blinks. The pattern differs in the speed of closing and opening, a degree of squeezing the eye and in a blink duration. The eye blink lasts approximately 100-400 ms. We propose to exploit state-of-the-art facial landmark detectors to localize the eyes and eyelid contours. From the landmarks detected in the image, we derive the eye aspect ratio (EAR) that is used as an estimate of the eye opening state. Since the per frame EAR may not necessarily recognize the eye blinks correctly, a classifier that takes a larger temporal window of a frame into account is trained.

Blink detection method: For every video frame, the eye landmarks are detected. The eye aspect ratio (EAR) between height and width of the eye is computed.

$$\mathrm{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

where p1, . . ., p6 are the 2D landmark locations, depicted in Fig. 1. When the eye is open, the value is almost constant, but it approaches zero when the eye is closed. This value does not depend on head posture or distance, and there are small individual differences when the eyes are open. Since eye blinking is performed by both eyes synchronously, the EAR of both eyes is averaged. Due to the simplicity of the calculation formula, it has excellent real-time performance and shows high robustness. However, although we are trying to classify irregular movements such as eye thinning, absent stretching, or only eyes looking down, detection is difficult at the threshold. Also, although it is claimed to have high and straightforward detection accuracy.

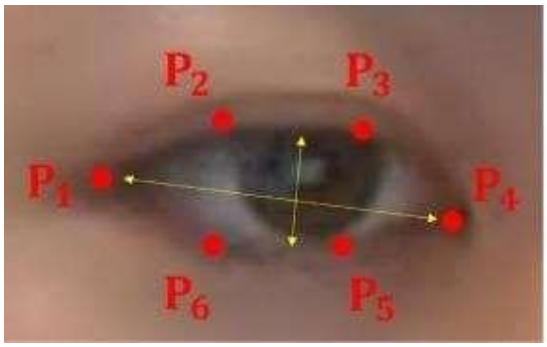
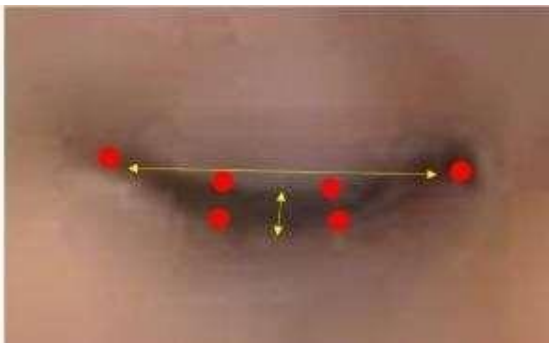


Fig-1



EAR is mostly constant when an eye is open and is getting close to zero while closing an eye. It is partially person and head pose insensitive. Aspect ratio of the open eye has a small variance among individuals and it is fully invariant to a uniform scaling of the image and in-plane rotation of the face. Since eye blinking is performed by both eyes synchronously, the EAR of both eyes is averaged. An example of an EAR signal over the video sequence.

Importing Necessary Libraries:

The first step is usually importing the libraries that will be needed in the program.
The required libraries to be imported to Python script are:

- SciPy
- Imutils
- Numpy
- Argparse
- Time
- Datetime
- Dlib
- Opencv
- Google text to speech

Defining Necessary Functions:

We are defining necessary functions that can be used in the latter part of the program. The necessary functions are

1. **playaudio(text):**

   In this function, we are translating the text input to a speech by using gTTS and saving the translated speech to the output1.mp3 file. We are returning the output1.mp3 to the calling function.

2. **popupmsg(msg):**

   Creating an instance of Tk initializes the interpreter and creates the root window.

3. **eye_aspect_ratio(eye):**

   We can Calculate Eye Aspect Ratio:

   - Compute the Euclidean distances between the two sets of vertical eye landmarks (x, y)-coordinate.
   - Compute the Euclidean distance between the horizontal eye landmark (x, y)-coordinates.
   - Compute the eye aspect ratio using the above formula and then return ear to the calling function.

Construct Argparser:

We are using argparse library to parse command-line arguments.ArgumentParser() is a predefined class

Flags:

--shape-predictor: This is the path to dlib's pre-trained facial landmark detector.

--video: This is used to access a video file that is residing on the disk. If you want to use a webcam, you can simply omit this.

Defining Important Constants

Let us define two important constants

- **EYE_AR_THRESH**
  - By using ear value, we determine if a blink is taking place in the video stream.
  - A "blink" is registered when ear value falls below a certain EYE_AR_THRESH and then rises above the **EYE_AR_THRESH.**

- We default it to a value of 0.3 as this is what has worked best for my applications, but you may need to tune it for your application.

- **EYE_AR_CONSEC_FRAMES**
  - This value is set to 3 to indicate that three successive frames with an eye aspect ratio less than EYE_AR_THRESH must happen for a blink to be registered.

Let us initialize two counters for counting the number of blinks.

- **COUNTER**
  - It is the total number of successive frames that have an eye aspect ratio less than EYE_AR_THRESH
- **TOTAL**
  - It is the total number of blinks that took place while the script has been running.

Get The Face Land Marks Using Dlib

We first load the detector using the g**et_frontal_face_detector()** and facial landmark predictor dlib.shape_predictor from dlib library.

Capturing The Input Frames

The average number of blinks per minute should be at least 10-14. Hence, setting the eye_thresh default value to 10. But this can be modified based on the application to improve accuracy.

The function **datetime.datetime.now()**.minute gives the minute at that instant.

There are two ways we can capture the input video

1. using in-built webcam

2. using video file residing on the disk

Converting Frames to Grayscale Channels

cv2.cvtColor(frame, flag): is used for colour conversion.

cv2.COLOR_BGR2GRAY: The flag is used to convert the coloured image to grayscale.

Detect links

The aspect ratio will be approximately constant while the eye is open, and it will quickly fall to zero when a blink occurs. We need to determine the threshold for a blinking ratio that is near to zero.

Alerting The User

Calculate the average number of blinks:

Calculating the average number of blinks per minute by using the below code.

Initiate the alarm and popup message:

If the total blink count is less or more than the average blink count for the stipulated time(calculated for every minute incrementally), then an alert is initiated using audio and popup messages to take rest by calling the playaudio and popup functions.

Display The Result

The cv2.putText function displays the number of blink and ear on the OpenCV window once a blink count is detected.

The cv2.imshow() function always takes two more functions to load and close the image. These two functions are cv2.waitKey() and cv2.destroyAllWindows(). Inside the cv2.waitKey() function, you can provide any value to close the image and continue with further lines of code.

Run The Application

To access the built-in webcam execute the following command in anaconda prompt:

python app_eye.py --shape-predictor shape_predictor_68_face_landmarks.dat

To access video file residing on the disk execute the following command in anaconda prompt

python app_eye.py --shape-predictor shape_predictor_68_face_landmarks.dat –video filename.mp3

3. EXPERIMENTAL INVESTIGATIONS

In the realm of physiological responses, the human body's intricate subtleties often hold the key to understanding its underlying mechanisms. One such phenomenon, which has gained attention in recent times, is the analysis of strain through the study of eye blinking patterns.

The interplay between psychological and psychological factors in the context of strain and stress has led researchers to explore the potential of eye blinking as a non-intrusive and easily accessible indicator. This article delves into the experimental investigations on strain analysis based on eye blinking, highlighting its significance, methodologies, findings, and potential applications.

Significance and Rationale: Stress and strain become ubiquitous aspects of modern life, impacting physical, emotional, and mental well-being.

Methodologies: Experimental investigations into strain analysis via eye blinking employ diverse methodologies to capture and analyze blinking patterns in controlled environments. One common approach involves the use of electrooculography (EOG), which records the electrical potential generated by eye movements.

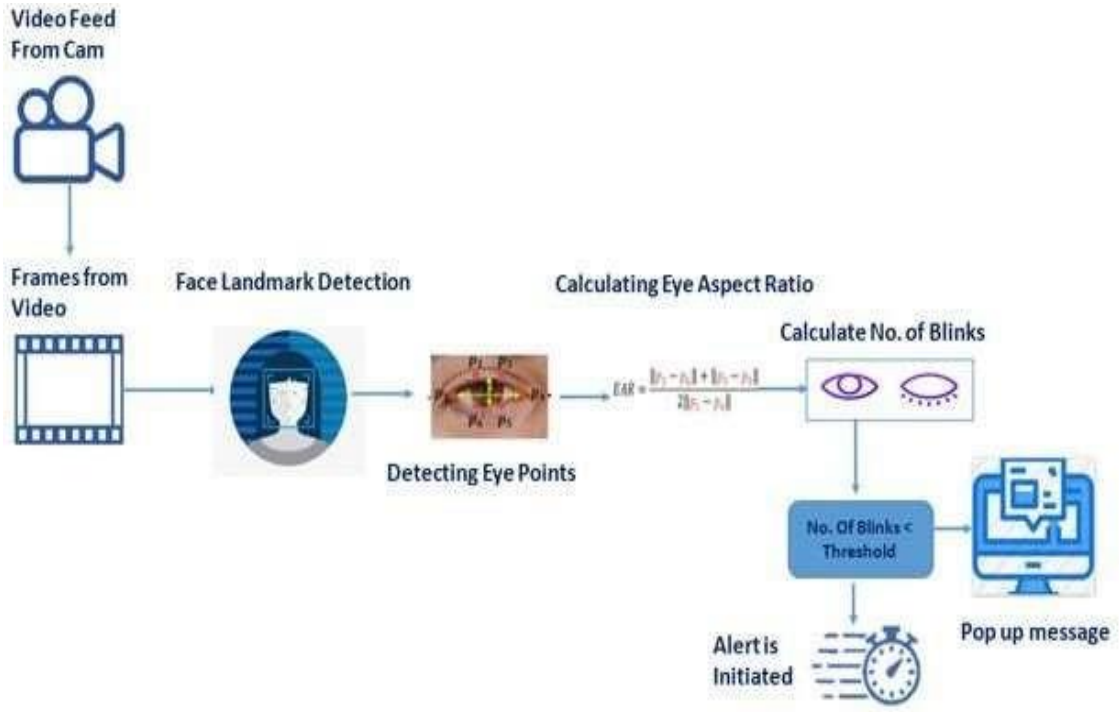
Conclusion: Experimental investigations on strain analysis based on eye blinking have illuminated a new avenue for understanding the human stress response, the involuntary and dynamic nature of blinking, further research, interdisciplinary collaboration, and ethical considerations are vital to unlocking the full potential of eye blinking analysis in the realm of strain and stress analysis.

Experimental environment:

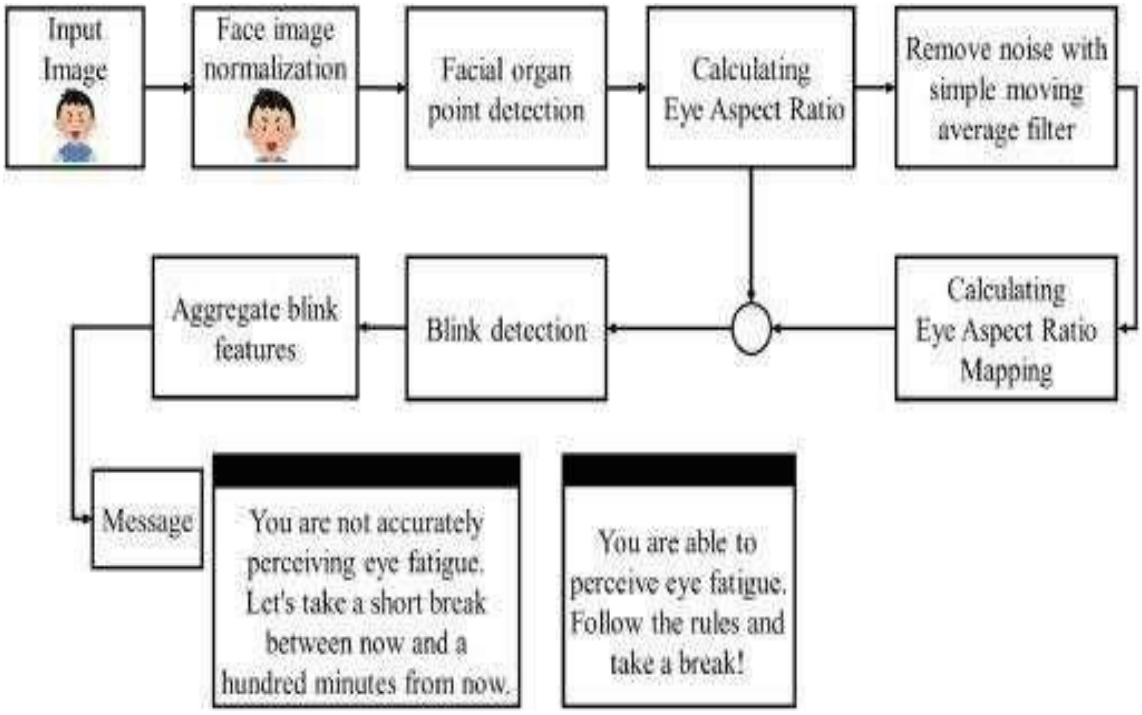
In this experiment, we worked indoors, shutting out the outside light to not depend on the weather conditions of the day. The room temperature was set at 26 ° Celsius, and the air conditioning was turned off before the start of the experiment in order to prevent the eyes from drying out. The luminous intensity of the display was set to the highest luminance that could be set. All the subjects were tested one by one in the same room, and no outsiders were allowed in the room while they were working to elicit their concentration.

4. THEORETICAL ANALYSIS

4.1Block diagram: -

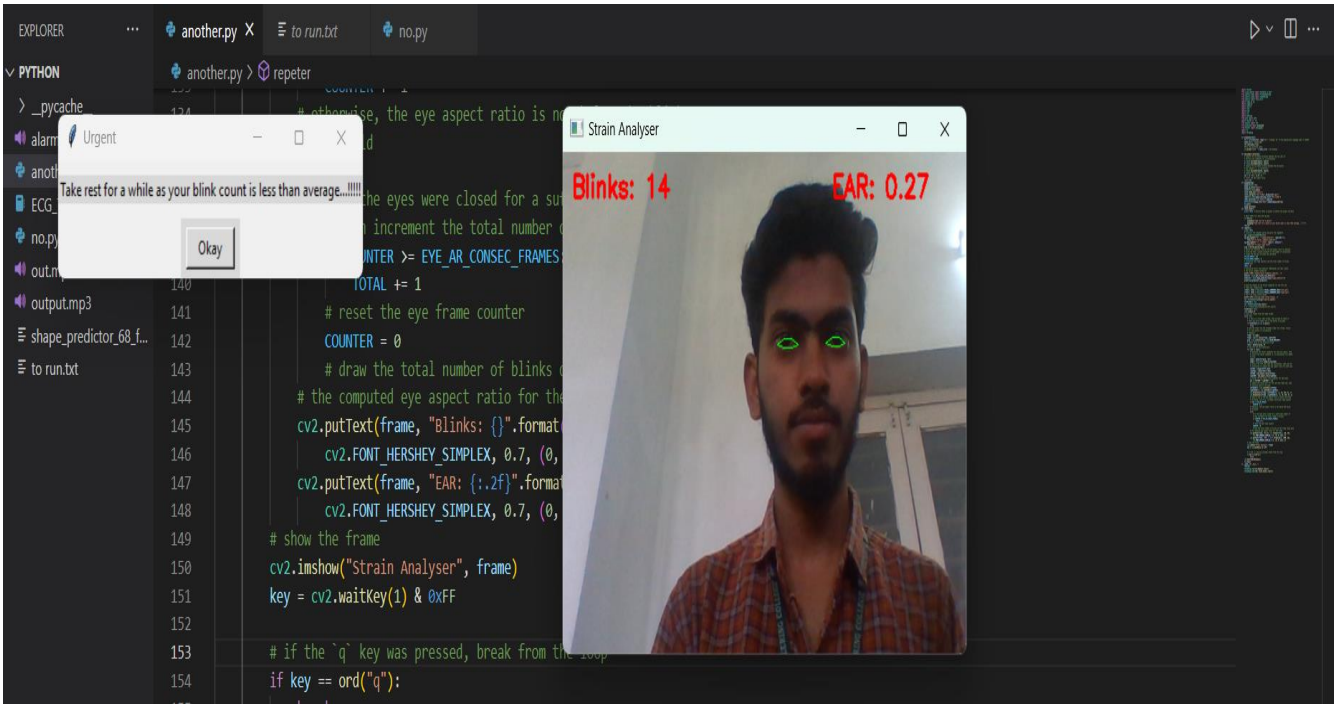


5. FLOWCHART



6. RESULT

Eye blinking can provide insightful results for strain analysis. By quantifying blink frequency, duration, and patterns, researches can deduce cognitive load and emotional stress. Increased blinks might indicate heightened strain, as sustained tasks strain visual and cognitive systems. Blink asymmetry could signify localized stress. Correlating blink data with environmental factors offer richer context. Strain may evoke rapid, irregular blinks. Analyzing blinking alterations during tasks aids ergonomic optimization. Overall, blink based strain analysis presents a non-intrusive, real-time method to gauge human responses, enhancing workplace design, human-computer interaction, and psychological state assessment.



7. ADVANTAGES & DISADVANTAGES

Advantages:

1. **High-resolution analysis:** Blinking, especially in advanced imaging techniques, might offer high-resolution data, enabling the detection of small or subtle strain changes.
2. **Simplicity and cost-effectiveness:** If blinking-based strain analysis can be implemented with standard equipment and without the need for expensive sensors, it could be a cost-effective alternative to existing methods.
3. **Minimization of disturbances:** Some traditional strain measurement techniques can affect the behavior of the material being analyzed. Blinking-based analysis, if appropriately designed, might minimize these disturbances.

Disadvantages:

1. **Inaccuracy:** Eye blinking is not a highly precise or reliable indicator of strain. The intensity and frequency of blinking can vary significantly between individuals and even within the same person under different conditions. This variability makes it difficult to establish consistent and accurate measurements of strain.
2. **Limited range of information:** Eye blinking can provide some information about a person's stress levels or fatigue, but it lacks the ability to capture a comprehensive picture of strain. Strain is a complex physiological and psychological phenomenon, and relying solely on eye blinking may not fully capture the intricacies involved.
3. **Context dependence**: Blinking patterns can vary depending on the context. For example, a person may blink more frequently while reading, using a computer, or watching TV, even if they are not under significant strain. This context dependence makes it challenging to differentiate between normal blinking and blinking related to strain.

8. APPLICATIONS

1. Stress Monitoring and Mental Health: The project can be used to monitor stress levels and assess mental well-being based on patterns of eye blinking. It could help individuals become more aware of their stress levels and provide insights for mental health professionals to support their patients.
2. Workplace Safety: In high-stress work environments, such as aviation, transportation, or healthcare, the project can be applied to monitor fatigue and alertness levels of workers to ensure safety and prevent accidents.
3. Preventing Computer-Related Eye Strain: The project's findings can be used to develop software that reminds computer users to take breaks or adjust their screen time based on eye blinking patterns to reduce eye strain.
4. Classroom and Education: Eye blinking analysis can be applied to assess students' engagement and attention levels during lectures or educational videos, helping educators adjust their teaching methods.

9. CONCLUSION

The "Strain Analysis based on Eye Blinking" project is a valuable advancement in stress monitoring and well-being assessment. By combining computer vision, machine learning, and real-time analysis, it offers practical applications with the potential to positively impact diverse aspects of people's lives, from workplace productivity to mental health support. As research in this field continues to evolve, the project's findings and methodologies will contribute significantly to ongoing efforts in stress management and human-computer interaction, fostering healthier and more productive living environments.

We introduced a novel video-based eye-blink detection approach designed to alert computer users about potentially hazardous blinking behavior. Our method surpasses existing preventive software, typically relying on keyboard and mouse activity. The

approach involves a two-level analysis, enabling us to identify blinks and various other eye movements accurately. To enhance performance, we intend to implement GPU-based optical flow estimation instead of simple normal flow calculation. Additionally, upgrading the Lucas-Kanade tracker will enable more precise tracking of rapid and significant facial movements.

10. FUTURE SCOPE

“Strain Analysis based on Eye Blinking" has promising prospects for future advancements and applications. As technology and research continue to evolve, the project's potential scope can expand into various areas of development and growth:

1. Mental Health Support: By analyzing eye blinking patterns, researchers and clinicians could gain insights into mental health conditions, contributing to the early detection and treatment of certain psychological disorders.
2. Workplace Safety and Productivity: Eye blinking-based strain analysis could be employed to assess fatigue levels in workplaces, especially in industries where alertness is crucial, such as transportation, healthcare, and aviator.

11. BIBLIOGRAPHY

1. Ali, N. H., Ahmad, F., & Hussin, N. H. (2019). Eye Blink Detection using Image Processing Technique: A Review. Journal of Physics: Conference Series, 1150(1), 012080.

2. Bhushan, B., Pal, R., Saini, S., & Verma, S. K. (2019). Eye Blink Detection for Human-Machine Interaction. International Journal of Computer Applications, 182(42), 38-41.

3. Gamage, A. U., Rathnayaka, R. M. C. A., & Meegama, R. P. (2021). Blink Detection for Strain Analysis and Eye Health Monitoring. 2021 IEEE International Conference on Data Science and Advanced Analytics (DSAA).

4. Jain, S., Mittal, A., & Rastogi, A. (2017). Eye Blink Detection Using OpenCV for Driver Drowsiness Detection. 2017 2nd International Conference for Convergence in Technology (I2CT).

Source code

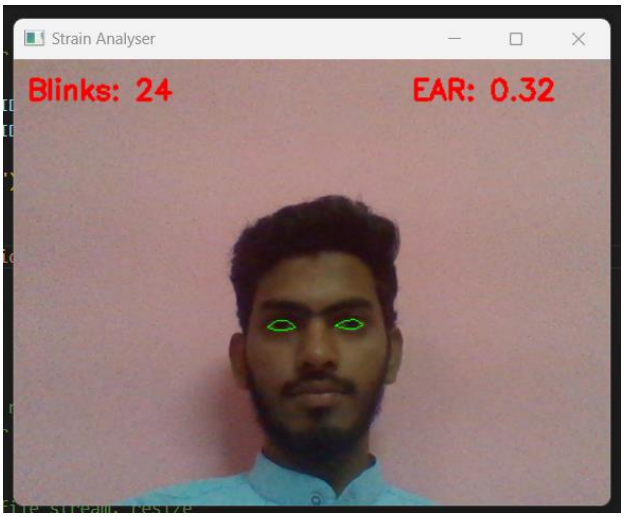
```
import tkinter
from scipy.spatial import
    distance as dist
from imutils.video import
    FileVideoStream
from imutils.video import
    VideoStream
from imutils import
    face_utils
import numpy as np
import argparse
import imutils
import time
import dlib
import cv2
import datetime
from gtts import gTTS
import tkinter as  tk
from tkinter import ttk
from playsound import
    playsound
from tkinter import
    messagebox
import os
import threading

def playaudio(text):
    tts = gTTS(text=text,
        lang='en')  # Change
        'en' to the appropriate
        language code if
        needed
    audio_file =
        "output.mp3"
    tts.save(audio_file)
    #Play the generated
        audio
    os.system('start ' +
        audio_file)  # On
        Windows

def eye_aspect_ratio(eye):
    # compute the euclidean
        distances between the
        two sets of
    # vertical eye landmarks
        (x, y)-coordinates
    A =
        dist.euclidean(eye[1],
        eye[5])
    B =
        dist.euclidean(eye[2],
        eye[4])
    # compute the euclidean
        distance between the
        horizontal
    # eye landmark (x, y)-
        coordinates
    C =
        dist.euclidean(eye[0],
        eye[3])
    # compute the eye aspect
        ratio
    ear = (A + B) / (2.0 * C)
    # return the eye aspect
        ratio
    return ear
def popupmsg(msg):
    popup=tkinter.Tk()

        popup.wm_title("Urg
        ent")
    style=ttk.Style(popup)
    style.theme_use('classic')

        style.configure('Test.
        Tlabel',background='a
        qua')

        label=ttk.Label(popup
        ,text=msg,style='Test.
        TLabel')

        label.pack(side='top',f
        ill="x",pady=10)

        B1=ttk.Button(popup,
        text="Okay",comman
        d=popup.destroy)
    B1.pack()
    popup.mainloop()
def reset_total():
    global TOTAL  #
        Declare TOTAL as
        global to modify the
        global variable

    # Reset TOTAL to 0
        every 60 seconds
    if TOTAL<8:
        playaudio("Take rest
        for a while")
        popupmsg("Take rest
        for a while as your
        blink count is less
        than average...!!!!!")
    TOTAL=0
def repeter():
    global TOTAL
    # construct the argument
        parse and parse the
        arguments
    ap =
        argparse.ArgumentPa
        rser()
    ap.add_argument("-p", "-
        -shape-predictor",
        required=True,
        help="path to facial
        landmark predictor")
    ap.add_argument("-v", "-
        -video", type=str,
        default="",
        help="path to input
        video file")
    args =
        vars(ap.parse_args())
    # define two constants,
        one for the eye aspect
        ratio to indicate
    # blink and then a second
        constant for the
        number of
        consecutive
    # frames the eye must be
        below the threshold
    EYE_AR_THRESH =
        0.3
    EYE_AR_CONSEC_FR
        AMES = 3
    # initialize the frame
        counters and the total
        number of blinks
    COUNTER = 0
    TOTAL = 0
    # initialize dlib's face
```

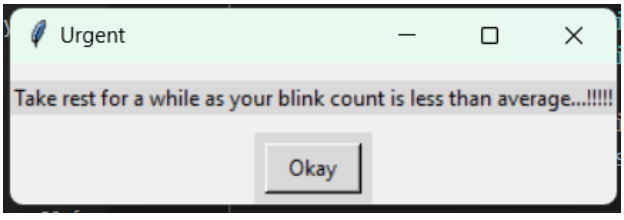
```
    detector (HOG-based)
    and then create
# the facial landmark
    predictor
print("[INFO] loading
    facial landmark
    predictor...")
detector =
    dlib.get_frontal_face_
    detector()
predictor =
    dlib.shape_predictor(a
    rgs["shape_predictor"
    ])
print(type(predictor),pre
    dictor)

# grab the indexes of the
    facial landmarks for
    the left and
# right eye, respectively
(lStart, lEnd) =
    face_utils.FACIAL_L
    ANDMARKS_IDXS[
    "left_eye"]
(rStart, rEnd) =
    face_utils.FACIAL_L
    ANDMARKS_IDXS[
    "right_eye"]
# start the video stream
    thread
print("[INFO] starting
    video stream
    thread...")
vs =
    FileVideoStream(args
    ["video"]).start()
fileStream = True
vs =
    VideoStream(src=0).s
    tart()
# vs =
    VideoStream(usePiCa
    mera=True).start()
fileStream = False
time.sleep(1.0)
# loop over frames from
    the video stream
while True:
    # if this is a file video
    stream, then we need
    to check if
    # there any more
    frames left in the
    buffer to process
    if fileStream and not
    vs.more():
        break
    # grab the frame from
    the threaded video file
    stream, resize
    # it, and convert it to
    grayscale
    # channels)
    frame = vs.read()
    frame =
    imutils.resize(frame,
    width=450)
    gray =
    cv2.cvtColor(frame,
    cv2.COLOR_BGR2G
    RAY)
    # detect faces in the
    grayscale frame
    rects = detector(gray,
    0)
    # loop over the face
    detections
```

```
    for rect in rects:
        # determine
the facial landmarks
for the face region,
then
        # convert the
facial landmark (x,
y)-coordinates to a
NumPy
        # array
        shape =
predictor(gray, rect)
        shape =
face_utils.shape_to_n
p(shape)
        # extract the
left and right eye
coordinates, then use
the
        # coordinates
to compute the eye
aspect ratio for both
eyes
        leftEye =
shape[lStart:lEnd]
        rightEye =
shape[rStart:rEnd]
        leftEAR =
eye_aspect_ratio(left
Eye)
        rightEAR =
eye_aspect_ratio(right
Eye)
        # average the
eye aspect ratio
together for both eyes
        ear =
(leftEAR + rightEAR)
/ 2.0
        # compute the
convex hull for the
left and right eye,
then
        # visualize
each of the eyes
        leftEyeHull =
cv2.convexHull(leftE
ye)
        rightEyeHull
=
cv2.convexHull(right
Eye)

        cv2.drawCont
ours(frame,
[leftEyeHull], -1, (0,
255, 0), 1)

        cv2.drawCont
ours(frame,
[rightEyeHull], -1, (0,
255, 0), 1)
        # check to see
if the eye aspect ratio
is below the blink
        # threshold,
and if so, increment
the blink frame
counter
        if ear <
EYE_AR_THRESH:

        COUNTER
+= 1
        # otherwise,
the eye aspect ratio is
not below the blink
        # threshold
```

```
                else:
                        #
            if the eyes were
            closed for a sufficient
            number of
                        #
            then increment the
            total number of blinks
                        if
            COUNTER >=
            EYE_AR_CONSEC_
            FRAMES:

                TOTAL += 1
                        #
            reset the eye frame
            counter

                COUNTER =
            0
                        #
            draw the total number
            of blinks on the frame
            along with
                # the
            computed eye aspect
            ratio for the frame

                cv2.putText(fr
            ame, "Blinks:
            {}".format(TOTAL),
            (10, 30),

                cv2.FONT_H
            ERSHEY_SIMPLEX,
            0.7, (0, 0, 255), 2)

                cv2.putText(fr
            ame, "EAR:
            {:.2f}".format(ear),
            (300, 30),

                cv2.FONT_H
            ERSHEY_SIMPLEX,
            0.7, (0, 0, 255), 2)
            # show the frame
            cv2.imshow("Frame",
            frame)
            key = cv2.waitKey(1)
            & 0xFF

            # if the `q` key was
            pressed, break from
            the loop
            if key == ord("q"):
                break
        cv2.destroyAllWindows(
            )
        vs.stop()
    if __name__=="__main__":
        TOTAL=0
        threading.Timer(0,repete
            r).start()
        threading.Timer(60,
            reset_total).start()
```

```
PS D:\Jilani\Python> python app.py --shape-predictor shape_predictor_68_face_landmarks.dat
[INFO] loading facial landmark predictor...
<class 'dlib.dlib_pybind11.shape_predictor'> <dlib.dlib_pybind11.shape_predictor object at 0x00000143EC57F530>
[INFO] starting video stream thread...
```





Output: