Cairo University
Faculty of Engineering

Credit Hour system

Project Proposal

Real-Time Eye State Analyzer

Presented By:

1. Youssef Ahmed Abdelmoneim *(1220211)*
2. Youssef Mohamed Abdelfatah *(1220299)*
3. Hana Ahmed Foad *(1220289)*
4. Mazen Mohamed Sayed *(1220018)*

Project Overview: This project aims to develop a computer vision application that analyzes facial images in real-time to assess eye condition. The system will detect potential irritation (redness) and estimate fatigue (brightness, closure) by processing video or static images. The entire pipeline will be built using classical image processing techniques, primarily from the OpenCV library, focusing on color space analysis and feature detection without relying on deep learning models.

Key Objectives:

1. **Automated Eye Detection:** Reliably locate and isolate eye regions from a given facial image.
2. **Redness Quantification:** Analyze the sclera (white part) of the eye to quantify the level of redness and classify it as "Normal" or "Irritated."
3. **Fatigue Estimation:** Analyze eye brightness and aperture to provide a basic estimation of user fatigue or poor lighting conditions.
4. **Symmetry Analysis:** Compare the metrics (redness, brightness) between both eyes to detect potential anomalies or asymmetrical conditions.
5. **Real-Time Classification:** Provide an immediate, frame-by-frame visual classification for the user.

Methodology & Project Pipeline

The system will operate in a sequential pipeline, as detailed below:

1. Image Acquisition

- **Source:** The system will be designed to work with static images (e.g., JPEG, PNG).
- **Assumptions:** For optimal performance, the system assumes moderate, even lighting and a generally forward-facing subject.

2. Eye Region Detection (ROI Isolation)

- **Technique:** A pre-trained Haar Cascade classifier (haarcascade_eye.xml or haarcascade_frontalface_default.xml followed by haarcascade_eye.xml) will be used.
- **Process:**
    1. The input image is converted to grayscale.
    2. The Haar classifier's detectMultiScale function is called to find bounding boxes for the face and then the eyes.
    3. These bounding boxes (Regions of Interest, or ROIs) are cropped from the main image, effectively isolating the left and right eyes for individual analysis.

3. Color Space Conversion

- **Technique:** The cropped RGB eye ROIs are converted to the **HSV (Hue, Saturation, Value)** color space.
- **Rationale:** The HSV space is ideal for this task as it separates color (Hue) from its intensity (Value) and purity (Saturation). Redness is a specific *hue*, making it easier to isolate than in the RGB space where light and shadow heavily influence the R, G, and B

values.

4. Redness Analysis

- **Technique:** Color Thresholding and Ratio Calculation.
- **Process:**
    1. **Masking:** A band threshold is applied to the **Hue channel**. Since red exists at both the beginning and end of the hue spectrum (0-10 and 170-180 in OpenCV), a dual-range mask is created. A secondary threshold on the **Saturation channel** (e.g., S > 50) is applied to filter out desaturated, non-relevant pixels (like pale skin or gray tones).
    2. **Ratio Calculation:** The system computes the Redness Ratio for the sclera region (note: a more advanced implementation would first mask out the iris/pupil).
    `Redness Ratio = (Number of "Red" Pixels) / (Total Pixels in Eye ROI)`
    3. **Classification:** This ratio is compared against a pre-defined threshold to classify the eye as "Normal" or "Irritated."

5. Brightness and Fatigue Estimation

- **Technique:** Grayscale Value Analysis.
- **Process:**
    1. The eye ROI is converted to grayscale.
    2. The average pixel intensity (mean value) of the entire ROI is calculated.
    3. This average brightness is classified into categories:
        - **Normal Lighting:** Brightness is within an expected range.
        - **Low Light / Fatigue:** Brightness is significantly low. This can indicate that the eyelids are partially or fully closed (fatigue) or that the ambient lighting is poor.

6. Symmetry Comparison

- **Technique:** Metric Comparison.
- **Process:**
    1. When two eyes are successfully detected, the Redness Ratio and Average Brightness are calculated for each.
    2. The absolute difference between the left and right eye metrics is computed.
    3. If abs(Left_Redness - Right_Redness) or abs(Left_Brightness - Right_Brightness) exceeds a certain tolerance, a "Symmetry Warning" is flagged. This can indicate uneven lighting or a localized issue in one eye.

7. Classification and Display

- **Output:** The final processed video stream or image is displayed to the user.
- **Visualization:** Text labels are overlaid on the bounding boxes for each eye, providing a clear, color-coded classification:
    - 🟢 **Normal**

    - 🔴 **Red Eye / Irritated**
    - 🟡 **Low Light / Fatigue**

Technologies to be Used

- **Programming Language:** Python 3.x
- **Core Library:** OpenCV (opencv-python) for all image processing tasks (Haar cascades, color space conversion, thresholding, contour detection).
- **Numerical Operations:** NumPy for efficient array manipulation and calculations.

Potential Applications

- **Digital Wellness:** A simple tool to remind office workers or students to take a break if signs of eye strain are detected.
- **Driver Safety:** A basic component within a larger driver-assist system to provide warnings for drowsiness.
- **Telehealth:** A preliminary check for users to self-assess eye irritation before a virtual consultation.