# Facial Emotion Recognition and Detection Application

Ioana NAGIT, Andreea-Ramona OLTEANU, Ion LUNGU
The Bucharest University Of Economic Studies, Bucharest, Romania
ivana.nagit@gmail.com; ion.lungu@ie.ase.ro

*The purpose of the present paper is to study the process of face detection and recognition, followed by the ability to detect the drowsiness state of an individual. This experiment has been part of the bachelor thesis and aims to highlight some of the true power of Artificial Intelligence* [1]. *For a full perspective on the topic, an experiment was held in order to emphasize the flexibility and power of facial detection. It aims to analyze the real-time possible drowsiness of a driver by using a key point facial landmark detection and warn the individual when necessary.*

**Keywords**: *Artificial Intelligence, Face Recognition, Facial Landmarks, Eye Blink Detection*

## Introduction

Acknowledging human gestures and instinctual impulses have always captured the attention of analysts because the ability to peruse one's specific gesture helps to amend the human emotional quotient in emotional-computer interaction. To expound the setting of human developments, there are some perspectives that can be taken into consideration, such as voice tone or facial expressions.

The psychological theory of Emotional Intelligence has its origin way back in 1997, when it was developed by the American psychologists Peter Salovey and John D. Mayer [2]. Also known as EI or EQ, the emotional quotient is the individual's ability to recognize one's own emotions but also those of others, to distinguish between feelings and to correctly identify them. All the information received guides the individual's way of thinking and interacting in order to adapt to the environment.

Five main components comprise EI [3], and those are:

- Self-awareness – capacity to recognize and get individual temperaments and feelings, as well as their impact on others;
- Self-regulation - capacity to control or divert troublesome moods and the affinity to think before acting;
- Internal motivation – an affinity to seek after certain objectives with vitality and perseverance;
- Empathy – part of social awareness representing the capacity to understand the temperament of other people;
- Social skills – capability of overseeing connections and building systems [4].

When the English mathematician Alan Turing came up in 1950 with his then mocked idea which supported that machines could be determined to think similarly to a human [5], the concept of Artificial Intelligence first appeared as a possible conceivable reality. His paper - "Computing Machinery and Intelligence" [4] - created some sort of opposition to decide whether this super-new notion is even feasible. The question was on everyone's lips: Could a machine comply with the emotional intelligence of a human being and act accordingly? It took a while for individuals to recognize the genuine control of AI and in 1956, computer and cognitive scientis John McCarthy held the first ever academic conference on this new, controversial subject.

The field of Artificial Intelligence is wide and usually seems to surpass reality. From Turing's Test [6], which assumes that a computer can actually think and give responses similarly to a human, to the inception of the same test and even the contradiction which states that a machine could never invoke an emotional response or originate anything, this marks only the beginning of what now has become a continuous evolution of technology.

A very discussed topic in the field of AI is the Facial Emotion Recognition (FER). Facial Emotion Recognition is a thriving subject of research in which the Human-Robot Interaction is based on evaluating, developing, and planning interactional environments for smart systems to form cognitive and emotional interaction with the help of a few communication channels that link humans to robots. Cognitive scientists keep trying to discover more accurate methods of recognizing facial human expressions and their gestures.

Being able to get the emotions of humans with high accuracy represents a huge advantage in numerous settings, such as monitoring the reactions and feelings of people focused on performing certain tasks or watching an ad. It is important to be able to distinguish between the states of a human because this can be further used in developing applications that may help people struggling with disorders such as dementia and autism spectrum. It might also contribute to building an emotional report of an employee/student or maintaining an accurate profile of a patient that needs to be kept under observation.

## 1. Drowsiness and eye blink detection

Drowsiness state is common among people, especially for drivers that need to stay awake for a long period of time. It is also known as the main cause of car accidents, because driving while being drowsy is very similar to driving under the influence of alcohol, thus it increases the chances to be involved in a car crash by three times. According to the National Sleep Foundation [8], staying awake for 18 hours straight makes the equivalent of a blood-alcohol concentration of 0.05%, while 24 hours of no sleep equals a blood-alcohol concentration of 0.10%.

Among youthful drivers, driving exhausted is quite common due to way of life components. Young people need more rest than grown-ups; weariness may influence youngsters more than grown-ups. Most proficient drivers should adapt to fatigued driving on a visit premise due to work-related components. Approximately half of all professional drivers have less than typical rest time before embarking on a long-distance trip.

Drowsiness is a very discussed topic at the moment, and a robust motivation came from trying to explore the vast world of artificial intelligence and get a glimpse of what it is really like to connect on a whole different level with deep learning. Another incentive was the analogy with the continuous desire of doing multiple things at once instead of taking turns and giving all of the attention to the immediate task that needs to be completed. Moreover, developing two different applications, an Android and a Python one, respectively, showed the flexibility, huge adaptability, and ease of implementing something that acts in both cases as an alert system.

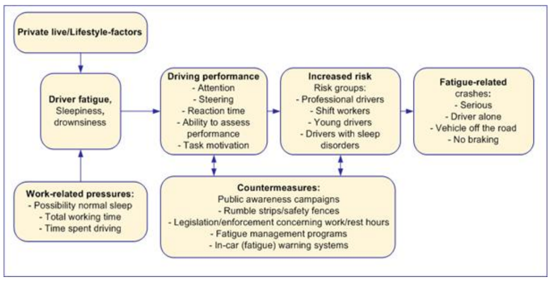

Private live/Lifestyle-factors
Driver fatigue, Sleepiness, drownsiness
Driving performance - Attention - Steering - Reaction time - Ability to assess performance - Task motivation
Increased risk Risk groups: - Professional drivers - Shift workers - Young drivers - Drivers with sleep disorders
Fatigue-related crashes: - Serious - Driver alone - Vehicle off the road - No braking
Work-related pressures: - Possibility normal sleep - Total working time - Time spent driving
Countermeasures: Public awareness campaigns - Rumble strips/safety fences - Legislation/enforcement concerning work/rest hours - Fatigue management programs - In-car (fatigue) warning systems

**Fig. 1.** Key issues regarding driver

In order to make this first experiment work properly, a method that can determine for how long an individual's eyes have been closed is needed. Therefore, if the eyes of the driver are closed for a given period of time, then an alarm will be played with the sole purpose of bringing the person back to its senses and deciding whether it is time for a nap or if he was just a little distracted. Because, as good as it works for drowsy people, it may come in handy when staring at the phone or in any other part rather than the road while driving.

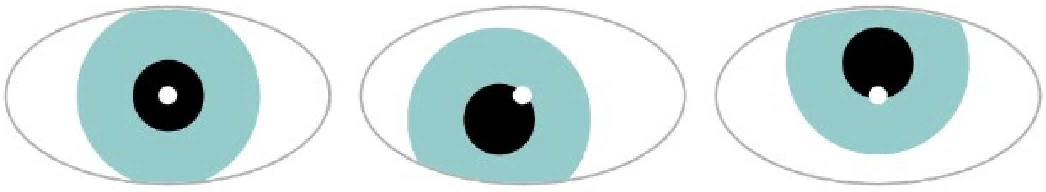


**Fig. 2**. Infrared light reflection on user's eye correlative to the pupil used to compute the direction of his gaze [6]

The general stream of the drowsiness system is very direct and requires a camera that is able to monitor the face of the driver (e.g. Microsoft LifeCam HD-3000). Once the face of the driver is found, the facial landmark detection is applied, and it extracts the region of the eyes. Further on, the eye aspect ratio is computed in order to decide whether the eyes of the individual are open or closed. An alarm
is played if the eye aspect ratio determines that the eyes have been closed for a long period of time.
Eye tracking represents the method of measuring the point where one is looking, the motion of an individual's eye correlative to the head. For more than 20 years, eye tracking has been considered an extreme innovation that helps with the analysis and detection of user interface issues.
The main focus of this method is on the points of fixation (Figure 2). More accurately, they represent specific zones in which a potential user's gaze stops long enough and lingers for a sufficient amount of time so that they can handle what they have just seen. The motion of the user's eyes between these points is also known as „*saccade*" [6].
Facial landmarks have their purpose in localizing and representing specific regions of a potential user's face such as the eyes, nose, eyebrows, and mouth [7]. It has faced a real quick development within the computer vision community since it has numerous applications. For instance, the capability to successfully detect emotion through specific facial gestures, approximating gaze direction, and even the famous face swapping is made with the help of facial landmarks.
Knowing if a person is paying attention or not, especially in systems that require emotion estimation, is done like in the figure below:

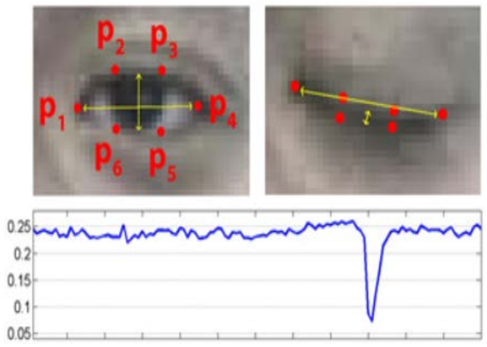


**Fig. 3.** Eye Aspect Ratio

Figure 3 highlights that every time there is a squint, the proportion between the four points within the eyes becomes smaller. When the eye is open, the EAR remains constant. When the eye is fully closed, the four points turn into a straight line and the EAR drops significantly from 0.25 to 0.05.
Nowadays there are so many applications on both personal computers and smartphones camera programs. For achieving
the desired result, the landmark detector has to find the corners of the designated area (mouth, eyes, nose) and connect those points. A lot of algorithms are implemented with the help of the OpenCV free, cross-platform library.
Basically, when talking about facial landmarks, we are actually discussing about the detection of a subset of shape prediction matter. So, the process takes two steps: localizing the face and detecting key facial structures in a specific region of

interest. Starting from this point, the developer can choose on which part of the face he wants to focus and create certain tasks.

Of course, before all this can be emphasized, a thorough face detection must be implemented, and it is usually done either by using the built-in Haar cascades of OpenCV library, by applying a pre-trained Histogram of Oriented Gradients [8] and Linear Object Detection, or by using deep learning algorithms. Because the Viola-Jones type detector [6] is a bit old and it really requires a good amount of time to just tune the parameters. Therefore, the second approach was desired and adopted for this experiment. The next subchapter will describe exactly how the process of face detection works.

This paper's focus on eye blinking detection highlights the implementation of DLib library, a modern C++ toolkit that holds machine learning calculations for making complex computer software that can genuinely solve real-world problems. Kazemi and Sullvian's paper [9] was used for implementing the facial landmark detector. This method uses a training data set of manually labeled facial landmarks on a specific image together with the likelihood of distance between sets of input pixels.

The final result may be a facial landmark finder that can be utilized to identify facial regions of interest in real-time with tall quality forecasts. The DLib library uses its pre-trained facial landmark and estimates the facial structure with the help of 68 coordinates (part of the iBUG 300-W dataset) that create the facial structure of an individual. All the coordinates can be seen there as well. They start from the right side of the face, then continue with the right eyebrow, moves to the left eyebrow, which is immediately followed by the node, left and right eye, and finally the mouth.

Using OpenCV to recognize faces means following two simple steps, which is basically detecting the face and extracting the embeddings that quantify each face in a given image. The latter mentioned step involves using a model that is a combination between Python and Torch implementation [10].

Firstly, an image or a video is given as input to the face recognition pipeline. Secondly, the face detection algorithm is applied and the location of the face in the given image is detected. Facial landmarks can optionally be computed so that an accurate alignment of the face can be achieved. Face alignment identifies the geometric structure of faces and attempts to obtain a canonical alignment based on rotation and scale. Even though it is optional, face alignment proved to give a significant increase in accuracy regarding face recognition.

Passing the input image or video through the deep neural network looks something like this:

To prepare a face recognition model by using deep learning means that each input group of information includes an anchor, a positive, and a negative image. The anchor and positive image contain the same face, while the negative image does not share the same identity. With the help of the deep neural network and the Triplet Loss Function [10], the network can assess faces and give vigorous embeddings that are suitable for face recognition.

Choosing the right method or the right solution to implement a specific remains the responsibility of the developer. It is highly recommended to start from a very specific context because deep learning is a very powerful tool, and in order for everything to work out smoothly, there needs to be a simple way of putting things together.

## 2. Solution Implementation

It is important to be aware of computer vision's field and how it is related to the need to use OpenCV. Computer Vision [11] represents a field which trains computers to transcribe and acknowledge the visual world. It is an indispensable feature for self-driving cars, photo-correction applications, and robotics.

OpenCV is the gigantic open-source library for computer vision and image handling and plays a major, vital part in today's frameworks. By utilizing it, one can handle images and videos to distinguish objects, handwriting of a human, or faces.

When it coordinates with different libraries, such as NumPy, Python is able to handle the OpenCV cluster structure for analysis. To recognize the image pattern and its different highlights, a vector space is used, and then multiple mathematical operations are performed on these highlights.

The Python based experiment begins with importing the OpenCV library together with the other required packages and libraries. The alarm function is defined, and it takes the path to the WAV audio file chosen to alert the driver. It will be called when the computed EAR is lower than the threshold so that the driver can be notified.

Next, the Euclidian distance between the sets of eye landmarks need to be computed, so it is required to create a function which does exactly that. To get a good accuracy, both vertical and horizontal Euclidian distances between the coordinates need to be computed, and then simply return the EAR result. This value stays approximately constant if the driver's eyes are open. A sudden decrease will take place during a blink, and the EAR will get a value close to zero. If the driver's eye is completely closed, then the EAR will stay constant, with a much lower value than the EAR when the eyes are open.

Each eye has six distinct landmark locations, and the counting starts from the outer corner of the eye in a clockwise motion until it is fully covered. With both eyes detected, the function can compute the EAR and further establish in what state the driver's eyes are in.

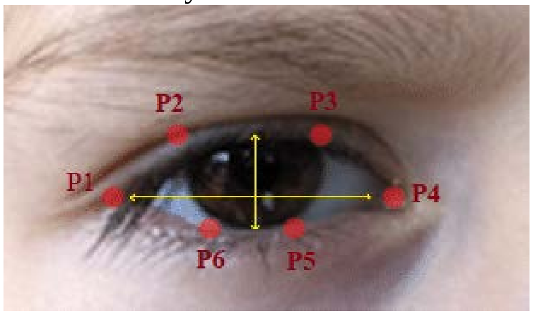


**Fig. 4.** Landmark locations [22]

In each video frame, EAR is computed by using the formula presented below, where p1-p6 represent landmark locations [15]:

$$EAR = \frac{\|p2-p6\|+\|p3-p5\|}{2*\|p1-p4\|}$$

Constructing the argument parse is also required, so the code will have three command line arguments:

- the shape predictor;
- the alarm;
- the webcam.

It is important to know that the shape predictor is really the path to the pretrained facial detector of DLib. The path to the WAV audio file is optional, but it was used in order to keep the purpose of the application alive. The last command line argument refers to the built-in webcam of the user.

The main part starts from this point on, because the next step aims to define the variables that indicate the blink and the number of consecutive frames, respectively, plus a bool variable for the alarm and a frame counter. If EAR is lower than the given threshold, then the number of frames the driver has kept his eyes closed for will be counted. Consequently, if this counter has a bigger value than the predefined number of consecutive frames, the alarm is sounded.

The input threshold was initialized with a 0.3 value for the very simple reason that it usually gives the best results. The given number of frames is 50, which means that if for 50 consecutive frames the driver's eyes are closed, then the boolean is updated and the alarm plays. Here it is up to the desired level of the application's sensitiveness. For example, if the developer does not want to hear the alarm every time he just looks down, then he needs to increase the number of consecutive frames.

It has been discussed in a previous chapter about DLib's features and the existing relation between the library and the HOG [8] that can train very accurate human detectors. Coming back to the code, the histogram was also instantiated together with the facial landmark predictor. The facial landmarks that are produced by

DLib act similarly to an indexable list, so it was needed to get the indexes of both left and right facial landmarks. Thus, the eye regions were extracted with little to no effort with the help of a slice of an array.

Instantiation of the video stream thread and giving a little time for the camera to properly warm up are next added to the code, followed by a for loop that literally loops over the frames gotten from the video stream. The next frame is read, converted to grayscale, and resized. DLib's face detector is further applied in order to find the face in the image.

Facial landmark detection must be applied next, to get all important regions of the driver's face and convert the obtained result to NumPy array. Using the array slicing described in the paragraph above extracts the exact coordinates of the driver's eyes in order to compute the EARs and get a better estimate by averaging both left and right eye ratios.

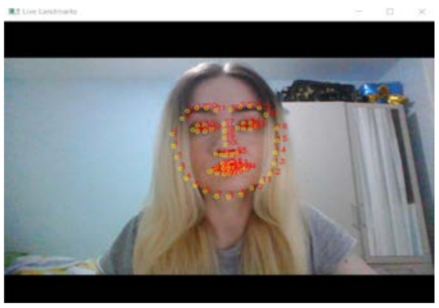
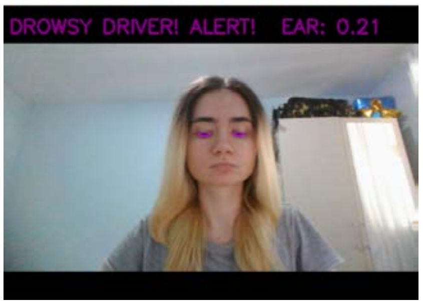


**Fig. 5.** Facial landmark coordinates

Now both eyes can be visualized after computing the convex hulls, and it is time to see if the state of the driver is drowsy or not. This is done by comparing the input threshold with the EAR. The counter is incremented each time EAR is below the threshold and when it exceeds the number of consecutive frames given as input, it is automatically assumed that the driver is getting drowsy (Figure 6).



DROWSY DRIVER! ALERT! EAR: 0.21

**Fig. 6.** Drowsy driver

On top left the message that warns the driver is displayed and on top right the computed EAR can be seen. If the driver is in a normal state, then the driver will only see on the screen the eye aspect ratio displayed.

The alarm sound is checked, and it needs a separate thread where the calling of the alarm can be done. In this way, it is ensured that the main program is not going to stop from execution. If the result of EAR has a bigger value than the input threshold, it means the driver is awake and looking at the road, so the program keeps tracking his eyes. Also, in this happy case, the alarm needs to be turned off.

Displaying the output frame concludes the implementation of the code, and an exit key is added in order to stop the whole process of drowsiness detecting.

## Conclusions

The paper "Facial Emotion Recognition and Detection Application" captures the existing trends in the world of machine learning, focusing mostly on face and eye detection. With limited resources, it is hard to compare the application with those created by corporations like Microsoft or Google. However, developing this demo came as a challenge and an unexpected surprise.

My individual contribution is emphasized by developing both applications from scratch, integrating the services offered by Python (OpenCV) and Google Cloud (Google Vision API) and finding a data set for training the neural network with the purpose to accurately identify the face and extract the region of interest, which in this case was represented by the eyes.

Possible future implementations could mean managing to expand the drowsiness detector with a yawning one and with the help of a 3D convolutional neural network to be able to detect the face and eyes, even though the camera is not positioned on the car dash.

Another major improvement could represent the capacity to turn it into a mobile-friendly app and provide certain

alternatives to the driver when the drowsiness state is detected, such as suggesting the nearest parking lot or coffee place.
Machine learning and deep learning have already become vital in our everyday life, and as dangerous and exciting as they may seem at a first glance, it takes some courage to dive into the unknown, having as weapons only your IQ, uncontrollable desire to keep adapting, and, of course, a good sense of humor.

## Bibliography

[1] I. Nagit, "Facial Emotion Recognition and Detection Application," 2020.
[2] J. D. Mayer, " Annual Review of Psychology," in *Human Abilities: Emotional Intelligence,* 2008, pp. 507-536.
[3] Z. Swijtnik, "Daniel Goleman's five components of emotional intelligence," [Online]. Available: https://web.sonoma.edu/users/s/swijtink/teaching/philosophy_101/paper1/goleman.htm.
[4] [Online]. Available: https://web.sonoma.edu/users/s/swijtink/teaching/philosophy_101/paper1/goleman.htm.
[5] "Alan Turing," [Online]. Available: https://courses.cs.washington.edu/courses/csep590/06au/projects/history-ai.pdf.
[6] "Eye Tracking: What is it for and when to use it," [Online]. Available: https://usabilitygeek.com/what-is-eye-tracking-when-to-use-it/.
[7] "Facial LAndmark Detection in OpenCV4," [Online]. Available: https://levelup.gitconnected.com/facial-landmark-detection-in-opencv4-616f9c1737a5.
[8] B. T. N. Dalal, "Histograms of Oriented Gradients for Human Detection".
[9] J. S. V. Kazemi, "One Millisecond Face Alignment with an Ensemble of Regression Trees," 2014.
[10] D. K. J. P. F. Schroff, "FaceNet: A Unified Embedding for Face Recognition and Clustering," 2015.
[11] R. Szeliski, "Computer Vision: Algorithms and Applications," September 3, 2010.
[12] K. K. Y.H. Byeon, "Facial Expression Recognition Using 3D Convolutional Neural Network," *International Journal of Advanced Computer Science and Applications(ijacsa),* 2014.
[13] S. Lin, "An Introduction to Face Recognition Technology," vol. 3, no. 1, 2000.
[14] A. Turing, "Mind," *Computing Machinery and Intelligence,* vol. 59, pp. 433-460, 1950.
[15] J. C. T. Soukupova, Real-Time Eye Blink Detection using Facial Landmarks, February 3-5, 2016.
[16] "Fatigue," [Online]. Available: https://ec.europa.eu/transport/road_safety/sites/roadsafety/files/pdf/ersosynthesis2018-fatigue.pdf.
[17] J. T. M. Stevenson, On the road to prevention: road injury and health promotion, 2014.
[18] "Introduction to OpenCV-Python Tutorials," [Online]. Available: https://docs.opencv.org/master/d0/de3/tutorial_py_intro.html.
[19] D. E. King, Dlib-ml: A Machine Learning Toolkit, 2009.
[20] "Vision AI," [Online]. Available: https://cloud.google.com/vision.
[21] "IoT based Smart Driver Assistance," [Online]. Available: https://www.rfwireless-world.com/Articles/IoT-based-Smart-Driver-Assistance-System-Architecture.html.

**Ioana NAGIT** is a graduate of the Faculty of Economic Cybernetics, Statistics and Informatics at the Bucharest University of Economic Studies Economic Informatics, with a Bachelor’s degree in Economic Informatics – English module. She is currently pursuing a Master’s degree in Databases - Support for Business and has a keen interest in Big Data, Machine Learning, UX Design and Cloud Computing.

**Andreea Ramona OLTEANU** is a graduate of the Faculty of Economic Cybernetics, Statistics and Informatics at the Bucharest University of Economic Studies, with a Bachelor’s degree in Economic Informatics in English. She continued down the path of Economic Informatics and is now pursuing a Masters' degree in Databases-Support for Business, being passionate about Business Intelligence, Machine Learning, and Data Warehousing.

**Ion LUNGU** is a Professor at the Economic Informatics Department within the Faculty of Economic Cybernetics, Statistics and Informatics at the Bucharest University of Economic Studies. He has graduated the Faculty of Economic Cybernetics in 1974, holds a PhD diploma in Economics from 1983 and, starting with 1999 is a PhD coordinator in the field of Economic Informatics. He is the author of 22 books in the domain of economic informatics, 57 published articles (with two ISI-indexed articles) and 39 scientific papers published in conferences proceedings (with five papers ISI-indexed and 15 included in international databases). He participated (as director or as a team member) in more than 20 research projects that have been financed from national research programs. He is a CNCSIS expert evaluator and member of the scientific board for the ISI-indexed journal Economic Computation and Economic Cybernetics Studies and Research. He is also a member of the INFOREC professional association and an honorary member of the Economic Independence academic association. His fields of interest include: Databases, Design of Economic Information Systems, Database Management Systems, Decision Support Systems, Executive Information Systems.