STRAIN ANALYSIS BASED ON EYE BLINKING

AN INDUSTRY ORIENTED UG PHASE -II

Submitted to

JAWAHARLAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted By

SHARANYA THOTA (21UK1A05Q4)

SAI KUMAR SAMUDRALA (22UK5A0521)

HUSSAIN JAFAR HUSSIAN (21UK1A05Q3)

Under the guidance of

Ms. T. SUSHMA

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) – 506005

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE WARANGAL

BOLLIKUNTA, WARANGAL(T.S)-506005

2021-2025

CERTIFICATE OF COMPLETION

INDUSTRY ORIENTED UG PHASE -II

This is to certify that the **UG PROJECT PHASE-II** entitled **"STRAIN ANALYSIS BASED ON EYE BLINKING"** is being submitted by **SHARANYA THOTA (21UK1A05Q4), SAI KUMAR SAMUDRALA (22UK5A0521), HUSSAIN JAFAR HUSSAIN (21UK1A05Q3)** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year 2024-2025, is a record of work carried out by them under the guidance and supervision.

Project Guide	**Head of the Department**
Ms. T. Sushma	**Dr. K. Sharmila Reddy**
(Assistant Professor)	(Professor)

External Examiner

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. SYED MUSTHAK AHAMED**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this **UG PROJECT PHASE -II** in the institute.

We extend our heartfelt thanks to **Dr. K. Sharmila Reddy**, Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the **UG PROJECT PHASE -II.**

We express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the Mini Project and for their support in completing the **UG PROJECT PHASE -II**.

We express heartfelt thanks to the guide **Ms.T. SUSHMA**, Assistant professor, Department of CSE for her constant support and giving necessary guidance for completion of the **UG PROJECT PHASE-II**.

Finally, we express our sincere thanks and gratitude to our family members, friends for their encouragement and outpouring their knowledge and experience throughout the project.

SHARANYA THOTA	**21UK1A05Q4**
SAI KUMAR SAMUDRALA	**22UK5A0521**
HUSSAIN JAFAR HUSSIAN	**21UK1A05Q3**

TABLE OF CONTENTS:

TOPIC **PAGENO**

1. INTRODUCTION 2
2. CODE SNIPPETS 3

2.1 PYTHON CODE 3

3. CONCLUSION 14
4. APPLICATIONS 15
5. ADVANTAGES 17
6. DISADVANTAGES 18
7. FUTURE SCOPE 19
8. BIBLIOGRAPHY 20
9. HELP FILE 21

LIST OF FIGURES PAGE NO

Figure 1: Import Necessary Libraries …………………………………………………….....3

Figure 2: playaudio(text)……………………………………………………………….3

Figure 3: popupmsg(msg)…………………………………………………...........4

Figure 4: eye_aspect_ratio(eye)…..……………………………………………………4

Figure 5: Construct Argparser ………………………………………………………….4

Figure 6: Eye_AR_Consec_Frames ……………………………………….................4

Figure 7: datetime.datetime.now()………………………………………………………5

Figure 8: capture the input video …………………………………………….............5

Figure 9: Computing convex hull for eyes ……………………………………………..6

Figure 10: Detect links ………………………………………………………………......6

Figure 11: Calculate the average number of blinks…………………………….…….6

Figure 12: Initiate the alarm and popup message……………………………….……..6

Figure 13: putText function …………………………………………………….……7

Figure 14: Home Page of Strain analysis based on eye blinking…………………………….12

Figure 15: Final output images……………………………………………………….…13

1. INTRODUCTION

Blinking is a reflex, which means your body does it automatically. Babies and children only blink about two times per minute. By the time you reach adolescence that increases to 10 to 14 times per minute.

Detecting eye blinks is important for instance in systems that monitor a human operator vigilance, e.g. driver drowsiness, in systems that warn a computer user staring at the screen without blinking for a long time to prevent the dry eye and the computer vision syndromes, in human-computer interfaces that ease communication for disabled people. There should be an application that monitors to let the user know that he might get strained.

A neural network model is built which alerts the user if eyes are getting strained. This model uses the integrated webcam to capture the face (eyes) of the person. It captures the eye movement and counts the number of times a person blinks. If blink count deviates from the average value (if the number of blinks is less or more), then an alert is initiated by playing an audio message along with a popup message is displayed on the screen appropriately.

2. CODE SNIPPETS

2.1 PYTHON CODE

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, playsound, Tkinter.

Importing the Libraries

```
# import the necessary packages
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
```

Figure 1. Import Necessary Libraries

```
def playaudio(text):
    speech=gTTS(text)
    print(type(speech))
    speech.save("../output1.mp3")
    playsound("../output1.mp3")
    return
```

Figure 2. playaudio(text)

```
LARGE_FONT= ("Verdana", 12)
NORM_FONT = ("Helvetica", 10)
SMALL_FONT = ("Helvetica", 8)
```

```
def popupmsg(msg):
    popup = tk.Tk()
    popup.wm_title("Urgent")
    style = ttk.Style(popup)
    style.theme_use('classic')
    style.configure('Test.TLabel', background= 'aqua')
    label = ttk.Label(popup, text=msg,style= 'Test.TLabel')
    label.pack(side="top", fill="x", pady=10)
    B1 = ttk.Button(popup, text="Okay", command = popup.destroy)
    B1.pack()
    popup.mainloop()
```

Figure 3.popupmsg(msg)

```
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
```

Figure 4.eye_aspect_ratio(eye)

```
# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
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())
```

Figure 5.Construct Argparser

```
EYE_AR_THRESH = 0.3
EYE_AR_CONSEC_FRAMES = 3
```

```
COUNTER = 0
TOTAL = 0
```

Figure 6. Eye_AR_Consec_Frames

Capturing the input frames:

```
eye_thresh = 10
before =datetime.datetime.now().minute
```

Figure 7.datetime.datetime.now()

```
if not args.get("video", False):
    # Taking input from web cam
    print("[INFO] starting video stream...")
    vs = VideoStream(src=0).start()
    time.sleep(1.0)
    #before =datetime.datetime.now().minute

else:
    print("[INFO] opening video file...")
    #Taking input as video file
    vs = cv2.VideoCapture(args["video"])
    time.sleep(1.0)
```

Figure 8.capture the input video

facial landmark detection:

```
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=450)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # detect faces in the grayscale frame
    rects = detector(gray, 0)
```

```
    # loop over the face detections
    for rect in rects:
        shape = predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]
        leftEAR = eye_aspect_ratio(leftEye)
        rightEAR = eye_aspect_ratio(rightEye)

        ear = (leftEAR + rightEAR) / 2.0
```

```
        # compute the convex hull for the left and right eye, then
        # visualize each of the eyes
        leftEyeHull = cv2.convexHull(leftEye)
        rightEyeHull = cv2.convexHull(rightEye)
        cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
        cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
```

Figure 9.Computing convex hull for eyes

```
        # 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 COUNTER >= EYE_AR_CONSEC_FRAMES:
                TOTAL += 1
            # reset the eye frame counter
            COUNTER = 0
```

Figure 10.Detect links

<u>Alerting the user:</u>

```
now = datetime.datetime.now().minute
no_of_min = now - before
print(no_of_min, before, now)
blinks = no_of_min * eye_thresh
```

Figure 11.Calculate the average number of blinks

```
if(TOTAL < blinks - eye_thresh):
    playaudio("Take rest for a while as your blink count is less than averag
    popupmsg("Take rest for a while!!!! :D")
    cv2.putText(frame, "Take rest for a while!!!! :D", (70, 150),
    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
elif(TOTAL > blinks + eye_thresh):
    playaudio("Take rest for a while as your blink count is more than averag
    popupmsg("Take rest for a while!!!! :D")
    cv2.putText(frame, "take rest for a while!!!! :D ", (70, 150),
    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
```

Figure 12. Initiate the alarm and popup message

Display the result:

```
cv2.putText(frame, "Blinks: {}".format(TOTAL), (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
```

```
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF
    if key == ord("q"):
        break

cv2.destroyAllWindows()
vs.stop()
```

Figure 13.putText function

Python Code:

```
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
```

```
def playaudio(text):
    speech = gTTS(text)
    print(type(speech))
    speech.save("../output1.mp3")
    playsound("../output1.mp3")
    return
LARGE_FONT =  ("Verdana",12)
NORM_FONT =  ("Helvetica",10)
LARGE_FONT =  ("Helvetica",8)
def popupmsg(msg):
    popup = tk. Tk()
    popup.wm_title("Urgent")
    style = ttk.Style(popup)
    style.theme_use('classic')
    style.configure('Test.TLabel', background= 'aqua')
    label = ttk.Label (popup, text=msg,style= 'Test.TLabel')
    label.pack(side="top", fill="x", pady=10)
    B1 = ttk.Button (popup, text="Okay", command = popup.destroy).pack()
    popup.mainloop()
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
# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
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())
def eye_blink():
    EYE_AR_THRESH = 0.3
    EYE_AR_CONSEC_FRAMES = 3
    COUNTER = 0
    TOTAL = 0
    print("[INFO] loading facial landmark predictor...")
    detector = dlib.get_frontal_face_detector()
    predictor = dlib.shape_predictor(args['shape_predictor'])
    #predictor =dlib.shape_predictor(args['shape_predictor'])
    #predictor = dlib.shape-predictor(args['shape-predictor'])
    print(type(predictor),predictor)
    (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
    (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
     eye_thresh = 10
    before = datetime.datetime.now().minute
    if not args.get("video", False):
        print("[INFO] starting video stream..")
        vs = VideoStream(src = 0).start()
        time.sleep(1.0)
    else:
        print("[INFO] Opening video file...")
        vs = cv2.VideoCapture(args["video"])
```

```
    time.sleep(1.0)
while True:
    frame = vs.read()
    if frame is None:
        print("unable to capture")
        break
    frame = imutils.resize(frame, width=450)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    rects = detector(gray, 0)
    for rect in rects:
        shape = predictor(gray,rect)
        if shape is None:
            print("shape predictor returning none")
            continue
        shape = face_utils.shape_to_np(shape)
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]
        leftEAR = eye_aspect_ratio(leftEye)
        rightEAR = eye_aspect_ratio(rightEye)
        ear =(leftEAR + rightEAR) / 2.0
        leftEyeHull = cv2.convexHull(leftEye)
        rightEyeHull = cv2.convexHull(rightEye)
        cv2.drawContours(frame, [leftEyeHull], -1, (0,255,0),1)
        cv2.drawContours(frame, [rightEyeHull], -1, (0,255,0),1)
        if ear < EYE_AR_THRESH:
            COUNTER+= 1
        else:
            if COUNTER >= EYE_AR_CONSEC_FRAMES:
                TOTAL+= 1
```

```
  COUNTER = 0
 now = datetime.datetime().now().minute
 no_of_min = now - before
 print(no_of_min, before, now)
 blinks = no_of_min * eye_thresh
 if(TOTAL < blinks-eye_thresh):
   playsound("Take rest for a while as yourblink count is less than average")
   popupmsg("Take rest for a while!!!!!! :D")
   cv2.putText(frame, "Take rest for a while!!!!! :D",
(70,150),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,0,255),2)
 elif (TOTAL > blinks + eye_thresh):
   playsound("Take rest for a while as yourblink count is more than average")
   popupmsg("Take rest for a while!!!!!! :D")
   cv2.putText(frame, "Take rest for a while!!!!! :D",
(70,150),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,0,255),2)
 cv2.putText(frame, "Blanks:
{}".format(TOTAL),(10,30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,0,255),2)
 cv2.putText(frame, "Ear: {:.2f}".format(ear),(300,30),cv2.FONT_HERSHEY_SIMPLEX,
0.7, (0,0,255),2)
 cv2.imshow("Frame", frame)
 key = cv2.waitkey(1) & 0xff
 if key == ord('q'):
   break
cv2.destroyAllWindows()
vs.stop()
```

App.py:

```
from flask import Flask, render_template,request
from app_mg import eye_blink
app = Flask(__name__)
@app.route('/', methods=['GET','POST'])
def index():
    if request.method == 'POST':
        eye_blink()
        return render_template('index.html')
    return render_template('index.html')
if __name__=="__main__":
    app.run(debug=True)
```

Results:

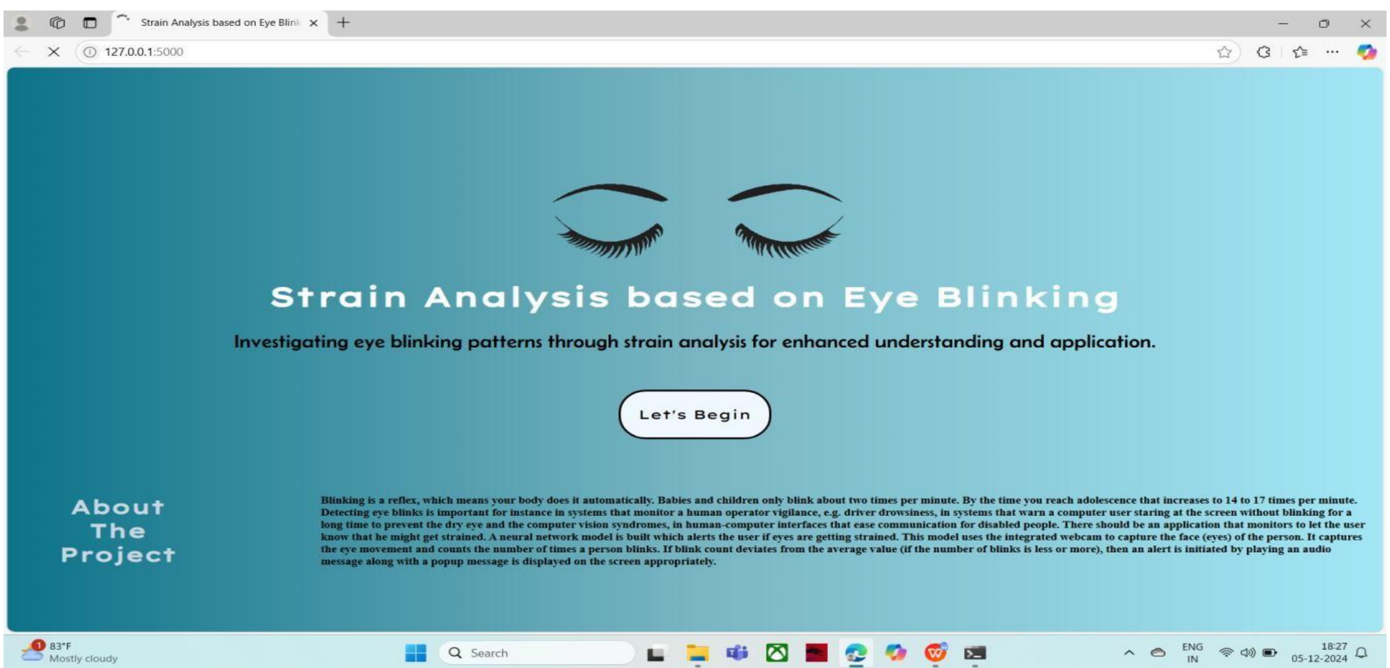
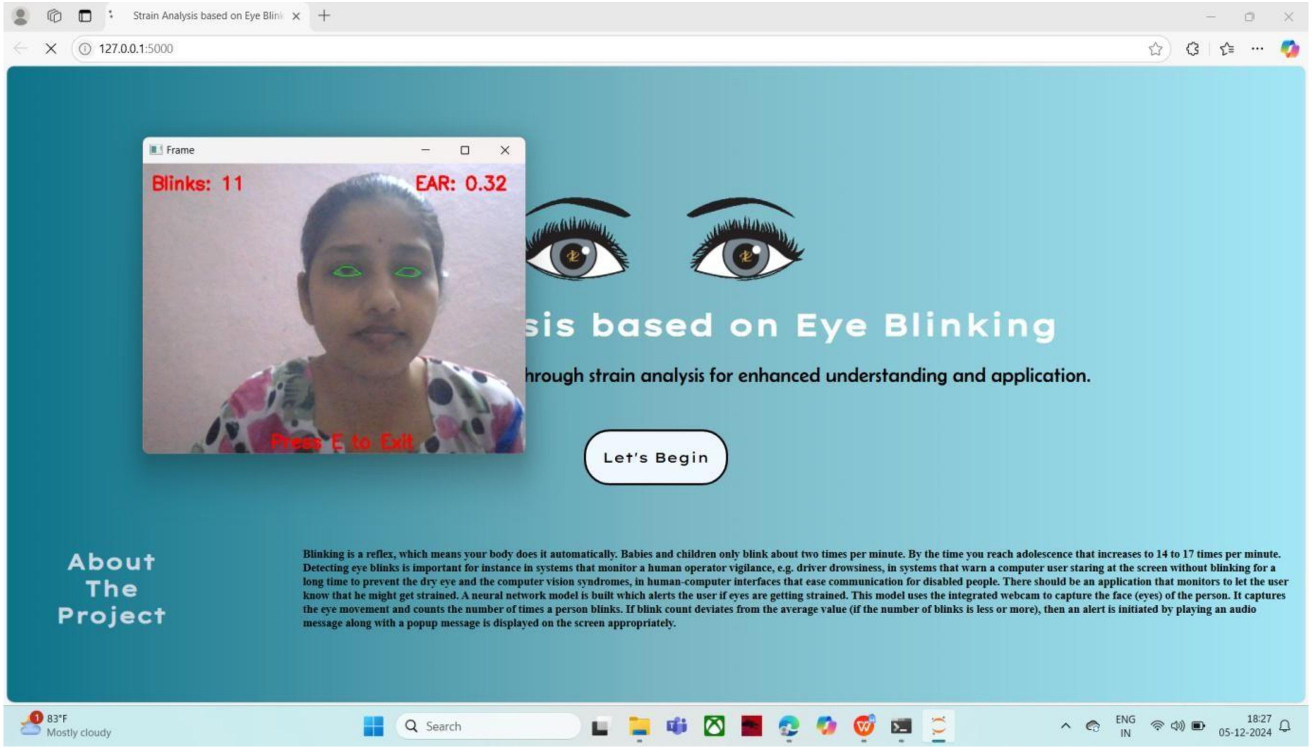


Figure 14: Home Page of Strain analysis based on eye blinking



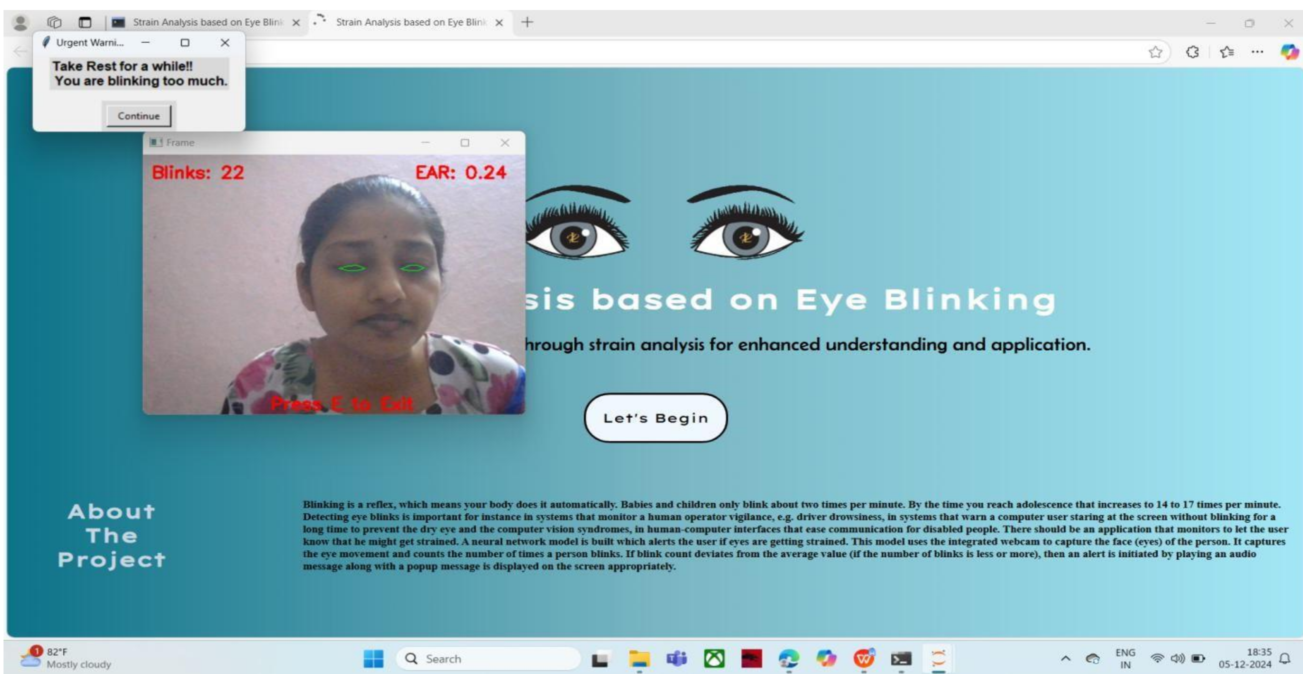


Figure 15: Final output images

3. CONCLUSION

The conclusion of a strain analysis project based on eye blinking can be summarized as follows:

1.Understanding Eye Blinking Mechanism: Eye blinking plays a vital role in the mechanics of the human face. It involves complex muscle and tissue movements that can be analyzed using strain gauges or other sensor technologies to measure deformation and strain in the facial muscles.

2.Strain Measurements: Through the use of strain analysis techniques, such as strain gauges, the project likely provided insights into the magnitude and distribution of strain during blinking. These measurements help in understanding how muscles like the orbicularis oculi contract and expand during each blink, offering a precise view of the biomechanics involved.

3. Data Interpretation: The collected data can be used to establish a correlation between the strain measurements and the frequency or intensity of blinking. It can also reveal information about the efficiency of the muscle contraction, variations in blinking patterns, and possible discrepancies in muscle performance under different conditions.

4.Applications: The results of this strain analysis could have multiple practical applications, such as improving the design of prosthetics, aiding in medical diagnostics (e.g., facial paralysis), or enhancing the development of eye-tracking technologies for human computer interaction or virtual reality applications.

5.Conclusion : The project successfully demonstrated the potential of strain analysis in studying the biomechanics of eye blinking. Further research could focus on refining sensor technologies, expanding the sample size, and exploring additional factors (such as fatigue or age-related changes) that influence eye blinking and facial muscle strain. In summary, the strain analysis of eye blinking provides valuable insights into the biomechanical behavior of facial muscles and can serve as a foundation for future research and technological advancements in related fields.

4. APPLICATIONS

Applications of Eye Blink–Based Strain Analysis

1. **Digital Eye Strain Detection for IT Professionals**
    - Helps monitor fatigue caused by long screen time in software engineers, designers, and office workers.
    - Issues alerts to take breaks and reduce eye strain.
2. **Driver Drowsiness Detection Systems**
    - Can be integrated into vehicle safety systems to detect sleepiness or reduced alertness based on eye blink patterns.
    - Prevents accidents by triggering early warnings.
3. **Healthcare and Remote Patient Monitoring**
    - Useful in monitoring patients with neurological disorders or fatigue symptoms.
    - Tracks eye blink frequency for medical research or diagnosis.
4. **Human-Computer Interaction (HCI)**
    - Enhances accessibility by using blinks as input controls for physically challenged users.
    - Provides fatigue-based feedback for adaptive systems.

5. **E-Learning & Online Education**
   - Monitors student engagement during long virtual sessions.
   - Detects signs of fatigue and encourages breaks to maintain focus.
6. **Gaming and VR Applications**
   - Integrates with VR/AR systems to monitor strain and maintain user comfort.
   - Can adjust brightness or pause gameplay if fatigue is detected.
7. **Workplace Wellness Programs**
   - Helps companies implement ergonomic solutions by tracking employee strain through blink patterns.
   - Promotes health and productivity.
8. **Research in Cognitive Workload and Mental Fatigue**
   - Used in psychological and cognitive studies to understand fatigue levels based on eye behavior.
   - Aids in improving user experience and interface design.

5. ADVANTAGES

1. Non-Invasive: Eye blinking-based strain analysis is completely non-invasive, making it a safer alternative to other methods like sensors attached to the skin or invasive procedures.

2. Easy to Implement: The technology to track eye blinking (such as cameras, infrared sensors, or eye-tracking devices) is relatively accessible, and the software for strain analysis can be programmed with existing algorithms.

3. Real-Time Monitoring: This approach allows for real-time monitoring of strain or stress responses through eye blink frequency, duration, or pattern, which can provide immediate feedback on a person's condition.

4.Continuous Data Collection: Eye blinking can be continuously observed without requiring direct involvement from the subject, making it suitable for long-term studies or monitoring without disruption.

5. Stress or Fatigue Detection: This method can potentially be used for detecting emotional or physical stress, fatigue, or cognitive load by analyzing blinking patterns, which are often correlated with these states.

6. Accessibility: Eye tracking and blink detection devices can be integrated into smartphones or wearables, providing an accessible method for strain analysis without specialized equipment.

6. DISADVANTAGES

1.Limited Specificity: Eye blinking patterns can be influenced by various factors, such as emotional state, fatigue, or environmental conditions, which might make it difficult to isolate strain or stress from other variables.

2.External Factors: Factors such as lighting, individual differences (e.g., dry eyes, age), and camera quality can affect the accuracy of eye blinking detection, leading to unreliable results.

3.Data Interpretation Complexity: The interpretation of blinking patterns might require advanced algorithms and significant data processing, making it difficult to derive accurate strain values without robust models.

4. Not Suitable for Severe Strain: Eye blinking may not be a sensitive indicator of severe physical strain or injury, where other methods such as physiological sensors (heart rate, blood pressure) might be more effective.

5.Limited User Awareness: People may not always be aware of subtle changes in their blinking patterns, meaning that strain-related insights may not be consciously recognized, limiting their practical use.

6.Subjectivity and Variability: Blinking patterns vary significantly between individuals due to genetics, age, and other personal factors. This makes it harder to create a standardized model for strain analysis that works universally.

7. FUTURE SCOPE

1.Wearable Health Monitoring Devices- Strain analysis from eye blinking can be integrated into wearable devices, such as smart glasses or contact lenses, to monitor the health and wellbeing of individuals. The strain on the eye muscles during blinking may provide valuable data for detecting conditions like eye fatigue, stress, or even neurological disorders such as Parkinson's disease.

2.Early Diagnosis of Neurological Conditions- The analysis of eye muscle strain can help in diagnosing early signs of neurological conditions, including disorders related to motor control like Parkinson's disease or multiple sclerosis. Changes in blinking patterns and strain can serve as indicators for abnormal neurological function.

3.Eye Tracking for Human-Computer Interaction- Strain analysis can be applied to improve eye-tracking systems for more intuitive human computer interactions. For instance, devices that detect strain during blinking or other eye movements can be used for controlling virtual reality (VR) or augmented reality (AR) environments, enabling hands-free control and enhanced user experiences.

4.Fatigue and Stress Monitoring- Eye blinking patterns are affected by mental and physical fatigue. Strain analysis can be used to monitor stress levels and provide real-time feedback to users, which could be useful in applications ranging from high-stress professions to driver fatigue detection in automotive systems.

5.Personalized Eye Care- By monitoring strain and changes in eye muscle behavior, eye care professionals could create more personalized treatment plans for conditions like dry eyes, strabismus (crossed eyes), or blepharospasm (involuntary blinking). This can lead to improved therapies and the development of devices for more effective rehabilitation.

6.Artificial Intelligence and Machine Learning- The integration of AI and machine learning algorithms with strain analysis from eye blinking can open up new avenues for predictive modeling and automated diagnostics. By training models on large datasets of eye strain patterns, AI could accurately predict health issues, personalize treatments, and even predict moments of fatigue in users.

8. BIBLIOGRAPHY

1.Tereza Soukupova and Jan Cech.

Real-Time Eye Blink Detection using Facial Landmarks.

Center for Machine Perception, Czech Technical University, 2016.

https://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf

2.Kazemi, Vahid, and Josephine Sullivan.

One Millisecond Face Alignment with an Ensemble of Regression Trees.

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

https://doi.org/10.1109/CVPR.2014.241

3.OpenCV Documentation.

Open Source Computer Vision Library.

https://docs.opencv.org/

4.Dlib C++ Library by Davis E. King.

Machine Learning Library for Facial Landmark Detection.

http://dlib.net/

5.Satya Mallick.

Eye Blink Detection with OpenCV, Python, and dlib.

LearnOpenCV.com, 2017.

https://learnopencv.com/eye-blink-detection-using-opencv-python-and-dlib/

6.Python Software Foundation.

Python 3.11 Documentation.

https://docs.python.org/3/

7.Flask Documentation.

Flask Web Development Microframework.

https://flask.palletsprojects.com/

8.Shervin Emami.

Real-Time Eye Detection and Tracking Using OpenCV.

http://www.shervinemami.info/

9. HELP FILE

PROJECT EXECUTION:

STEP-1: Launch **GOOGLE COLAB** from your browser.

STEP-2: Open the code file named eye_blink_strain_analysis.py.

STEP-3: Import all required libraries such as: cv2 (OpenCV), dlib, imutils, scipy, pyttsx3 or playsound for alerts.Check for any import errors.

STEP-4: Create a folder named EYE_BLINK_PROJECT on your **DESKTOP**.

STEP-5: Create home.html to display the project home page with instructions to start monitoring.

STEP-6: Launch **Visual Studio Code (VS Code)**.

STEP-7: Open **Spyder**, set the path of eye_blink_strain_analysis.py, and run the file.

STEP-8: Run app.py (Flask backend) to generate a local URL like localhost:5000.

STEP-9: Copy and paste the URL into your web browser.

STEP-10: The home page will appear.

STEP-11: Click **"Start Monitoring"** to begin real-time analysis.

STEP-12: The system will:

• Detect face and eye landmarks

• Calculate eye dimensions and EAR

• Monitor blinking patterns

• Display blink count on screen

• Trigger alerts (audio + popup) if strain is detected

STEP-13: The system helps detect eye strain and suggests screen breaks to prevent fatigue.