International Science Journal of Engineering & Agriculture
2023; 2(5): 1-8
https://isg-journal.com/isjea
doi: 10.46299/j.isjea.20230205.01.
ISSN: 2720-6319

INTERNATIONAL
SCIENCE GROUP

Real-time eye blink detection using general cameras: a facial landmarks approach

Youwei Lu
Oklahoma State University, Stillwater, USA
ORCID 0009-0009-8707-9623

To cite this article:
Youwei Lu. Real-time eye blink detection using general cameras: a facial landmarks approach. International Science Journal of Engineering & Agriculture. Vol. 2, No. 5, 2023, pp. 1-8. doi: 10.46299/j.isjea.20230205.01.
Received: 08 01, 2023; **Accepted**: 09 14, 2023; **Published**: 10 01, 2023

Abstract: Eyes are essential in Human-Computer Interaction (HCI) as they provide valuable insights into a person's thoughts and intentions. However, current eye movement analysis methods require specialized equipment or high-quality videos, making them less accessible and usable. This paper proposes a real-time eye blink detection algorithm that uses standard cameras, making it widely applicable and convenient. The approach achieves accurate and efficient eye blink detection by leveraging the Eye Aspect Ratio (EAR) algorithm and facial landmarks technique. This paper developed a Python application and conducted tests using a laptop webcam to validate the performance and practicality of the algorithm across various settings. The algorithm's effectiveness depends on carefully tuning parameters such as threshold and frame rate. The results demonstrate the algorithm's potential in real-time eye blink detection, with potential applications in drowsiness detection during driving, prevention of Computer Vision Syndrome, and assistance for individuals with disabilities. By enabling eye blink detection using commonly available cameras, the algorithm paves the way for integrating eye movement analysis into everyday devices and systems, enhancing user experience and enabling more natural interactions. Further refinement and optimization of this approach hold promise for a wide range of applications in HCI, healthcare, and beyond, opening up new possibilities for research and innovation.

Keywords: Eye Blink Detection, Facial Landmarks, Eye Aspect Ratio (EAR), Human-Computer Interaction (HCI), OpenCV, deep learning, Python.

1. Introduction

Eyes are the window to the soul. Eye movements provide a rich and informative window into a person's thoughts and intentions; thus, studying eye movement becomes a hot and challenging area in Human-Computer Interaction (HCI) research. Eye-movement detection research includes eye tracking, gaze, blinking, and pupil movements; those research fields are usually overlapped or interchangeable [1, 2]. A significant drawback of the current approaches is that they are either working on photos or with requirements of very high-quality videos. In this paper, a real-time eye blink detection algorithm is developed and implemented into the device with general cameras. The algorithm should be able to detect eye-blink movements and ensures a high accuracy regardless of the quality of the videos.

2. Object and subject of research

As the most visible eye movement, this paper studies eye-blink detection. In this paper, a real-time eye blink detection algorithm is developed and implemented into devices with general cameras. The algorithm should be able to detect eye-blink movements and ensures a high accuracy regardless of the quality of the videos. The video is supposed to be shot with a general camera under any dynamic or illumination conditions, described as `in the wild' [3]. The design utilizes the Eye Aspect Ratio (EAR) algorithm [4] along with the facial landmarks technique [5]. After developing the algorithm, the application is developed on a laptop and tested using the webcam camera. The programming language used is Python, as it has a rich collection of deep-learning libraries.

3. Target of research

This paper aims to provide a practical eye-blink application that does not have special requirements on devices or photo qualities.

In this paper, facial landmarks are modeled, real-time facial landmark detection is studied in the next section, and real-time eye-blinking is addressed. The applications will be developed using Python on a laptop and tested using laptop cameras. A summary of this study can be found in [6].

4. Literature analysis

Eye movement detection is beneficial and has a wide application. For example, they are assessing the level of drowsiness during driving [7, 8]. Drowsiness during driving is extremely dangerous to both the driver and other people. With the correct capture of eye-blinking, the adaptive algorithm [7] can deal with the intra-individual variability of the blinks or determine if the driver is sleepy. Another example is preventing Computer Vision Syndrome [9]. The Syndrome is very common to nowadays office workers and developers, yet not easy to be diagnosed, and early detection could help to prevent the Syndrome. With the correct capture of eye-blinking, Support Vector Machine (SVC) [10, 11] classifiers with different kernel functions, which diagnose as a feature of the Syndrome. The eye-blinking technology can also help people with disabilities to operate devices purely using their eyes.

Eye-movement technologies start with face detection, as the computer must find the face first. Current face detection algorithms include Viola-Jones, Robust Face Detection Using the Hausdorff Distance, Convolution neural network cascade, and Detecting faces using Eigenface, along with their efficiency and feasibility [12]. For the eye-movement study, current eye-tracking technologies include Electro-Oculography, Scleral Search Coils, Infrared Oculography, and Video Oculography [1]. In contrast, gaze-tracking technologies include Feature-based Gaze Estimation and Appearance-based Gaze Estimation [1].

A significant drawback of the current approaches is that they are either working on photos or with requirements of very high-quality videos. They usually implicitly impose too strong requirements on the setup in the sense of a relative face-camera pose, image resolution, illumination, and motion dynamics. For instance,[13] proposes a camera system to estimate the position and orientation of the eyes. The model uses lines of LED lights, and the differences between cornel reflections and LED lines calculate pupils' positions. This model accurately models the 3D point of gaze and can guide this research. Note that it requires a great camera as well as other devices. In other words, it is hard to yield optimal results using a standard camera, like the webcam in each laptop, or the cameras in our cell phones, which limits the application of eye-movement detection. Therefore, nowadays, robust real-time facial and eye-movement detection using a general camera is an ardent demand from users. Note that a general camera-based computer vision process is also helpful in dye detection and many other industrial applications. [14, 15]

Many attempts in this area of eye gaze or movement detection are reported. Four new features are proposed by [16] compared to old methods to locate the eye positions. The white areas in the four corners of the sclera and the ratio of these white areas to the corners-area are used from each corner to calculate the eye position. This technique allows one to track eye position using a general camera, which can inspire this research. A drawback of this paper is that it focuses on a steady picture rather than real-time videos. A real-time algorithm to detect eye blinks. The algorithm - Eye Aspect Ratio (EAR) - only uses a single scalar quantity in each frame, which makes the algorithm very feasible. The paper shows that a general camera with a facial landmark detector is precise enough to estimate the eye-opening level. The simple algorithm provides a simple-to-implement tool while guaranteeing state-of-the-art results. [5] shows an algorithm to locate the facial landmarks in real-time. This algorithm calculates and finds the local classifiers by fitting a generic 3D model, and both the position and orientation are estimated from the image frames. This algorithm makes face detection possible using a general 3D model. This algorithm is essential for eye tracking, and the same authors published a real-time eye blink detection [4] using the facial landmarks proposed in this paper. As this paper wants to detect face and eye blinks using a general camera instead of a particular device, the low quality of video or photo should be acceptable, and noise in the photo should be considered. For that order, a person-specific model is constructed through incremental updating of the generic model [3]. Thus, it can annotate facial data captured under unconstrained 'in the wild' conditions. This study is about real-time eye tracking, other than steady photo-eye detection, and a method for blink detection from video sequences using a view-based remote eye gaze tracker (REGT) component [17]. The most important advantages are initialization and automatic updating in case of tracking failures. It inspires this research with eye-tracking algorithms in real time.

5. Research methods

Before detecting the eyes, the first step should be finding a face in a photo. This paper uses facial landmarks to localize a face and represent salient regions of the face, including eyes, eyebrows, nose, mouth, and jawline.

Given a photo (or any regular ROI that specifies the Object of Interest, as mentioned in [16] [17]), a shape predictor attempts to localize critical points of interest in the shape detected in the photo. There are several different shapes in one's face, and facial landmarks detection is a subset of the shape prediction problem.

The goal of facial landmarks detection is to find critical facial structures on the face using shape detection, and there are two steps in the process:

- Localize the face shape in the image.
- Detect the fundamental facial structures on the face.

For the location of the face, many deep learning-based algorithms can be used, like HOG + Linear SVM detector [3]. A built-in Haar cascade algorithm is in Open-CV. Through this step, the face is in a bound box (the coordinates of the face in an image).

Given the face founding, the facial structures appear in that region (face bounding box). A good algorithm would achieve the following tasks for this purpose:

- Apply a training set of labeled facial landmarks on an image. The images are with specific coordinates of regions surrounding each facial structure.
- The probability of distance between pairs of pixels.

This project uses dlib, and the pre-trained facial landmark set in dlib uses the location of 68 (x, y)-coordinates to map the facial structures. That data set follows the 68-point coordinates defined in the iBUG 300-W dataset [18]. Figure 1 shows the 68 points. Note that there are also other facial landmark training sets, such as the 194-point model in HELEN.

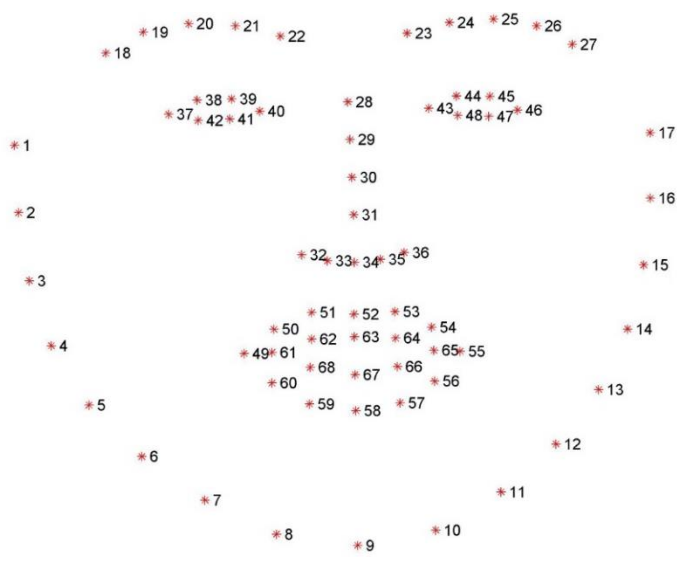

Figure 1. The 68 points mark-up annotations. [18]

Build the facial landmark detector using Python, with dlib and OpenCV. The dlib has a rich library of machine learning algorithms and tools for creating complex applications, and OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV provides a common infrastructure for computer vision applications and accelerates the use of machine perception in commercial products.

There is a mismatch between dlib and OpenCV in that dlib's trained facial landmark detector returns a bounding box rectangle object, but OpenCV takes the box as (x, y, width, height) coordinates. Therefore, a transform function is needed, and the following function in the package 'imutils' achieves this purpose.

```
    def face_to_bb(rect):
x = rect.left()
y = rect.top()
w = rect.right() - x
h = rect.bottom() - y
return (x, y, w, h)
```

A similar issue exists between dlib and that the 'shape' returned from dlib is a collection of 68 coordinates, but Python prefers a NumPy array, thus the following function will do the transform:

```
    def shape_to_np(shape, dtype="int"):
coords = np.zeros((68, 2), dtype=dtype)
for i in range(0, 68):
    coords[i] = (shape.part(i).x, shape.part(i).y)
return coords
```

When everything is set up, and the image input is processed by dlib, one can use the command cv2.imshow to show the bounding box and the 68 coordinates found in faces.

With the face detected, the next step is finding the face structures in the face. Examining the image in Figure 1, one can easily label the facial structures using the indexes, providing reasonable distance tolerance. The typical facial structures are listed below:

- Mouth: through (48 to 68)
- Right eyebrow: through (17 to 22)
- Left eyebrow: through (22 to 27)
- Right eye: through (36 to 42)
- Left eye: through (42 to 48)
- Nose: through (27 to 35)
- Jaw: through (0 to 17)

Using the detector function in dlib to find the structures. The code snippet is added to the application and is shown below:

```
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["shape_predictor"])
```

The app must detect the facial landmarks in real-time in a video stream of images. To do this, we need to use the 'VideoStream' library.

Compared to the detection application in an image, video facial landmark detection takes place in each video frame. Therefore, the application adds the following code snippet and the process is under each frame of the while loop:

```
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=450)
```

With sets of eyes found from the facial landmarks in real-time from the previous sections, the next step is to analyze the eye blink movement.

The traditional image processing algorithms of eye blinking should collect the eye location and track the white region of the eyes, and that is why the traditional methods require high-quality photos or devices for those data processing. Instead, the eye-aspect ratio (EAR) is an elegant solution much easier to implement. The algorithm relies on the ratio of distances between facial landmarks of the eyes.

In the eye aspect ratio (EAR) algorithm, each eye represents 6 (x, y)-coordinates, and there is a relation between the width and height of these coordinates [4]. Figure 2 shows the coordinates locations and their relations. The red dots are the same points found in facial landmarks used in the pre-trained facial landmark set in dlib, which uses the location of 68 (x, y)-coordinates to map the facial structures. That data set follows the 68-point coordinates defined in iBUG 300-W dataset [18]

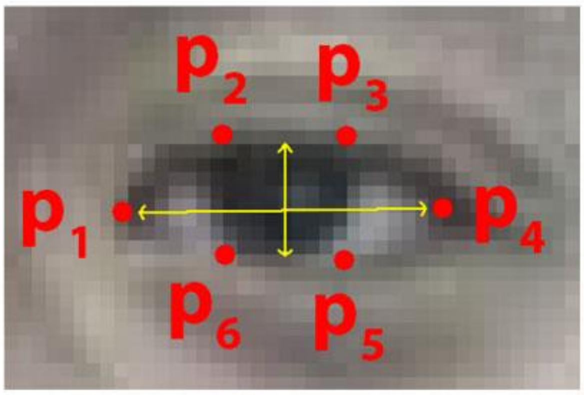

Figure 2. Landmark location of an eye. [4]

An eye aspect ratio (EAR) then can be calculated from each eye as follows:

$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}. \quad (1)$$

This EAR number indicates the distance between the vertical eye landmarks and the horizontal eye landmarks. For a specific eye, the ratio keeps almost constant while the eye is open but suddenly falls to nearly zero at the time of blinking. Therefore, this ratio number is a fast, reliable checkmark for eye blinking.

One can avoid any imaging process with EAR but rely on the EAR number to find out if eye blinking happens. This method is highly reliable, as the number drops dramatically to almost zero. A study [4] proves that the number at the time of eye close is less than 0.1, and the difference between high and low numbers is big enough to define ‘open’ and ‘close’ conditions, precisely as the high and low volt definition used widely in electric circuits.

Coding the equation with the location data from the facial structures is straightforward. See the following code snippet:

```
    for rect in rects:
        shape = predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)
        leftEye = shape[leftStart:leftEnd]
        rightEye = shape[rightStart:rightEnd]
        leftEAR = eye_aspect_ratio(leftEye)
        rightEAR = eye_aspect_ratio(rightEye)
        ear = (leftEAR + rightEAR) / 2
```

With the EAR obtained, we can determine if a blinking happens, and the blink counts as long as the EAR numbers are less than the threshold, while the eye close takes longer than the radio frame. The code could be like the one below, in which the blinkFrame is the variable for blink times counted:

```
        if ear < EAR_THRESHOLD:
            blinkFrames += 1
        else:
            if blinkFrames >= EAR_FRAMES:
                total += 1
```

There is still an uncertainty in the application of the equation on how to determine the threshold, like the noises or eye behaviors (let's say a person blinks an eye but does not completely close it) may cause the radio number to vary from the steady number. Either a too-big or too-small threshold may make the application miss some blinks.

The frame rate of the video is also an issue. If the rate is too big and slower than the blink action, the detection is inaccurate. Although regular cameras should be fast enough nowadays, a study on this could help set the primary standard.

6. Research results

The key parameters that affect the results are the threshold and video frame. The frame is easy to handle, as usually the blink is very fast, and the more petite frame is always better for accurate counting. The only drawback is to filter out the eye close or sleeping, just in exceptional cases, and it can be handled with a small amount of change in code. Let us take driving drowsiness detection, for example; too frequently, eye-blinking is a definite symbol of drowsiness, and the eye-close is even more dangerous. Therefore, ‘false’ detection of ‘sleeping’ as ‘blinks’ can increase driving safety, so there is no need to filter that out. One can add that upper limit in the code to filter the sleeping out if needed.

For the EAR threshold, however, the number needs study. Either a too-big or too-small threshold may make the application miss some blinks. If the threshold is too big, the application may not count some blinks; if the threshold is too small, it may be shorter than an actual blink.

Table 1 shows three tests on the same video with different threshold settings. The accurate blink count is 9, indicating that a threshold of 0.2 is precise. On the same video, testing with a threshold of 0.1 and 0.3 both resulted in a smaller number of counts than the actual result.

Table 1. Blink tests with different thresholds

Test #	Threshold	EAR	Blinks Result
1	0.1	0.30	3
2	0.2	0.29	9 (accurate)
3	0.3	0.18	7

7. Prospects for further research development

Future work on this topic is possible in the following fields:

• Smartphones with cameras are prevalent and easy to access. Applying the algorithms developed in this thesis to mobile apps is reasonable. It provides easier use of eye-blink applications on the go and can integrate with other mobile apps, especially virtual reality applications.

• Eyes are the window to the soul, and eye-blinking sometimes is one part of an emotion. Different types, frequencies, and lengths of blinks and their combinations have different meanings. After the detection of blinking, analyzing those can help us to predict the emotion or obtain more information from a human.

8. Conclusions

In this paper, a real-time eye blink detection algorithm is developed and implemented into a laptop with general cameras using Python. The facial landmarks are modeled, and then the shapes in the face are detected. Next is the real-time facial landmarks development, and finally, the real-time eye blinking detection is developed and tested in a Python application on a laptop.

The facial landmarks are modeled with the data set following the 68-point coordinates defined in the iBUG 300-W dataset. The typical face structure shapes can be obtained from the facial model, and the process can be done within each video stream frame. An eye aspect ratio (EAR) can be calculated with the eyes' coordinates found, and the EAR number indicates the distance between the vertical eye landmarks and the horizontal eye landmarks. The ratio number is used as a fast, reliable checkmark of eye blinking.

As the significant parameters that affect the application functions, the video frame and the EAR threshold are discussed and tested in the application. It is found that the video frame is not a big issue in modern cameras, but the EAR threshold needs to be tuned for an accurate result.

References:

1) Chennamma, H. R., & Yuan, X. (2013). A survey on eye-gaze tracking techniques. *Indian Journal of Computer Science and Engineering*, 388-393.

2) King, D. (2014, August 28). *Real-Time Face Pose Estimation*. Retrieved from Dlib C++ Library: http://blog.dlib.net/2014/08/real-time-face-pose-estimation.html

3) Asthana, A., Zafeiriou, S., Cheng, S., & Pantic, M. (2014). Incremental face alignment in the wild. *Conference on Computer Vision and Pattern Recognition* (pp. 1859-1866). Columbus, OH, USA: IEEE.

4) Soukupová, T., & Čech, J. (2016). Real-Time Eye Blink Detection using Facial Landmarks. *21st Computer Vision Winter Workshop.* Rimske Toplice, Slovenia: 21st Computer Vision Winter Workshop.

5) Čech, J., Franc, V., & Matas, J. (2014). A 3D Approach to Facial Landmarks: Detection, Refinement, and Tracking. *22nd International Conference on Pattern Recognition* (pp. 2173-2178). Stockholm, Sweden: IEEE.

6) Lu, Y. (2023). REAL-TIME EYE BLINK DETECTION WITH GENERAL CAMERAS. *The 28th International scientific and practical conference "Science and development of methods for solving modern problems"(pp. 198-201).* Melbourne, Australia: International Science Group.

7) Baccour, M. H., Driewer, F., Kasneci, E., & Rosenstiel, W. (2019). Camera-based eye blink detection algorithm for assessing driver drowsiness. *Intelligent Vehicles Symposium (IV)* (pp. 987-993). Paris, France: IEEE.

8) Lu, X., Song, Y., Chen, B., Liu, X., & Hu, W. (2022). A novel deep learning based multi-feature fusion method for drowsy driving detection. *Industry and agriculture*(3449), 3449.

9) Pal, M., Banerjee, A., Datta, S., Konar, A., Tibarewala, D. N., & Janarthanan, R. (2014). Electrooculography based blink detection to prevent computer vision syndrome. *International Conference on Electronics, Computing and Communication Technologies (CONECCT)* (pp. 1-6). Bangalore, India: IEEE.

10) He, W., Hu, W., Yang, Y., Shen, H., Wu, Y., Song, Y., & Liu, X. (2022). Improved left-and right-hand tracker using computer vision. *Student scientific research*(3), 21-29.

11) Xie, Z., Hu, W., Zhu, J., Li, B., Wu, Y., He, W., & Liu, X. (2022). LEFT AND RIGHT HAND TRACKER BASED ON CONVOLUTIONAL NEURAL NETWORK. *Актуальные вопросы современной науки образования*, (pp. 61-67).

12) Srivastava, A., Mane, S., Shah, A., Shrivastava, N., & Thakare, B. (2017). A survey of face detection algorithms. *International Conference on Inventive Systems and Control (ICISC)* (pp. 1-4). Coimbatore, India: IEEE.

13) O'Reilly, J., Khan, A. S., Li, Z., Cai, J., Hu, X., Chen, M., & Tong, Y. (2019). A novel remote eye gaze tracking system using line illumination sources. *Conference on Multimedia Information Processing and Retrieval (MIPR)* (pp. 449-454). San Jose, CA, USA: IEEE.

14) Lu, Y., & Pagilla, P. R. (2014). Modeling of Temperature Distribution in Moving Webs in Roll-to-Roll Manufacturing. *Journal of Thermal Science and Engineering Applications, 6*(4), 041012. doi:https://doi.org/10.1115/1.4028048

15) Lu, Y., Jee, C., & Pagilla, P. R. (2016). Design of a model-based observer for estimation of steel strip tension in continuous galvanizing/annealing lines. *2016 American Control Conference (ACC)* (pp. 3249-3254). Boston, MA, USA: IEEE. doi:https://doi.org/10.1109/ACC.2016.7525418

16) Al-Btoush, A. I., Abbadi, M. A., Hassanat, A. B., Tarawneh, A. S., Hasanat, A., & Prasath, V. S. (2019). New Features for Eye-Tracking Systems: Preliminary Results. *10th International Conference on Information and Communication Systems (ICICS)* (pp. 179-184). Irbid, Jordan: IEEE.

17) Torricelli, D., Goffredo, M., Conforto, S., & Schmid, M. (2009). An adaptive blink detector to initialize and update a view-based remote eye gaze tracking system in a natural scenario. *Pattern Recognition Letters*, 1144-1150.

18) Sagonas, C., Antonakos, E., Tzimiropoulos, G., Zafeiriou, S., & Pantic, M. (2016). 300 Faces In-The-Wild Challenge: database and results. *Image and Vision Computing*, 3-18.