WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Tracking Eye Movement Using Facial Keypoints and Image Processing

Parth Gupta[1], M.L. Sharma[2], K.C Tripathi[3]

[1,2,3] Maharaja Agrasen Institute of Technology, New Delhi, Delhi

Abstract

The modern world is witnessing a shift of activities from physical to the virtual mode at a monumental scale. This includes examinations that are now conducted online through video stream based proctoring. In online mode of examinations, the risk of candidates resorting to unfair means increases. This paper attempts to solve the problem of tracking the eye movement of a candidate who is taking a virtual exam, in order to detect if the candidate is looking away from the screen. We show how image processing techniques like thresholding and morphological transformations can be coupled with facial landmark detection models to achieve accurate tracking results. We present a computer vision pipeline consisting of a face detection model, facial landmark detection model and eye tracking modules. We show that in suitable physical conditions, the aforementioned pipeline tracks eye movement in real time.

1. Introduction

Post the covid 19 pandemic, the world has witnessed a massive shift of real-world activities to the virtual world. One of such activities is conducting examinations. In examinations that happen in virtual mode, the scope of using unfair means increases because the monitoring of a candidate through video feed exposes multiple loopholes. There is a need for a system that can help human proctors in conducting exams by expanding their capabilities to detect suspicious activities that hint at the use of unfair means in examinations.

Secondly, since a single proctor is responsible for monitoring the behavior of multiple candidates, the attention span of a proctor per candidate decreases as the number of candidates assigned to them increases. Our proposed system aims to create a preliminary proctoring layer, i.e., it helps the human proctor to channel their attention towards selected candidates who are found to be indulging in suspicious activity, when detected by the system.

We propose a solution to the problem which is to design a computer-vision based system that can track a candidate's eyeball movement and send an alert to relevant entities if the movement of eyeballs suggest that the candidate is cheating by looking away from the screen. The program can keep a record of the time for which the user was found to be looking away and if the recorded time exceeds the set threshold, it can generate a message to alert the candidate and/or the proctor.

2. Related Work

Several studies have been done in the field of face detection and facial landmark detections. They find applications in multiple other use cases like driver drowsiness detection, autofocus photography systems and even facial recognition[1]. Multiple algorithms have been proposed for predicting facial key-points with general trade-off between inference time and prediction accuracy. Therefore, we chose an algorithm which provides sufficient accuracy while maintaining real time processing speed.

2.1 Face Detection Using Dlib.[2]

In order to perform tasks like facial recognition, we can use the framework provided by the Dlib Library[3]. Dlib provides 2 pretrained models for performing face detection on an image. The Models take an image as input and predict the coordinates of the bounding boxes. The bounding boxes coordinates are the pixel level x and y coordinates of the 4 vertices of the rectangular area in the image within which a face has been detected with a high confidence.

Out of the 2 models provided in the Dlib library, the first model is lightweight and uses an SVM linear classifier. Whereas, the second model is computationally and memory consumption wise expensive, although more accurate, and uses a convolutional neural network to perform face detection. We have used the first model since our application needs to detect faces in real time scenarios and therefore accuracy can be traded off for a faster runtime.

2.2 Facial Landmark Detection using Dlib.

Landmarks in computer vision are the points that denote the location of useful features present in the region of interest. Facial landmarks are points that describe the position of facial features like eyes, nose, mouth, ears etc.

Dlib provides a regression trees ensemble learning based model for facial landmark detection. It is the implementation of 'One Millisecond Face Alignment with an Ensemble of Regression Trees' paper by Kazemi and Sullivan (2014)[4].

We have applied the Facial landmark detection algorithm only over the rectangular region specified by the bounding box predicted by the face detection layer. Subsequently, we have used landmarks around the eye sockets to segment out the user's eyes from the rest of the face.

3. Method

3.1 Overview

The video stream from the candidate's web camera would be sent as input to the system. The system will apply face recognition and feature detection algorithms to generate facial feature key-points map. Using the key-points map, the key points of eyes would be selected to segment out the eyes from the frame.

After extracting eyes from the frame, a threshold and contouring based segmentation would be done to segment out the pupils. Pixel level coordinates would be used to demarcate pupils and their positioning. The position would be used to classify if the candidate is looking away from the screen or not.

The movement of pupils with respect to the larger eye socket will be monitored. If the pupil moves towards the either extreme ends of the eye sockets, it denotes that the candidate is looking away. If the pupil is displaced from its mean position for more than a specified time interval, then a message to alert the authorities will be generated.

3.2 Face Detection

1. Feature extraction using HOG – Histogram of Oriented Gradients Method[5]. It is a technique to extract features from an image (Fig. 1). A given image is pre-processed using this method before being sent to the classifier. This step helps in reducing redundant computation and helps the algorithm focus on only the required features of the image instead of considering all the pixel intensities.

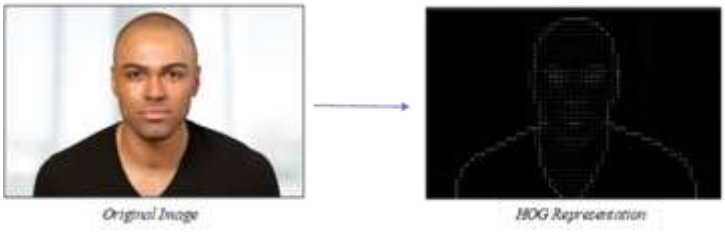

Figure 1: HOG Representation of an RGB Image

2. Linear SVM - Support Vector Machine. After feature extraction, the feature representation of the original image is sent to a pre-trained linear SVM model. The Linear SVM classifier predicts a linear boundary that maximizes the margin or the cartesian distance between the cluster of points belonging to different classes[6].

An image can be represented as a single point in a multidimensional space. Therefore, to identify if an image has a face or not is a classification problem of finding a hyperplane that can divide the multidimensional space, and also the points lying in them, into two classes – having face, and not having a face.

A window of varying size (smaller than the image) is slid across the image and the pixels of the image, lying inside a window are sent to the linear SVM model to predict if the window contains a face or not.

The coordinates of the window containing a face are given as final output (Fig. 2).

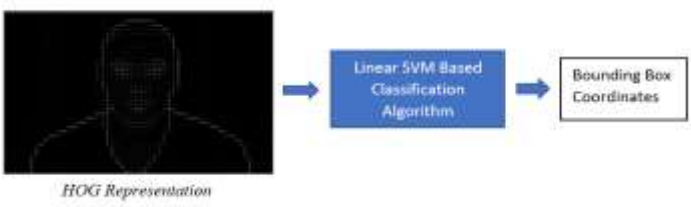

Figure 2: Working of Linear SVM

The bounding boxes can be plotted on the image as shown below (Fig. 3).

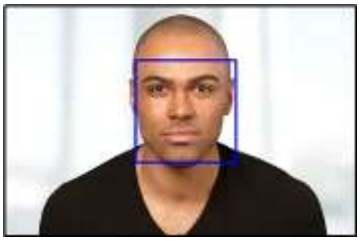

Figure 3: Plot of Bonding Box

3.3 Facial Landmark Detection

The problem of facial landmark detection is solved by using pre-trained models of the D-lib library[2] which are an implementation of "One millisecond face alignment with an ensemble of regression trees" by Kazemi, Vahid, and Josephine Sullivan[4].

The model is used to predict the location of 68 key-points on the face that map important facial features like eyebrows, eyes, lips, nose, face boundary etc (Fig. 4).

Figure 4: Plot of Facial Landmarks

3.4 Detecting Pupil

1. Creating Segmentation Mask for Eyes. Out of the 68 landmarks given by landmark detection model, the landmarks surrounding the eyes are at the following indices of the landmark vector:

$$left = [36, 37, 38, 39, 40, 41]$$

$$right = [42, 43, 44, 45, 46, 47]$$

We define a matrix of zeros having height and width dimensions equal to those of the image.

Followingly, the points specified by left and right eye landmarks are used to calculate a contour for each eye respectively. We then apply a function to fill the matrix cells lying inside the area bounded by each contour with the value of 255. Intensities 0 and 255 represent black and white colors respectively (Fig. 5).

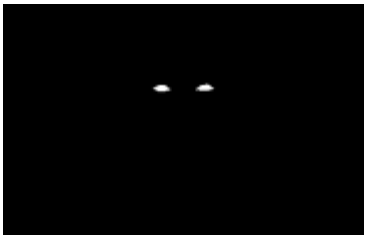

Figure 5: Eye Segmentation Mask

2. Segmenting eyes from the Image. Using the segmentation masks, we perform a bitwise 'and' operation over the original image (Fig. 6).

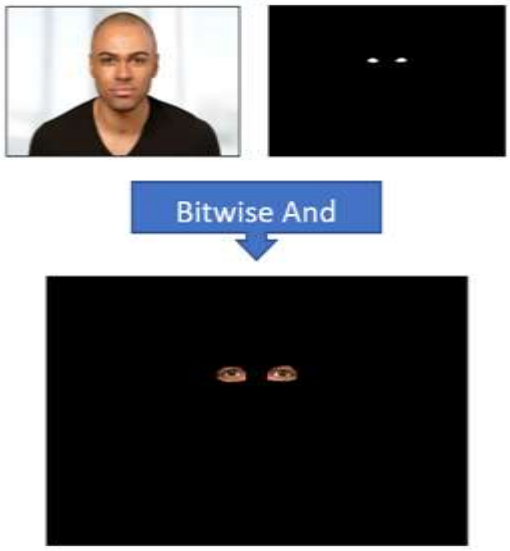

Figure 6: Generation of Eye Segmentation Output

3. Tracking Position of Pupil. After segmenting out eyes, the black background in the segmentation result is converted to white and then it is converted from a 3 channel RGB image to a grayscale image (Fig. 7).

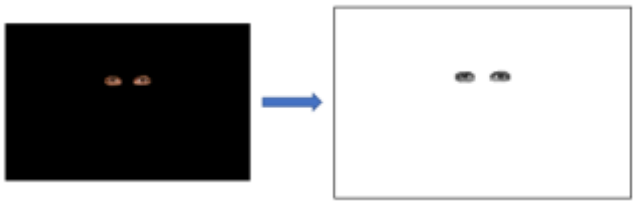

Figure 7: Conversion of Eye to Grayscale

Followingly, we have applied a pixel-intensity based thresholding method to segment out pupils since pupils are darker than the rest of the eyes and therefore the pixels with the lowest intensities belong to the pupil (Fig. 8).

Figure 8: Thresholding Result

The segmentation mask contains noise and imperfect segmented areas. Therefore, image processing techniques like morphological operations are applied to the segmentation mask to eliminate unwanted noise and refine the result[7] (Fig. 9). We have used the following mask pipeline:

1. Dilation (2 iterations) – Pixels are added along the boundary of a white blob to extend the area. This helps in enhancing the region of interest that is obtained after erosion.

2. Erosion (1 iteration) – The pixels near the boundary of a white blob are discarded or eroded. This helps in removing noise signals.

3. Median Blur (Kernel Size = 3) – The median of the pixels lying inside the sliding window are used as a new intensity of that pixel at that location of the window. It helps in smoothing the image and also removing any noise.

Figure 9: Output of mask pipeline

After the final segmentation mask has been generated, we infer the exact location of the left and right pupil using the contouring technique. We divide the image into 2 halves such that halves contain one eye each.

On each individual half, we use functions from the OpenCV library to find contours of all the white blobs present and select the contour which encompasses the greatest area in the image. Further, the center of the region bounded by contours is calculated using moments which essentially gives the center of mass of the ROI (Fig. 10).

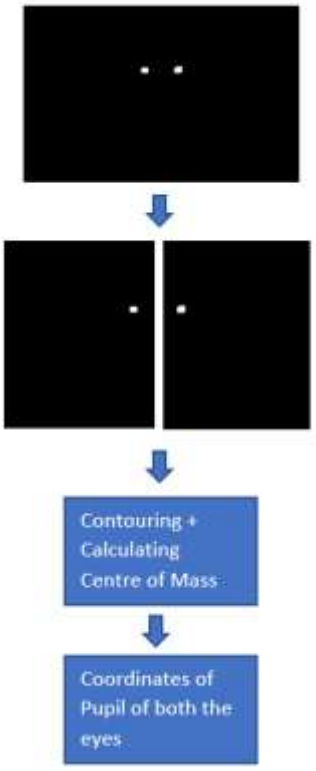

Figure 10: Calculation of Pupils' Coordinates

4. Monitoring the displacement of Pupil from the mean position. We obtain the coordinates of the pupils' center using the approach described above. We also use the coordinates of the extreme ends of the eye sockets (obtained using the landmark vector) and for each calculate the following distances:

a. Total width of the eye socket (w)

b. Distance from pupil to inner end of eye socket (di)

c. Distance from pupil to outer end of eye socket (do)

If $(di < w/4)$ *or* $(do < w/4)$:

It means that the person is looking away from the screen since their pupils have shifted significantly from the mean position. When this condition is satisfied, we can generate a message to alert the user and/or proctor that the candidate might be looking away from the screen.

Otherwise,
if $(di >= w/4)$ *and* $(do >= w/4)$:

It means that the shift in the position of the pupil is not large enough and is within permissible range. In this case, no alert is generated.

4. Experimental Results

Calibration Mode. When the program starts, it prompts the user to adjust the thresholding value using the trackbar attached to an image window (Fig. 11, 12). The initial position of the bar is at 0 and shall be increased gradually till at least one or preferably both the pupils' mask is displayed on the output window. This process is termed as calibration and needs to be done to adapt the algorithm to different camera angles, candidate's distance from the screen and different lighting conditions.

Figure 11: Uncalibrated State

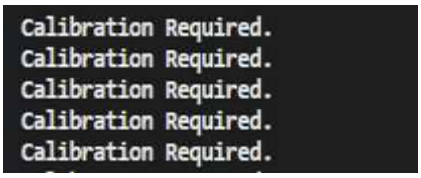

Figure 12: Message to prompt the user to calibrate the system

Performing Calibration. The trackbar is moved until the threshold value reaches 22. This means that all the black pixels in the eyes with intensity lesser than 22 will be filtered out (Fig. 13). This value turns out to be optimal for the given test picture.

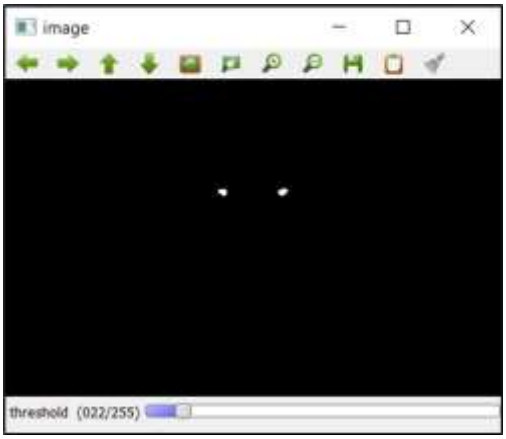

Figure 13: Output window with trackbar to adjust thresholding value

After Calibration. Upon calibration, the algorithm starts segmentation of eye pupils and tracks its movements. The green circles demarcate the innermost and outermost extreme ends of both the eye sockets. The red circle in the center superimposes the inferred position of the eye-pupil (Fig. 14). The red circle shows where the algorithm perceives the pupil of the candidate to be situated at a particular instant.

Figure 14: Tracking Mode

The coordinates of pupils and eye sockets are produced and messages are generated depending on the displacement of the pupil.

If the pupil's position is within the safe range, i.e., a range within which the candidate is perceived to be looking at the screen, then a green message is produced for those image frames (Fig. 15).

Figure 15: Messages when subject is perceived to be looking into the screen

In situations where the candidate's pupils are displaced beyond the safe range, the algorithm perceives that they are looking away from the screen and a message in red is printed (Fig. 16).

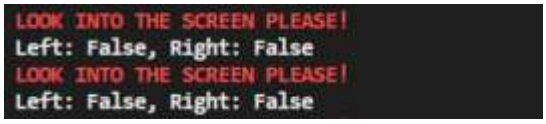

Figure 16: Messages when subject is perceived to be looking away from the screen

Limitations of the System.

1. The algorithm fails to work when a candidate is sitting in dim lighting conditions or when there is non-uniform illumination in the image scene. If the light source is closer to one eye, then the thresholding works only on one eye and does not produce accurate results.
2. Moreover, the algorithm performs poorly when a candidate is wearing spectacles. The regions around the eye-balls get distorted due to reflections on the eye glasses.
3. Lastly, the algorithms cannot handle cases where a candidate is sitting far away from the camera. This is because due to the limited resolution of the webcam, the entire eye socket of the subject is represented only by a few pixels and as a result, tracking accuracy decreases.

5. Conclusion

An intelligent proctoring system was created that uses machine learning techniques like computer vision and image processing to track the eye movement of a person using the video feed. The system could generate alerts in real time when the candidate was perceived to be looking away from the computer screen.

The system requires manual calibration during start-up to ensure that proper thresholding can be done. It was found the thresholding is sensitive to lighting conditions, subject's distance from the screen and any transparent/translucent obstruction between the camera lens and eyes (spectacles, tinted glasses etc.)

Under ideal conditions, that is, a uniform and well-lit scene with the subject situated 1-2 feet away from the webcam, the algorithm gave optimal performance. For each image frame in the video feed, the algorithm printed a message on the screen corresponding to the position of the pupil.

If the displacement of the pupil was within permissible range, a green message 'Looking straight' was printed on the screen. However, when the subject looks outside of the screen such that pupils displace beyond the permissible range, then the program prints a red message "Look into the screen please!" on the terminal.

6. Future Scope

Some improvements that can be made to the system are to make it more robust, also new features to detect usage of unfair means during examinations can be introduced. Time duration for which subject is found to be looking away can be measured and be used to classify them into different risk categories. Features of alerting proctor when multiple persons are detected in a single frame, and object detection module to detect presence of a smartphone in the image can be introduced.

7. References

[1] Zhao, W., Chellappa, R., Phillips, P.J. and Rosenfeld, A., 2003. Face recognition: A literature survey. ACM computing surveys (CSUR), 35(4), pp.399-458.

[2] http://dlib.net/

[3] King, D.E., 2009. Dlib-ml: A machine learning toolkit. The Journal of Machine Learning Research, 10, pp.1755-1758.

[4] Kazemi, V. and Sullivan, J., 2014. One millisecond face alignment with an ensemble of regression trees. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1867-1874).

[5] Dalal, N. and Triggs, B., 2005, June. Histograms of oriented gradients for human detection. In 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05) (Vol. 1, pp. 886-893). Ieee.

[6] Chang, C.C. and Lin, C.J., 2011. LIBSVM: a library for support vector machines. ACM transactions on intelligent systems and technology (TIST), 2(3), pp.1-27.

[7] Sreedhar, K. and Panlal, B., 2012. Enhancement of images using morphological transformation. arXiv preprint arXiv:1203.2514.