# Using Eye-Gaze and Eye-Blinking to enable typing on a Virtual Keyboard

Madhur Tandon, Mudit Garg, Jigme Lobsang Lepcha

Indraprastha Institute of Information Technology, Delhi

## Problem Statement

Differently abled people usually have trouble interacting with computers and similar interfaces. The standard input and output devices such as the mouse and the keyboard requires the use of one's hands. We propose to enable and empower people suffering with neurolocomotor diseases, paralysis, dysarthria, etc. by making a new form of input that exploits the information from one's eyes. The system shall leverage eye-specific features and cues such as eye-movements, eye-gaze direction, blinking of eyes to facilitate the user in typing of text through an on screen keyboard. This task requires several concepts of classical computer vision - some of them being histogram equalisation, erosion / dilation, object tracking, etc.

## Literature Review

Few Computer Vision Techniques used by us in our algorithm are mentioned below:

- Histogram Equalisation: This is a image quality improvement technique. This spreads the colour value in the image so that it looks more appealing. This increases the variance/contrast of the image. We use this technique to improve the quality of the eye images.
- Dilation: This technique replaces a pixel with the maximum value in the intersection of image and kernel applied at that pixel. Therefore it results in thinning of black objects.We use this technique to detect the centre of pupil of the eye.
- Histogram of Oriented Gradients: This technique is used for object detection. This techniques remove all the unnecessary information in the image and only keeps the rich information about gradients which are important for object detection. This technique is used by the Dlib library to detect faces.

## Database Details

The Application is User - Centric and thus, no dataset is required as such. They application will detect the faces, eyes from realtime video feed of the front camera in the computer system.

## Proposed Algorithm

### Frontal Face Detection

One of the first steps to ultimately detect eyes is to first detect a frontal face in a live camera feed. Since this is only a small component of the overall system, thus, we cannot afford to collect our own dataset / train our own models at this stage. Hence, the already trained model based on Histogram of Oriented Gradients and the SVM classifier is used. This model is provided by the Dlib library. The model is built with 5 HOG features - front, left, right, front (rotated left) and front (rotated right) looking images of faces.

The faces were taken from the LFW dataset. This method has the following Pros and Cons:

PROS: Fastest Method on CPU, Works well for frontal and slightly non-frontal faces, light-weight model, robust to small occlusions.

CONS: The bounding box excludes the forehead and the chin at times, The minimum size of faces need to be 80x80, etc.

For our application that has realtime requirements and needs to work with eyes, the Pros outweigh the Cons and thus, this model is directly used on each frame of the camera feed.

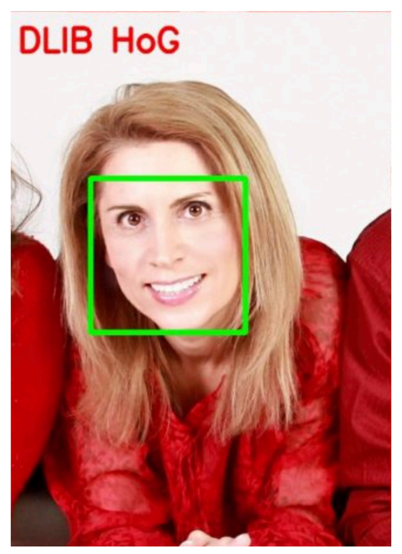


Fig 1: Frontal Face Detection using DLib Library

### Getting Facial Landmarks

To particularly extract the Eye-Region from the faces, the method of extracting facial landmarks from a given face is used. The facial landmarks correspond to a large number of facial regions - the most common ones being mouth, left eye, right eye, nose, jaw, etc. This task too is indeed a small stepping stone towards building the final system and thus, a pre-trained model is used for this task as well.

Specifically, the pre-trained model extracts 68 points of interest from each face that map to actual facial structures on an image frame.

As seen from the figure below, the points:

36, 37, 38, 39, 40, 41

and

42, 43, 44, 45, 46, 47

correspond to the left and the right eye respectively.

### Detecting the Pupil Centre

The mean of the 6 points corresponding to each eye represents the geometric centre of the eye in some sense. However, to determine the direction of gaze, one also needs to account for the coordinates of the centre of the pupil (which can change even if the geometric centre of the eye is fixed).

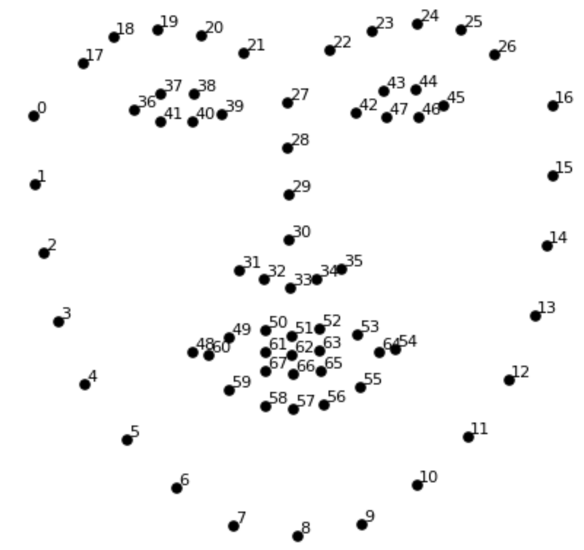

Fig 2: Visualisation of Facial Landmarks

To achieve this goal, we first calculate a rectangular region over each eye using offsets that are chosen empirically. These offsets are related to the rectangular face region returned by the Dlib's frontal face detector.

The extracted region is

- Converted to Grayscale
- Its histogram is equalised.
- A Binary threshold is applied and the lower and up- per limits are set to 60 and 255 respectively (chosen empirically).
- A dilation process is performed for 7 iterations with the structuring element as a 5x5 kernel of all 1s.
- Contour Detection
- Selection of Contour with maximum area - which represents the pupil.
- Centroid of that contour represents the centre of the pupil.

### Detecting the Eye Gaze

The window for each eye is then divided into slices and the location of the pupil is live tracked using the green dot shown here. If this pupil centre falls into the left slice or the right slice, the detected gaze is reported as right and left respectively.

Similar procedure is followed for detecting the directions Up and Down.

### Detecting an Eye Blink

The centre of points 37,38 and the centre of 40, 41 for the left eye is joined together to get a vertical line that passes through the geometric centre of the eye. The horizontal line is given by joining the extreme endpoints 36 and 39. The ratio of length of this vertical line to the horizontal line is then used to determine the blink. A similar procedure is followed for the right eye and the thresholds are chosen empirically.

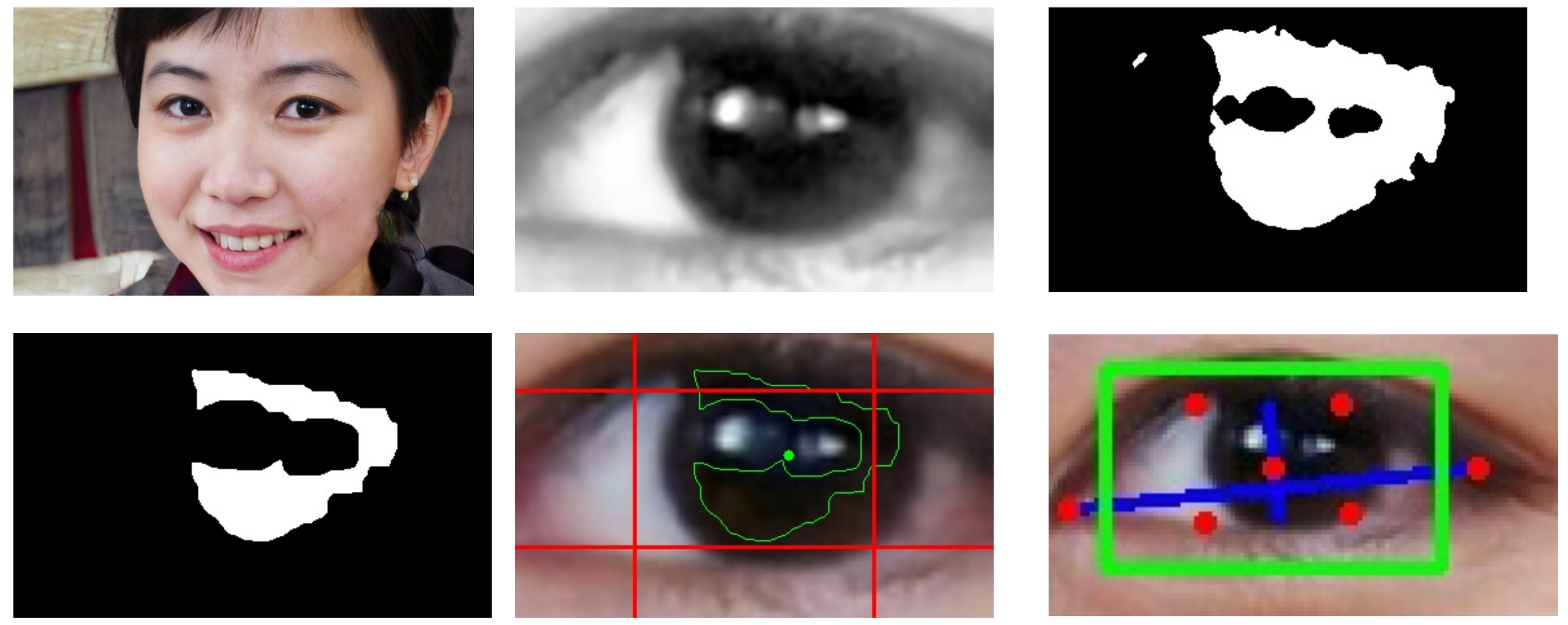

Fig 3: The Complete Pipeline

### Making the Keyboard

An on screen keyboard is needed (with big visual keys) so as to make it easier for the user to see and essentially type. This is made by in built OpenCV functions and using suitable placement of the keys.

### Integration with the Keyboard

The Eye Gaze Direction is used to navigate the keyboard giving a sense of live interactivity to the user. The feature of detection of an eye blink is used to select and type that particular key on the canvas. The highlighted key (showcased as white in colour) changes according to the Eye Gaze. This highlighted key can then be clicked using an eye blink.

### Results -

The whole pipeline has been designed to run readily. Although we cannot compare our results to the ground truth, we can observe the working of the system on almost all faces in real time, however some adjustments to the thresholds might be needed at times.