Virtual Keyboard

A thesis report submitted for BTP phase II
by

Avish Kabra
(Roll No. 160108044)

&

Chandan Agrawal
(Roll No. 160108043)

Under the guidance of
Dr. M.K. Bhuyan

DEPARTMENT OF ELECTRONICS & ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
June 2020

Abstract

Our project provides a vision-based human-computer interface. The interface senses and interprets eye-blinks as controls along with gaze detection methods. We used image processing methods like haar-like features for automatic face detection and eye tracking and blink detection based on landmarks. Voluntary eye blinks are differentiated from natural ones and are used to control the communicating window. These eye blinks serve as inputs and have pre-determined meanings varying upon the stage in which the controller is present. The typing process is further eased out by providing the next word predicting panel so that sentences can be completed quickly and with minimal efforts. Performance of both the methods was compared by using various face orientations, and words per minute were calculated by averaging the size of sentences typed in a minute over several attempts.

The added Gaze detection mechanism helps in easy controlling of the mouse. A heatmap is also generated for research purposes so that in future developments, this data can be used to design a more efficient layout for the controls. Features like scrolling are also added using this method.

Contents

List of Figures

1 Introduction

One out of 50 people suffer from paralysis, says Christopher & Dana Reeve Foundation's research. There can be various causes for paralysis, including intentional or accidental poisoning, illness, or injury.

The most basic way of defining paralysis can be loss of movement, which can be partial or full, usually in response to an injury or illness. Paralysis is a genuine and significant factor for any spinal cord injury (SCI) survivor and their families. The part affected depends on the type and location of the injury.

In severe cases of paralysis, an affected person cannot perform a majority of everyday tasks, and the inability to communicate with others is one of them. Communication is essential for anyone, and its requirement increases further if you are dependent on someone else. This becomes a significant issue; therefore, we decided to build something which can be used as an efficient solution to this problem. Even in some of the severe cases of disabilities, the person was able to blink his eyes voluntarily. On studying about it, we found out that the nerve connections that control blinking run through the trigeminal nerve and the facial nerve. Both of those nerves connect directly to the brain, not the spinal cord, so if the reason for paralysis is damage to the spinal cord, which is generally the explanation, then those nerves are not usually affected. Therefore we decided to use these voluntary blinks as an input mechanism for our communicator. The enhanced functionalities also provide eye tracking.

The eye-tracking system is the method of determining where and how someone is looking. Visual monitoring records actions that are not readily controllable or measurable, which is a primary explanation why they are viewed as a technical way of evaluating a product or device's functionality. Although there are many uses for this through education, market analysis, the medical sector, and corporate research, visual monitoring can be used to help people with disabilities achieve independence.

Hence, the eyes become an important input mode for improving the quality of life of people with severe motor disability by providing a communication channel to the outside world, a goal facilitated by enabling text entry using the eyes.

People get substantial information from their eyes. Eye typing offers new ways of communicating with family, colleagues, and relatives. The software has huge potential for persons with disabilities whose conditions prohibit them from interacting and causes isolation. Disabilities, such as ALS, cerebral palsy, or locked-in syndrome, are severe and often lead to complete loss of control over voluntary muscles, except the eye muscles, rendering the individual paralyzed and mute.

We programmed a human-machine interface with the concept of blink detection and gaze tracking. The human-computer interface is applied to an assistive device, namely a virtual keyboard, which is explicitly designed for the severely physically disabled. They can operate the proposed design by merely blinking their eyes, thus communicating with the outside world. Gaze detection is used to control mouse movements.

2 Literature Survey for Eye-tracking

The face is the mind index and eyes are the key to the soul. Eye movements offer a rich and detailed insight into the thoughts and intentions of an individual. And the study of eye movement will decide what people think based on where they look. Eye-tracking is the assessment of eye movement/activity, and gaze tracking (point of view) is the study of eye-tracking data with respect to the head / visual scene. Typically the eye- and head-position integration is used to determine the gaze location in the visual scene. Simple eye trackers monitor only the gaze direction relative to the head (with a head-mounted device, electrodes, sceleral coils) or a fixed eyeball position (systems requiring head-mounting).

Oculography is a method of documenting the location and movements of the eyes. There are four different methods of monitoring eye movements

Electro Oculography
In this process, sensors are attached to the skin around the eyes to determine that when eyes rotate, an electric field exists. The location of the eye can be determined by detecting slight variations in the skin potential around the eye. The horizontal and vertical movements can be documented separately by carefully positioning the electrodes.
However, even when there is no eye movement the signal will sometimes alter. This technique is not ideal for daily use, since it involves close contact with the patient by electrodes but is still widely used by clinicians.

Sceleral Search Coils
In a magnetic field, when a coil of wire moves, the field induces a voltage in the coil. When this coil is attached to the eye then an eye position signal is generated. Small wire coils are embedded in a contact lens to measure the human eye movements. This is injected into the eye after the application of local anesthetics. An integrated mirror inside the contact lens allows the measurement of reflected light. Alternatively, an active coil in the contact lens makes it possible to detect the position of the coil in a magnetic field. The benefit of such a system is its high precision and almost infinite resolution in time.

Infrared Oculography

The intensity of reflected infrared light is measured in infrared oculography. The disparity between the ratios of IR light reflected back from the surface of the eye carries the information on changes in the eye location. You can position the light source and the sensors on glasses.

Video Oculography

Video-based eye tracking is perhaps the most common tool used in professional eye trackers. The eye-gaze tracking was, until recently, a very complex and costly activity restricted for laboratory research only.

Previous work done on video-based

Single Camera Eye tracker

Some video-based eye trackers operate by illuminating the eye with a source of infrared light. Such light produces a glint on the eye's cornea, which is considered a corneal reflection. Glint was used as the point of comparison for gaze estimation in most of the existing work

Multi-Camera Eye Tracker

Multiple cameras are used either via wide-angle lens cameras or movable narrow-angle lens cameras to achieve these goals. In the literature, multiple camera systems either use different cameras on each eye or use a single camera to monitor head position to account for changes in head posture. Then merge all the camera details to determine the gaze position

Video oculography mechanisms gather information (image data) from one and sometimes more cameras. The first task is to detect the position of the eye inside the image. The angle of gaze can be calculated on the basis of the information collected from the eye regions and probably head position. The important parts of the eye include the pupil – the aperture that illuminates the eye, the iris – the colored muscle group controlling the pupil's diameter, and the scelera which is the white protective tissue covering the eye.

Feature-based gaze estimation

Feature-based approaches investigate human eye characteristics to recognize a range of eye defining features such as contours (limbus and pupil contour), eye corners, and corneal reflections are specific features used to measure the gaze. Feature-based methods aim to classify informative local eye features that are generally less susceptible to lighting and viewpoint variations

Appearance-based Gaze estimation

Appearance-based approaches specifically detect and monitor eyes based on the photometric dimension. Appearance related techniques use image data to predict the path of your attention by mapping image data to screen coordinates. The major appearance-based methods are based on a morphable model, grayscale unit images, appearance manifolds, Gaussian interpolation, and cross-ratio.

3 Detection methods

3.1 Haar Cascades

Haar feature-based cascade classifier is a machine learning-based approach that is used for object detection. It is a very effective method that uses a lot of positive and negative images to train a cascade function, which is used to identify objects in images. Object detection using Haar feature-based cascade classifiers was proposed by Paul Viola and Michael Jones in [1]. This technique can be used to detect any object, and it is well known to detect face and other body parts.

We are using Haar Cascade to detect faces and eyes. To use this algorithm, we need a lot of images of faces that will work as positive images in our case and a lot of images without faces, which will work as negative images. Then We need to extract features from these images. A Haar feature considers neighboring rectangular regions in a detection window at a specific location (White and black rectangular regions), sums up the intensities of pixels in each region, and measures the difference between these quantities, which are like the convolutional kernel. There are three types of Haar features that are used. (a) Edge Feature (b) Line Feature (c) Rectangular Feature. These features are shown in the below image.

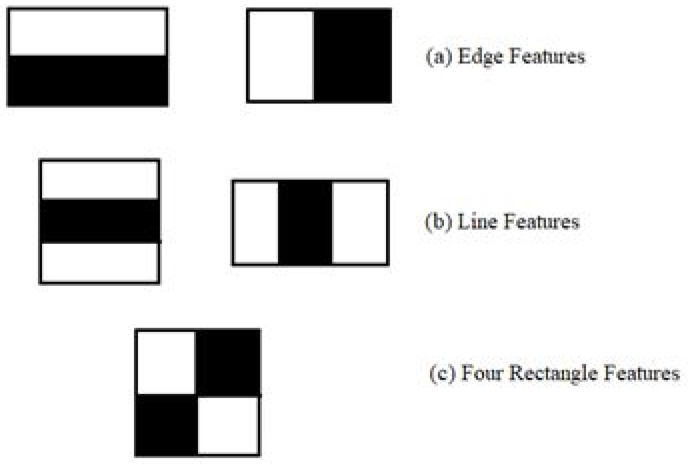
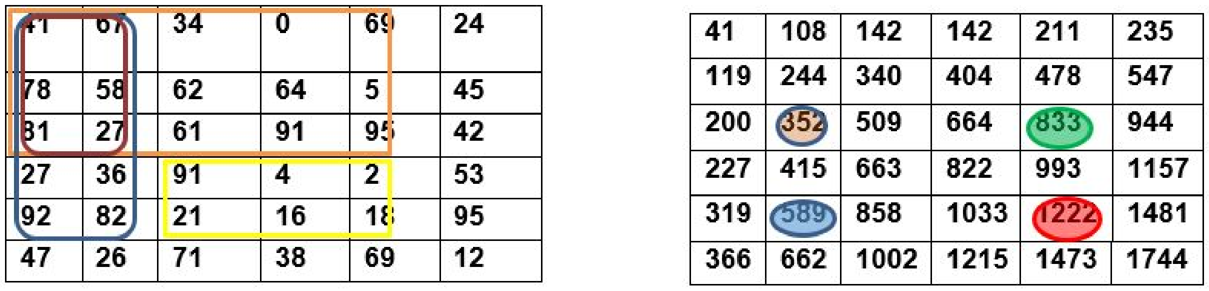


Fig 1. Types of Haar Features

To calculate a vast number of features, all possible sizes and location of the detection window need to be used. If we calculate features for all possible cases, it will take a lot of time, and a large amount of computation will be required.

3.1.1 Integral Images

Even for a very small detection window of 24*24, around 1,60,000 features needs be calculated. For each feature calculation, the sum of pixels under white and black rectangular window needs to be calculated.
To overcome this problem and perform our calculations fast, a technique known as Integral Images is used. The sum of the subset grid of any rectangular grid can be efficiently and quickly calculated using Integral Images. It is like a summed-area table.



41	67	34	0	69	24
78	58	62	64	5	45
81	27	61	91	95	42
27	36	91	4	2	53
92	82	21	16	18	95
47	26	71	38	69	12

41	108	142	142	211	235
119	244	340	404	478	547
200	352	509	664	833	944
227	415	663	822	993	1157
319	589	858	1033	1222	1481
366	662	1002	1215	1473	1744

Fig 2. Integral Image calculations

Integral sum of yellow part = 1222 - 833 - 589 + 352 = 152

In the above Images, Image 1 is the source grid, and Image 2 is the summation table. The summation of any sub-grid can be calculated, as shown in the figure.

Not all the features we calculated using the above technique are relevant. Most of them are Irrelevant. In the image shown below, two good features, Line and Edge features, have been considered. The chosen first feature (Edge feature) seems to focus on the property that the eye area is often darker than the nose and cheeks region. The second selected feature (Line Feature) is based on the property that the eyes are darker than the nose bridge. Nonetheless, the same windows used for cheeks or any other position are meaningless and irrelevant.

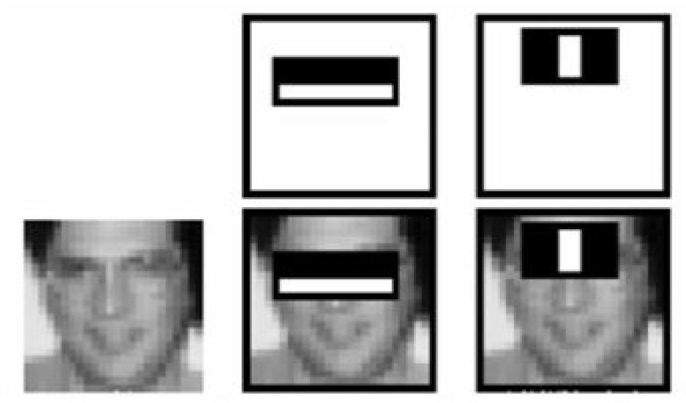


Fig 3. Feature Selection

Therefore we need to select the best features from all the features we have. To tackle this problem and to provide better results, the Adaboost algorithm is used. This algorithm creates a "strong" classifier as a linear combination of simple weighted "weak" classifiers.

3.1.2 Boosting

Boosting is a method of combining several weak classifiers' output to create one highly accurate robust classifier (committee). Adaptive Boosting, also called Adaboost successively, trains weak classifiers on weighted versions of the training data giving higher weights to currently misclassified instances, then combines these weak classifiers to generate a robust classifier.

A training set (x_1,y_1), … , (x_m,y_m) where $x_i \in \mathcal{X}$ the space of feature vectors extracted from the window of an image and $y_i \in \{-1, +1\}$, which is binary labels for positive and negative images, are input for Adaboost algorithm.

For $t = 1, \ldots, T$ this algorithm calls a weak learning algorithm repeatedly over a weighted training set.

For any iteration t, the weight of training element i is denoted by $D(i)$.

For the first iteration, $t = 1$ and for all i from 1 to m,

$$D_1(i) = \frac{1}{m}$$

After this, for each iteration, a weak hypothesis

$$h_t : \rightarrow \{-1, +1\}$$

is produced by training the weak learner over the weighted training set.

To calculate the error by this hypothesis, all the weights of misclassified elements of the training data is added.
Error in each iteration can be written as

$$\varepsilon_t = \sum_{i:h_t(x_i)!=y_i} D_t(i)$$

A coefficient $\alpha_t = \frac{1}{2} ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$ can be chosen for this hypothesis using the error value ε_t, which is inversely proportional to the error value.
Here $\alpha_t \geq 0$ if $\varepsilon_t \leq \frac{1}{2}$. (Error value must be less than half for any hypothesis to be better than random guessing).
The weights for the next iteration can be updated as

$$D_{t+1}(i) = \frac{D_t(i)\, e^{-\alpha_t y_i h_i(x_i)}}{Z_t}$$

Updated weight value insures to decrease the weights of correctly classified elements and increase the weights of incorrectly classified elements. This update of weights forces the next weak hypothesis to focus on elements that are incorrectly classified.

To make distribution of D_{t+1} sum up to 1, normalization factor Z_t is chosen.

where Z_t will be

$$Z_t = \sum_{i=1}^{m} D_t(i)\, e^{-\alpha_t y_i h_i(x_i)}$$

After T iterations we get the final hypothesis as

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x_i)\right)$$

The above hypothesis is strong, generated using T weak hypothesis with coefficients α_t.

3.1.3 Cascade Classifier

Over the input data (image), the classifier moves a window of fixed dimension during the detection process, and Haar features are determined for each segment of the data image. To separate objects from non-objects, the difference is compared with an already learned threshold. Cascade classifier is used to form a strong classifier from a large number of Haar features, which are necessary to detect an object with good accuracy because each Haar feature is a weak classifier.

Each stage of the cascade classifier has weak learners. Adaboost is used for the training of each stage. Each step of the classifier marks the area identified as either negative or positive for the current location of the sliding window. Positive is for the detection of object and negative is if the object is not detected. The classification of any particular window stops, and the slider is moved to the next location if the region of that window is identified as negative. If the last stage of the classifier detects any window as positive, then the classifier identifies it as an object.

We assume that there is no object of interest in most parts of the window; therefore, negative samples are rejected very fast. We are mainly interested in true positives, which are very rare also.

If a positive sample is incorrectly marked as negative, the situation is called the false negative. In cascade classifiers, if such a case occurs for a sample, classifications stop, and this mistake cannot be corrected. That’s why the rate of false-negative (object detected as non-object) must be low for our classifier. If a negative sample is labeled as positive (false-positive), that mistake can be fixed in the upcoming stages. Having more steps decreases the resultant false-positive rate, but it also lowers the true positive rate of classifier overall.

A large number of positive and negative samples are required to train Haar Cascade Classifier. Images with region specified as positive (region of interest) and negative (region out of interest) are needed to generate positive and negative samples. Haar Cascade is faster than normally used machine learning and deep learning algorithms like linear SVM and HOG (Histogram of Oriented Gradients). Haar cascade demands a lot of parameter tuning, and it could be less accurate.

HOG and linear SVM are typically more accurate than Haar Cascade because they generate fewer false positive.

Deep learning-based detectors can be very slow, but when trained correctly, they can outperform Haar cascade and HOG + Linear SVM in accuracy and robustness. Depending upon the depth and complexity of the model, using GPU inference, they can be sped up.

3.2 Landmark detection

Facial landmark identification is the task of identifying and monitoring main features on the face and at the same time being robust to various facial orientations due to movements and expressions. The proper identification of points of reference inside face images is crucial for a variety of computer-vision tasks such as blink detection. While it was an easy and simple task for human vision, decades of research and increased availability of quality data sets and a significant improvement in computer processing capacity have been necessary to achieve near-human precision in feature detection.

The positions between facial components and the facial contour catch rigid and non-stiff facial deformities due to the head movements and different expressions. They are, therefore, crucial for different tasks of facial analysis.

Finding face landmark constitutes of two main tasks :

Step #1: Detection of the face in the image.
Step #2: Detecting the principal facial structures

The first part has already been covered earlier. Also, the algorithm used for detecting a face in the live video does not cause any difference to the next task. The important task of this step is to get a region that includes a face by using Haar-Cascades, HOG, or any other technique.

Now that we have got a region that surely includes a face, we can start detecting the features of interest, which is mainly the region around the eye in our case.

Facial images that have been manually marked with specific coordinates around different facial structures are used as training images.

Positions of facial characteristics are found out using intensities of respective pixels by training a group of regression trees. Note that no feature extraction is taking place here.

The final result is a facial landmark detector, which can be used with high-quality predictions to identify facial features in real-time.

Dlib library consists of a facial landmark detector that is already trained. It can be used to map facial structures on the face.

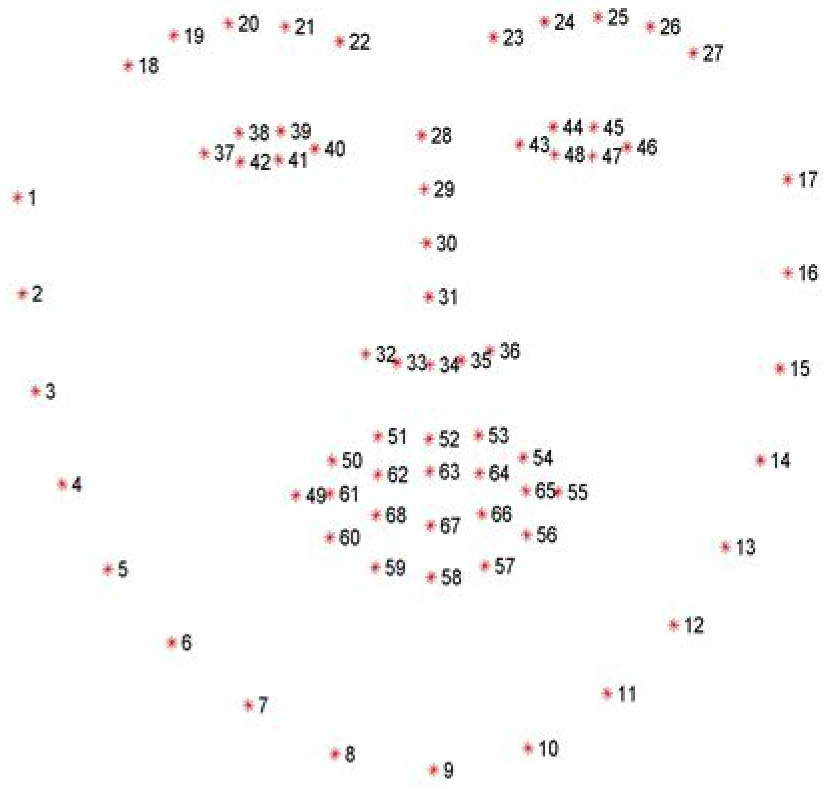


Fig 4. 68 points face mark-up

No matter what data set you're using, the very same dlib framework can be used to train a predictor of shape on your training data — this is helpful to train facial markers or even your customized shape predictors.

3.2.1 Detection of facial features

The dlib landmark detector returns a shape object with the 68 (x, y) coordinates of the regions with the facial feature. We converted this object to a NumPy array so that further implementation becomes easier. Now we start to work for the feature detection part.
But first, we have to identify the face in our input picture before we can detect facial landmarks. Once we get the locations of all the faces in the image, we can start applying facial landmark detection on each one of them

For every face detected in the video frame, we use facial landmark detection, which sends us 68 (x, y) coordinates that map the image with the particular facial features. We will then transform it into a NumPy array of shape (68, 2). A bounding rectangle is drawn around the detected face on the video frame. Then we loop over the detected facial landmarks and outline both the detected eyes.

By the end of this step, we had a detected face in the frame which was bounded by a green box, and both the eyes were marked by green contours.
Our next step would be to track the movement of eyes efficiently in the real-time scenario, which may include challenging situations like head movement or changing facial expressions.

3.2.2 Blink Detection

For the blink detection mechanism, we used the EAR metric, which is also termed as the eye aspect ratio. It was first introduced in [2]. In comparison to conventional image processing approaches that typically involve combinations of:

1. Eye localization.
2. Finding the white parts of the eye.
3. tracking if the white area of your eyes disappears for a period of time which indicates that a person has blinked

The aspect ratio is a very efficient solution requiring very simple calculations based on the distance ratio between the facial features of the faces. This method is simple, effective, and easy to implement for blink-eye detection. The first step of blink detection involves calculating the eye aspect ratio and using it to find whether a blink has occurred or not. Now we keep tracking facial landmarks and detecting the blinks. Since a person can also blink without intention, we held a certain threshold to differentiate them from the voluntary blinks.

3.2.3 Eye Aspect Ratio

To detect blinks, we are only concerned about two facial features, that is both the eyes.

We are only concerned about two sets of face structures when it comes to blink-detection – the eyes.
The 6 (x, y)-coordinates of each eye start on the left-hand side of the eye, then move in the direction of the clock around the rest of the region.

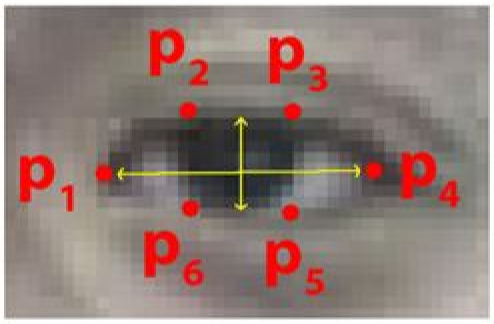


Fig 5. Eye Aspect Ratio

The width and height of these coordinates are related. A corresponding formula, the Eye Aspect Ratio (EAR), can be developed.

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

Where p1, …, p6 represents facial landmark locations in the 2-dimensional plane.

The numerator measured the distance between the vertical eye landmarks while the denominator calculated the distance between horizontal eye landmarks. The Denominator is normalized accordingly since there are two sets of points in the numerator but only one set of horizontal points

It can be seen that eye aspect ration will remain approximately constant when the eye is open, but as soon as the person closes them, EAR will rapidly fall to zero

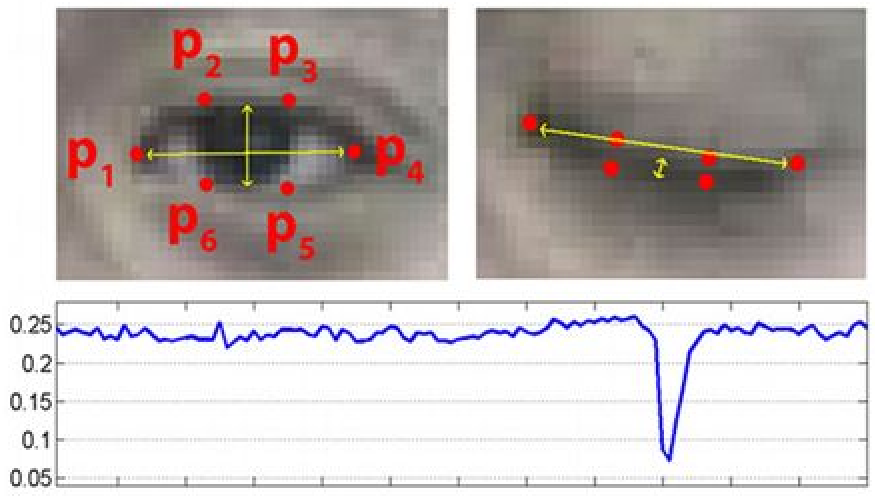


Fig 6. Change in Eye Aspect Ratio on blinking

There might be occasions when a user might have blinked naturally, and it was not intended at controlling the communicator, to handle such situations, a threshold is introduced which considers blink as voluntary only if the eye has been closed for more than a certain number of frames. So we have got a method to detect the voluntary blinks, and these will be used as inputs to control the communicator.

3.2.4 Mouth-Aspect-Ratio (MAR)

Motivated by this EAR feature, we tweaked the formula a little to get a metric that can detect open/closed mouth which has to be used to enable/disable mouse controls.

$$MAR = \frac{(||p2-p8|| + ||p3-p7|| + ||p4-p6||)}{(2||p1-p5||)}$$

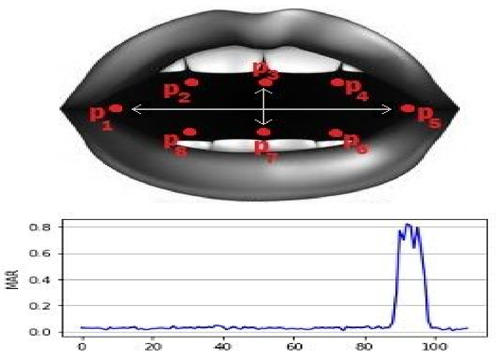


Fig 7. Mouth Aspect Ratio

4 Gaze Estimation

The direction of Gaze in the human eye can be estimated by the pupil and center of the eyeball where the center of the eyeball is unobservable in 2-D images. Gaze specific representation of this network architecture predicts a more accurate final output of 3-D gaze direction. This technique is inspired by the estimation of human pose where confidence or heat maps are being used where joint coordinates were directly regressed [7]. This helps in simplified mapping between the input image and joint position. Since the center of the eyeball can not be observed in 2-D image, this technique of heatmaps can not be applied directly for gaze estimation. Therefore this technique is using gazemaps for eye gaze estimation which is a pictorial representation of iris and pupil at the center of the eyeball. Rotation of an eyeball is depicted as an ellipse representing iris and a circle representing eyeball. Then the vector connecting the ellipse and the larger circle's center will define the direction of gaze. Thus a spherical eyeball as a circle and circular iris as an ellipse can be projected onto the image plane representing 3-D gaze direction. Hence the change in position of the ellipse will measure the change in gaze direction. However it is nontrivial to adopt the input image to suit our pictorial representation. An ellipse and a circular eyeball need to be fitted for provided data image and then rescaled and centered to the shape which is expected. Fully convolutional architecture can be used to perform this well.
This technique uses 'Gazemaps' as pictorial representation for 3-D gaze estimation. Two boolean maps are used for this representation and fully convolutional neural networks regress them.

4.1 Pictorial Representation of 3-D Gaze :

To provide direction of gaze in 3-D, the image of the eye as input data is processed for gaze estimation based on appearance. The direction of gaze is usually represented as a unit vector v of three elements or as a combination of two angles $g = (\theta, \phi)$ where the first one represents the pitch of the eyeball and the latter one represents yaw. This technique uses indirect mapping to g or v.

If input image of eye is set to x and values after regression is set to g, a conventional model for gaze estimation will be mapping $f : x \rightarrow g$. State of the art convolutional neural network architecture for gaze estimation which depends on appearance are LeNet-5, VGG-16, and AlexNet. By simply implementing newer CNN architectures, the improvement achieved in accuracies shows that mapping f is very complex. This technique introduces an intermediate representation of the input image of the eye, m. This new model is defined as

$$g = k \, o \, j(x)$$

where $k : m \rightarrow g$

and $j : x \rightarrow m$

This reduces the learning complexity of $j \;\&\; k$ significantly than the previous case where f was directly learned because of the application of neural network architecture with reduced complexity of model for the same work of gaze estimation with equivalent efficiency.

This intermediate representation is called gazemaps (m). We estimate these gaze maps (m) and 3-D gaze direction (g) from that. This job is reformulated into two steps :

(a) Reducing input eye image into gazemaps (minimum normalized form)
(b) Estimation of gaze using previously estimated gazemaps

To make sure that model $k : m \rightarrow g$ is easy, gazemaps estimated from the provided input image should be perceptibly close to input image, but just filters the details required to estimate the gaze. To achieve this, it is considered that the diameter of the average human eyeball is 24 mm [10] and the diameter of the average human iris is 12mm [11].

Then this model considers iris as a perfect circle and eyeball as a perfect sphere. The diameter of the eyeball is considered as $2r = 1.2n$. The dimension for this output image is $m * n$. Coordinates of the center of the iris (u_i, v_i) would be calculated as :

$$u_i = \frac{m}{2} - r' \sin \phi \cos \theta$$

$$v_i = \frac{n}{2} - r' \sin\theta$$

where $r' = r\cos\left(\sin^{-1} \frac{1}{2}\right)$,

$g = (\theta, \phi)$ is the direction of gaze.

The iris in the output image is considered as an ellipse in which the diameter of minor axis is $r|\cos(\phi)\cos(\theta)|$ and the diameter of major axis is r. Below Figure shows the estimated gazemaps where for one gaze direction g , two different boolean maps are generated.

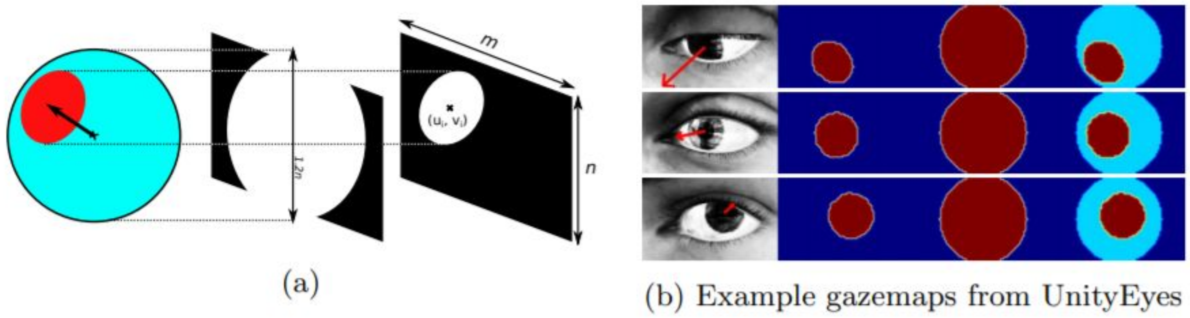


Fig 8. 3D gaze direction

It is no trivial job to predict gazemaps where only single input image of eye is used. There are several factors that need to be accounted for like partial occlusion, image artifacts, and also a simplified eyeball must match the provided input image based on the appearance of eyelid and iris. In order to generate the gazemaps, scaling should be done for the detected regions and then it should be centered. Therefore model $j : x \to m$ needs a more complex architecture than model $k : m \to g$.

4.2 Neural Network Architecture

This technique has two main steps for gaze estimation using neural networks:

(a) Interpreting gazemaps from input eye image using regression
(b) Estimation of gaze direction g from these gazemaps using regression

Second step can be easily implemented using any CNN architecture but the first step needs a fully convolutional architecture for regression which has been used in the estimation of human pose. Stacked hourglass architecture from Newell *et al.* [12] has been adopted to implement this. For estimation of human pose or detection of facial landmarks [13] where composite structural relationships required to be determined at different scales for estimation of the location of key points or occluded joints, this hourglass architecture has shown good results and is effective. For feature maps, multi-scale refinement is performed repeatedly in hourglass architecture, from which 1*1 convolutional layers can be used to extract required output confidence maps. Instead of heatmaps or classical confidence for positions of joint, this technique takes advantage of this fact and predict gazemaps using this network.

This uses a three hourglass module in a network of gazemap-regression with intermediate control applied only to the last module's gazemap outputs. Minimized intermediate loss can be calculated as :

$$L_{gazemap} = -\alpha \sum_{\rho \in P} m(\rho) \, log \, \widehat{m}(\rho)$$

where for ρ in the set P a cross-entropy between ground truth gazemap m and predicted $\widehat{m}$ is calculated. The value of the weight coefficient α is set to 10^{-5}. DenseNet is used for regression from gazemap to gaze direction g. Since it uses fewer parameters than previously used techniques like ResNet, it performs better in tasks of classification of the image. The loss for regression of gaze direction(per unit) can be calculated as :

$$L_{gaze} = ||g - \widehat{g}||_2^2,$$

where $\widehat{g}$ is the predicted direction of gaze.

4.3 Implementation of Gaze Estimation

4.3.1 Hourglass Network

Size of images is 150*90 which have been provided as input and throughout the network, size of 64 feature map is 75*45 which have been refined to implement this stacked hourglass network [12]. An initial convolution layer with stride 2 and filter size 7 is used to produce half-scale feature maps. To accompany this, batch normalization, activation of ReLU, and two residual modules before it is fed into hourglass network as input is followed.

This architecture contains 3 modules of hourglass. For the estimation of human pose, confidence maps of 2-dimension are common outputs, which are aligned in pixels with the input image. The task of gaze estimation is different from human pose estimation therefore this doesn't supervise the performance of every hourglass module intermediately. This allows multiple-scale processing of the input image over several layers, with the required features matched with the final result of gazemap representation. 1*1 convolutions are applied instead of this after the last hourglass module and then gazemap loss term is applied.

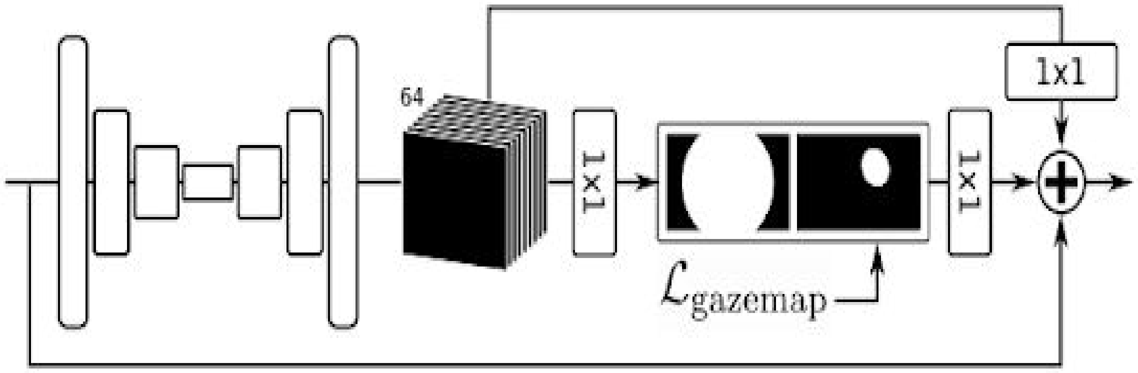


Fig 9. Supervision at output of hourglass

4.3.2 DenseNet

This pictorial representation makes it possible to learn a simplified function for the actual task of estimation of gaze. A very lightweight architecture of DenseNet is used to prove this. This regression network for gaze estimation consists of 0.5 compression factor, bottleneck layers, and per block 5 layers with 8 point rate of growth. As a result of this, at the end of DenseNet, this gets 62 feature maps and 62 features afterward by average global pooling. Finally, these features are mapped with g using a single linear layer. The final network is lightweight and contains 66,000 trainable parameters.

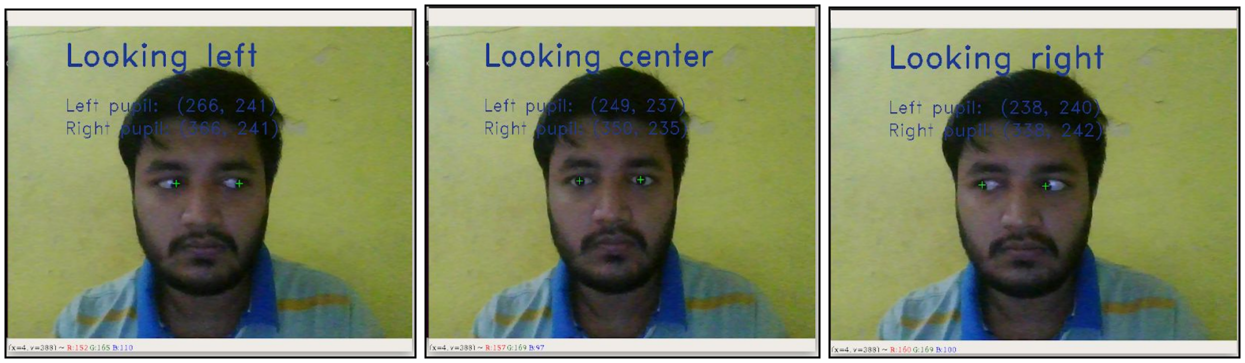


Fig 10. Tracking Gaze

4.4 Gaze point Heatmap

Heat maps and gaze plots are visualizations of the data that can simply and effectively explain essential aspects of visual behavior. Their proper and successful usage includes an understanding of the aims of eye-tracking study, as well as clarification as to the results one wishes to communicate.

Gaze plots display the location, ordering and time taken looking at the trigger sites, whether web page, print ads, or clip. So the gaze plot's primary purpose is to show the time series in which we look and where we gaze and when do we gaze there. The time spent looking is indicated by the diameter of the fixation rings, most generally represented as fixation length. The longer the gaze, the greater the circle.

They have their drawbacks, no matter how useful gaze plots can be. As static visualizations, they are most impactful when only one or two participants' data are shown as the display may become cluttered with more than a few overlaying the product image.
Heat maps display how the look over the stimulus is distributed. Unlike the gaze plot, in a static heat map, there is no detail about the order of look. The emphasis is not on individual fixations either. Alternatively, heat maps are a visualization that can easily show to dozens or even hundreds of observers at a time the object of visual attention. For a long time, heat maps were the eye-tracking poster boy. Broadly used and often misused, this visualization is rarely the most important output from an analysis of eye-tracking. And if built and interpreted incorrectly, heat maps can do more to confuse and deceive than to guide decision making. The crucial first step is to select the right basis for constructing a heat map. And to do so correctly, the research purpose and the argument to be made with the analysis must be taken into account.

In our case, The study involves changing keyboard layout to increase understanding and to facilitate a better rate of response to the action performed for a particular key click. Redesign requires manipulation that is intended to influence a cognitive mechanism, namely effort (in typing), fixation duration being the correct measure for eye tracking. In this case, for the current and proposed designs, we will create duration-based heat maps.

To understand which part of the screen users have viewed the most, we created a heatmap of similar dimensions as of screen. This data will help us in the future to create a more optimized keyboard layout that can be used more easily. The point where users have spent the most time gazing can be used to project the most frequently used keys. For generating the heatmap, our program takes a CSV file as an input where each line consists of three values: X-coordinate, Y-coordinate, and the duration for which it has been looked upon.

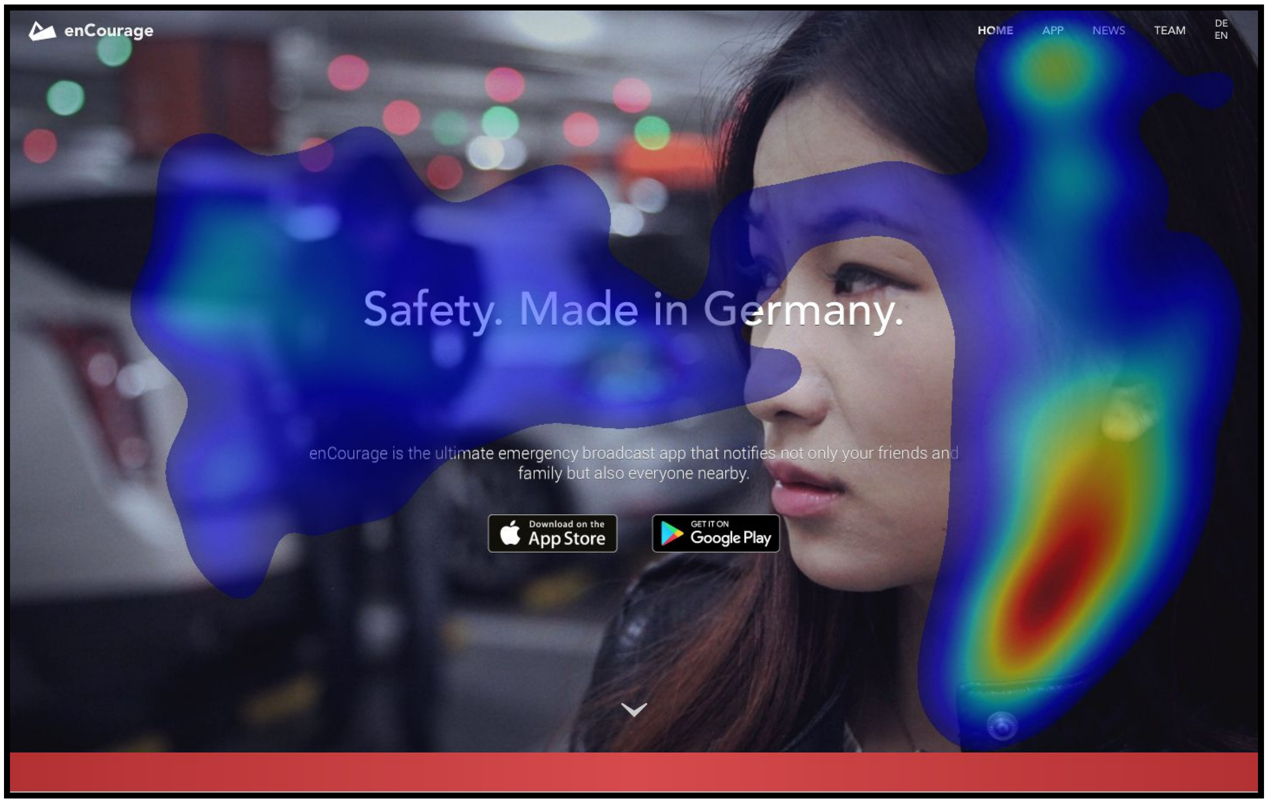

enCourage
HOME
APP
NEWS
TEAM
DE
EN
Safety. Made in Germany.
enCourage is the ultimate emergency broadcast app that notifies not only your friends and family but also everyone nearby.
Download on the App Store
GET IT ON Google Play

Fig 11. Gaze Heatmap

5 Communicator controls

The main window of our communicator consists of the 26 alphabet tabs, Space and backspace key, a word predicting panel, and white space, which contains all the written sentences. The layout of these 26 alphabets is not the same as a usual qwerty keypad. It is presented as a matrix with groups of consecutive alphabets sequentially broken into rows. A row determining sliding bar keeps moving across these rows until a voluntary eye blink is detected. When a blink is detected, that particular row is selected, and a column determining bar start sliding across all the letters of the selected row. On detection of the second blink, the corresponding letter is selected and displayed in the display bar. This same mechanism is followed for writing the complete sentence.

To further ease the process, a word predicting panel is also programmed so that after writing a few letters, the complete word can be directly selected from the provided prediction. Space and backspace keys can be used for the respective functions by a similar mechanism that was used for letter selection.

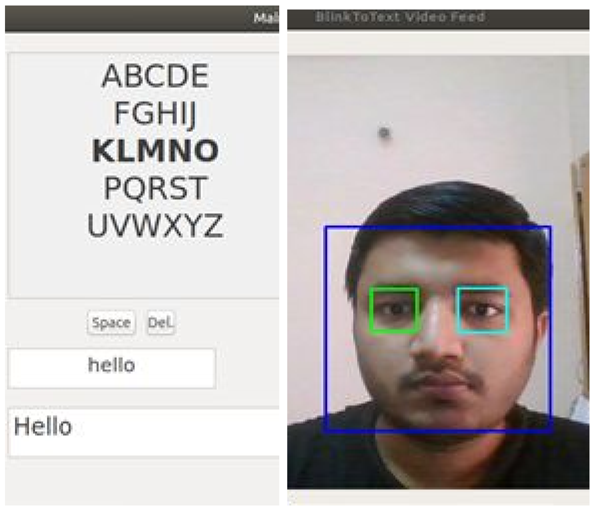


Fig 12. (a) Detection feedback frame (b) Communication control window

As an extension to the features of this keyboard, we included several extra controlling mechanisms such as mouse controls, left and right clicks, and scroller.

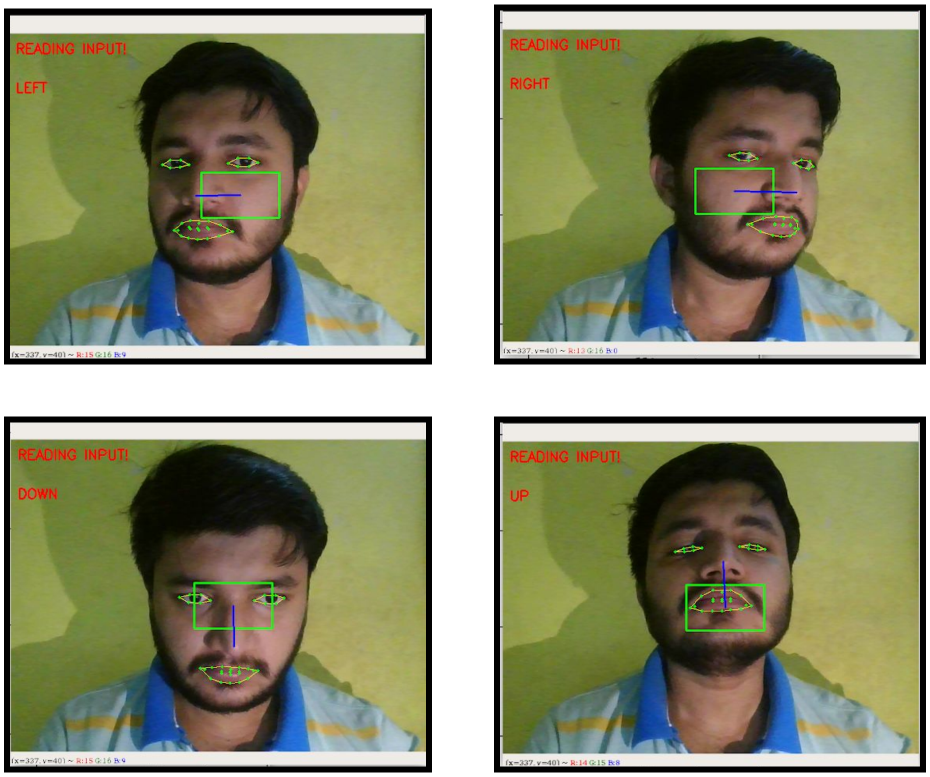


Fig 13. Mouse Pointer Controls

Providing multiple functionalities using only eye blinks is difficult and might lead to confusion therefore we provided a way by which a user can decide which operation he has to perform and can enable that particular functionality at that point in time. Grouping similar functions and enabling only one cluster at a time allowed us to reuse similar inputs like right or left wink for different operations.

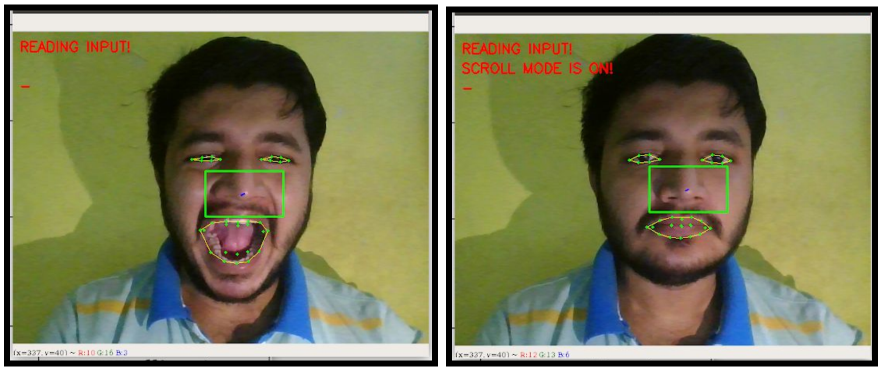


Fig 14. Activating Mouse Control

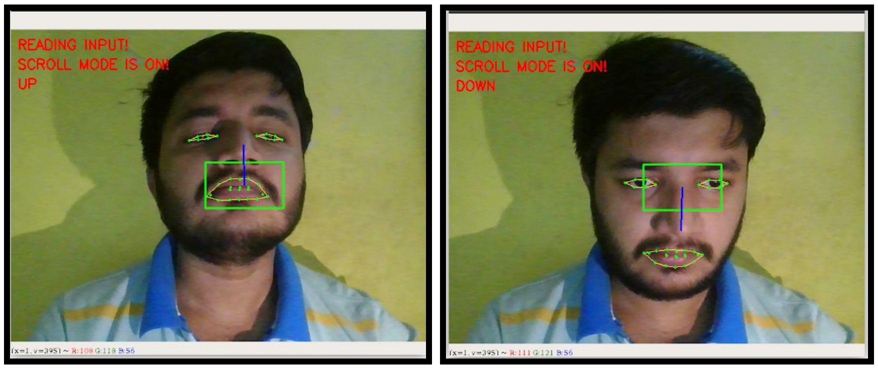


Fig 15. Scroller Controls

The keyboard functions will work normally as explained at the start of this section and if you wish to use the extended functionalities, you can activate them by providing related gestures. If you have to use mouse, opening of mouth will signal the program to enable or disable mouse options. Once the mouse option is selected you can move the tip of the nose in whichever direction you want to move the cursor. Right, and Left clicks can be performed by right and left eye blinks respectively. Scrolling function can be enabled and disabled by Squinting your eyes (The way we look in bright sunlight, with the eyes partly closed). Once it is enabled, you can use Head movements to scroll the current page accordingly.

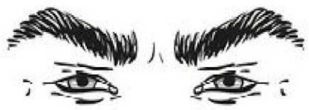
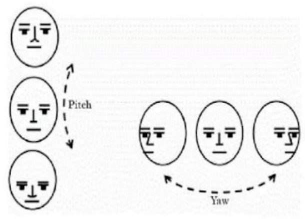
Action	Function
Opening Mouth	Activate / Deactivate Mouse Control
Right Eye Wink	Right Click
Left Eye Wink	Left Click
 Squinting Eyes	Activate / Deactivate Scrolling
 Head Movements (Pitch and Yaw)	Scrolling / Cursor Movement

Fig 16. Control Gestures

6 Observations

To calculate the number of words a user can write per minute we placed the laptop in front of the user's face at a distance of 60 c.m. We started the software and timer for 5 minutes and noted the number of words a user could write in the given interval. We repeated this process 5 times and calculated the average number of words the user can type

	Test 1	**Test 2**	**Test 3**	**Test 4**	**Test 5**
No. of words typed in 5 mins.	**9**	**10**	**8**	**10**	**7**
Words per minute	**1.8**	**2**	**1.6**	**2**	**1.4**

It was observed that using Landmark detection provided better results than other algorithms. It was able to detect blinks for a wide range of face orientations. As shown in the images below, both eyes are detected flawlessly which was not the case for other algorithms.

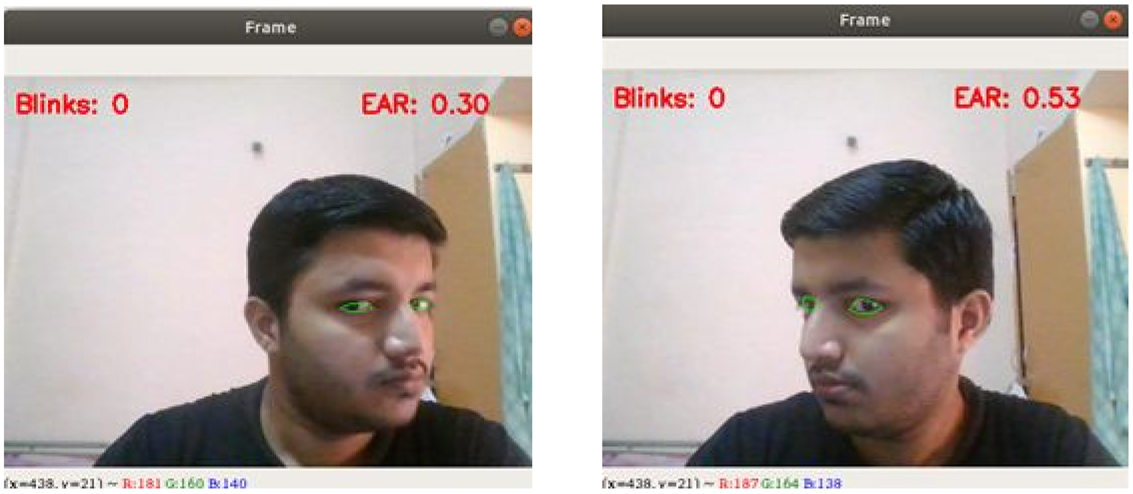

Frame
Blinks: 0
EAR: 0.30
(x=438, y=21) ~ R:181 G:160 B:140
Frame
Blinks: 0
EAR: 0.53
(x=438, y=21) ~ R:187 G:164 B:138

Fig 17. Eye detection at wide angles

7 Conclusion

This software can provide an easy way of communication with disabled people who are not able to move parts of their body voluntarily. With the help of this software, these people can express themselves without the help of others which makes them more independent. Since this software doesn't need any high-end hardware or costly devices to operate, it could be made available to people with a poor financial background also.
Knowing how and when people look is essential to understanding the distribution of attention. Eye-tracking is commonly used in psychological experiments such as implicit connection test, Iowa Gambling Assignment, as well as in paradigms of glance contingency.

In medical settings, it may also be necessary to monitor an individual's gaze. Research findings have shown the possible predictive strength of eye tracking in autism treatment, and in many other neurological conditions. Future uses in healthcare settings will be seeing the application of eye-tracking data in providing optimum patient care.

For many years, following the styles of gaze while people shop has been a hot force within neuromarketing. In order to incorporate optimum package design, shop layout, and point-of-sale presentations, being able to capture what consumers attend to or neglect can be critical.

This tracking data can provide important information into your website visitors' gaze patterns – how long does it take them to find a specific goods on your site, what kind of visual information do they disregard but should respond to. How should consumers look at your website? What are they gazing at, and just how much time do they spend watching?

8 Future Work

Algorithms to detect blink of the eye could be more robust and accurate. In our research, we found that Eyeblink detection using facial landmarks and eye aspect ratio works more accurately than using Haar features and cascade classifiers.

A working commercial platform with features like sending written sentences to other devices and many more could be launched in future to make it more user-friendly Working in a web browser and using any new application are the activities which can be made reachable with detection of eye blink in the future.

Not just the platform, but the underlying tech can be used to create a diverse range of tracking platforms each catering to the need of a certain kind of customer base. There can be target-specific applications to provide better business understanding in using neuro-marketing, Psychological surveys and methods, and following consumer patterns. Looking at the business point of view eye tracking can reveal:

- What people see on a computer or in the actual world
- When such visual elements obtain the attention
- How long every attachment lasts
- The sequence wherein these visual elements are received by the looker
- When a person's attention returns to a visual item that was explored before

9 Bibliography

1. Paul Viola, Michael Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001

2. Tereza Soukupová, Jan Čech, “Real-Time Eye Blink Detection Using Facial Landmarks,” 2016

3. Vahid Kazemi, Josephine Sullivan, “One Millisecond Face Alignment with an Ensemble of Regression Trees,” IEEE Conference on Computer Vision and Pattern Recognition, 2014

4. Ahmed Reda Amin EL-Barkouky, “Mathematical modeling for partial object detection,” University of Louisville, 2014

5. Docs, OpenCV. “Face Detection Using Haar Cascades.” OpenCV: Face Detection Using Haar Cascades, 4 Aug. 2017

6. Mathworks, Mathworks. “Train a Cascade Object Detector” Mathworks, 2017

7. Toshev, A., Szegedy, C.: Deeppose: Human pose estimation via deep neural networks. In: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1653–1660. CVPR ’14, IEEE Computer Society, Washington, DC, USA (2014). https://doi.org/10.1109/CVPR.2014.214

8. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: European Conference on Computer Vision. pp. 483–499. Springer (2016)

9. Zafeiriou, S., Trigeorgis, G., Chrysos, G., Deng, J., Shen, J.: The menpo facial landmark localisation challenge: A step towards the solution. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (July 2017)

10. Bekerman, I., Gottlieb, P., Vaiman, M.: Variations in eyeball diameters of the healthy adults. Journal of ophthalmology 2014 (2014)

11. Forrester, J.V., Dick, A.D., McMenamin, P.G., Roberts, F., Pearlman, E.: The Eye E-Book: Basic Sciences in Practice. Elsevier Health Sciences (2015)

12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)

13. Honari, S., Molchanov, P., Tyree, S., Vincent, P., Pal, C., Kautz, J.: Improving landmark localization with semi-supervised learning. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)