Detecting a Driver's Drowsiness/Distraction using Computer Vision

EE626 Course Project

Contributors

- Ameya Madhekar, 170121004, ameya170121004@iitg.ac.in
- Arvind Patidar, 170122005, arvin170122005@iitg.ac.in
- Ashish Bakoliya, 170122008, ashis170122008@iitg.ac.in

The Problem

According to a study done by Central Road Research Institute (CRRI), about **40%** of the **total road accidents** that take place on Indian highways & major roads are due to **sleep-deprived**, **fatigued drivers** who **doze off at the wheel**.

According to a report, **texting while driving** is **6x more likely** to cause an accident than driving drunk. In the US, **1 out of every 4 car accidents** is caused by **texting and driving** at the same time. A new report released by the World Health Organisation (WHO) states that drivers who use mobile phones are **four times more likely to crash**.

Aim

The aim of the project is to address the problem of **sleepy or distracted driver** to help **reduce the number of road accidents**.

We use Computer Vision to **detect the eye blinks** of the driver from the live video stream input given by the camera placed in front of the driver.

If the eyes are closed for a certain amount of time, we claim that the driver is **distracted or facing drowsiness** and alert the driver by playing an alarm.

Algorithm

1. **Detect the driver’s face** in the video stream.
2. If a face is found, we apply **facial landmark detection** to identify the **eye regions** from the face and extract them.
3. We then calculate the **Eye Aspect Ratio (EAR)** from the extracted eye regions to determine if the eyes are closed or not.
4. If the the EAR is **below a threshold** (i.e., eyes are closed) for a **certain amount of time**, we **sound the alarm**.

1. Face Detection

dlib [3] is a **toolkit** for making real world machine learning and data analysis applications in C++. It has been widely used for its **face detection** and **facial landmark detection** models.

For detecting the driver’s face, we load the face detector model that comes inbuilt with the dlib library. The model is based on **Histograms of Oriented Gradients (HOG) & linear SVM** for human face detection [2].

For each frame that we receive from the video stream, we first resize it to a width of 450 pixels, convert it to grayscale and then apply the dlib’s face detector to **find & locate the face** in the preprocessed image.

1. Face Detection

This widely used model is built out of **5 HOG [2] filters** – front looking, left looking, right looking, front looking but rotated left, and a front looking but rotated right. The model comes embedded in the header file of dlib itself.

The dataset [6] used for training the model consists of **2825 images** which are obtained from **LFW [7] dataset** and **manually annotated** by Davis King, the author of dlib.

We choose this model for face detection as it is the **fastest method** on CPU, works very well for **frontal and slightly non-frontal faces**, a **light-weight model** compared to others, and works well **under small occlusion**.

2. Facial Landmark Detection

Again, we use the dlib library [3] along with its **pretrained facial landmark detection model** [5] and apply it on the face extracted from the face detector to get the landmark points.

The model essentially tries to **localize and label the following facial regions**: Mouth, Right eyebrow, Left eyebrow, Right eye, Left eye, Nose, Jaw.

The model is able to estimate the **locations of 68 coordinates** (x, y) or **landmark points** that map the facial points on a person’s face. We then **extract eye regions** with simple array slicing to get the (x, y)-coordinates of both the eyes.

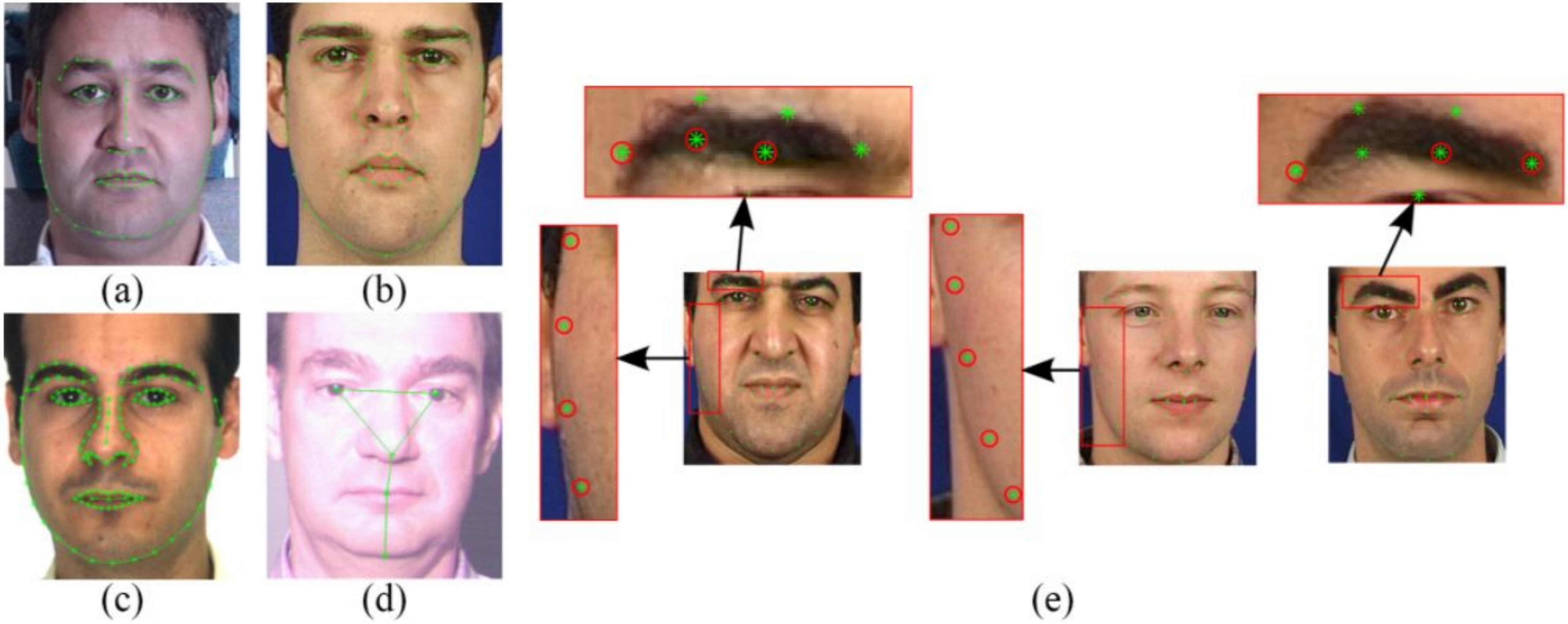
2. Facial Landmark Detection

The model is an implementation of the “One Millisecond Face Alignment with an Ensemble of Regression Trees” paper [8].

It is trained on the **iBUG300-W dataset** [4], consisting of **labeled facial landmarks** on images from standard face datasets. The dataset was created using a semi-automatic annotation methodology for annotating massive face datasets. These images specify the exact (x, y) coordinates of regions surrounding each facial structure.

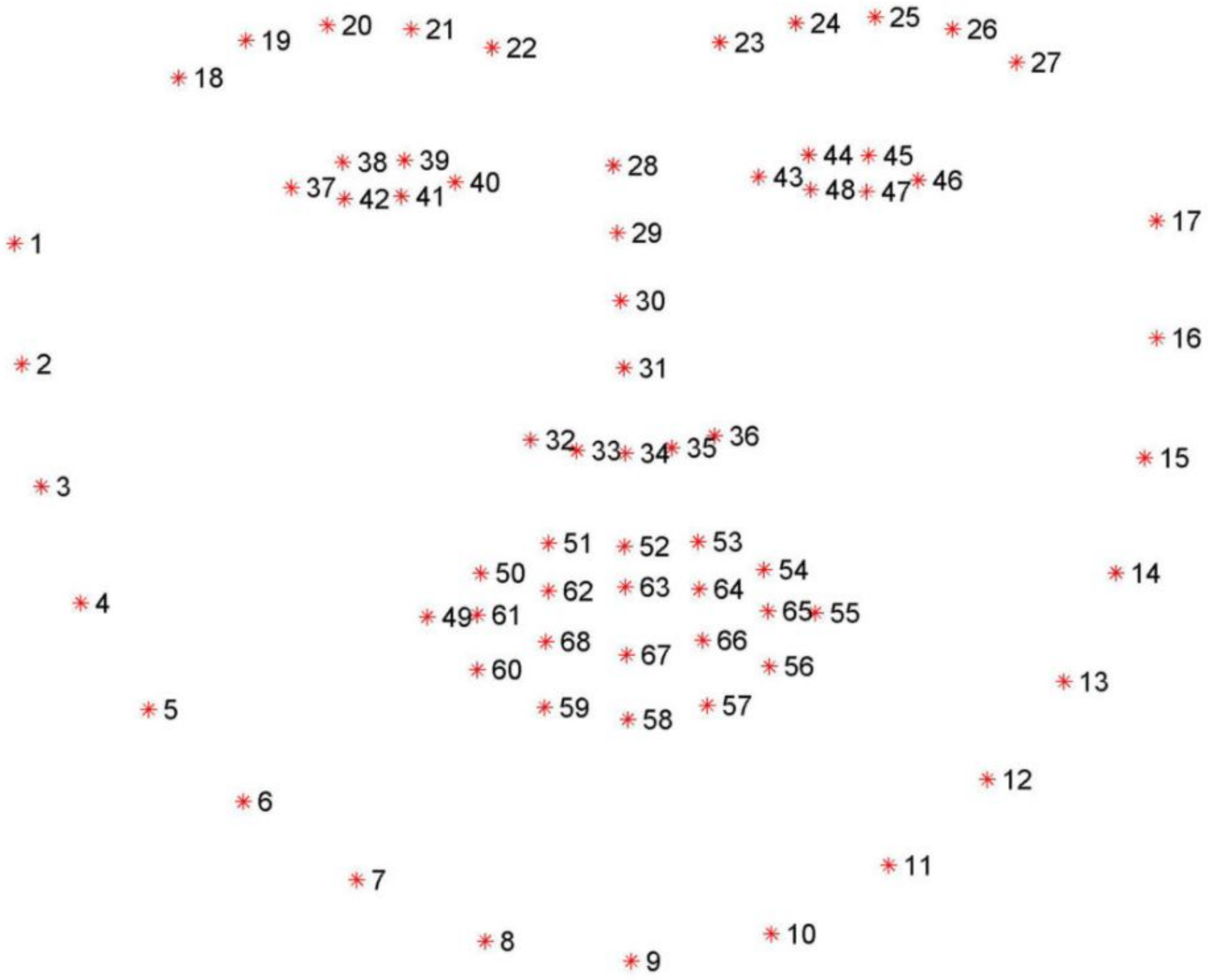
An **ensemble of regression trees** are trained to estimate the facial landmark positions directly from the pixel intensities themselves (i.e., no “feature extraction” is taking place).

The end result is a facial landmark detector that can be used to **detect facial landmarks in real-time with high quality predictions**.



[4] Sample annotated face images from the iBUG300-W dataset.

[3, 4] The 68 landmark points of a face as estimated by the dlib's facial landmark detection model.

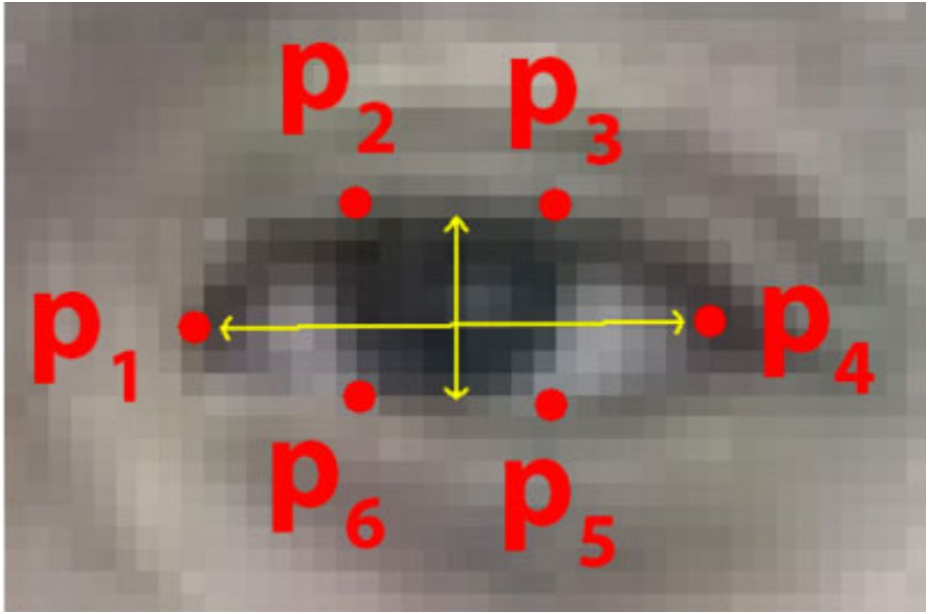


3. Eye Aspect Ratio (EAR)

Now that we have the array of coordinates of both the eyes, we can find the EAR.

Each eye is represented by **6 (x, y)-coordinates**, starting at the left-corner of the eye and then working clockwise around the remainder of the region.

We now calculate the Ratio using these 6 coordinates as shown in the next slide.



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

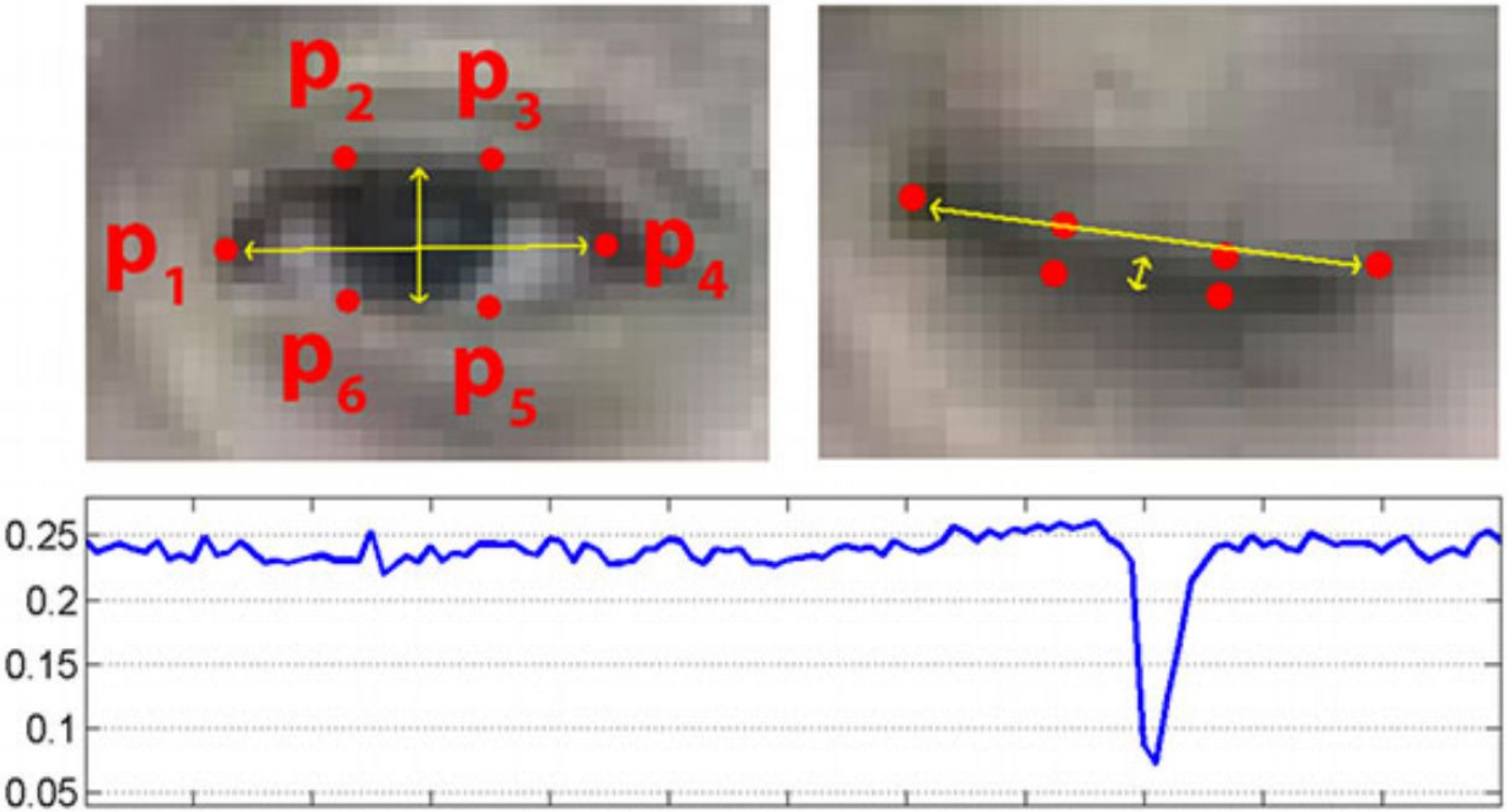
[1] As shown in the above left figure, an eye is represented with 6 points- pi, where i=1,..,6. Hence we have **two sets of vertical distances** (||p2-p6|| & ||p3-p5||) and **one set of horizontal distance** (||p1-p4||). EAR is simply the **ratio of the average of vertical distances to the horizontal distance** as shown in the above right fig.

3. Eye Aspect Ratio (EAR)

The eye aspect ratio is approximately **constant while the eye is open**, as there is no change in the horizontal & vertical distances.

But when the eye is closed, the vertical distance becomes almost zero while the horizontal distance remains the same. Hence, the **ratio falls rapidly to almost zero** when a blink is taking place. And rises back to its original value when blink is over.

We declare a constant EAR_THRESH as the threshold value. If the EAR **falls below this threshold value**, we determine that the person has blinked. We set EAR_THRESH = 0.3.



[1] Variations in the EAR as the eye is closed or opened. The EAR remains almost constant when the eye is open. When the eye is closed, there is a sudden dip in EAR value (almost 0). It rises again when eye is opened.

4. Drowsiness/Distraction Detection

Once we detect a blink with the help of EAR, we start counting the **number of consecutive frames** from the live video stream for which the **ratio stays below the threshold value** of EAR_THRESH.

This will help us know whether the eyes are **closed continuously for some amount of time** or whether it is **just a blink** for a few hundred milliseconds (a blink lasts for about 0.1 to 0.4 seconds).

We declare another constant EAR_CONSEC_FRAMES as the threshold value. If the EAR stays below EAR_THRESH for **more than EAR_CONSEC_FRAMES number of frames**, we determine that the person is drowsy or distracted and **sound the alarm** to alert the driver. We set EAR_CONSEC_FRAMES = 48.

Tech Stack used

Language: Python 3.7

Libraries: dlib, SciPy, imutils, threading, numpy, imutils, opencv

Platform: Visual Studio Code

References

1. Soukupová, Tereza and Jan Cech. “Real-Time Eye Blink Detection using Facial Landmarks.” (2016).
2. N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 2005, pp. 886-893 vol. 1, doi: 10.1109/CVPR.2005.177.
3. https://github.com/davisking/dlib, https://www.jmlr.org/papers/volume10/king09a/king09a.pdf, http://dlib.net/
4. C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, M. Pantic. 300 Faces in-the-Wild Challenge: The first facial landmark localization Challenge. Proceedings of IEEE Int’l Conf. on Computer Vision (ICCV-W), 300 Faces in-the-Wild Challenge (300-W). Sydney, Australia, December 2013. https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/
5. https://github.com/italojs/facial-landmarks-recognition/blob/master/shape_predictor_68_face_landmarks.dat
6. http://dlib.net/files/data/dlib_face_detector_training_data.tar.gz
7. Huang, Gary & Mattar, Marwan & Berg, Tamara & Learned-Miller, Eric. (2008). Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. Tech. rep..
8. V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1867-1874, doi: 10.1109/CVPR.2014.241.