# Virtual Keyboard

Avish Kabra, Chandan Agrawal and M.K Bhuyan

Department of Electronics and Electrical Engineering

## Abstract

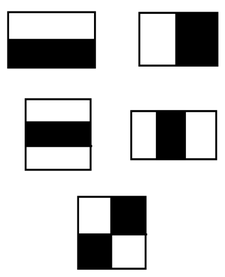
Our project provides a vision-based human-computer interface. The interface senses and interprets eye-blinks as controls. We used image processing methods like haar-like features for automatic face detection and eye tracking and blink detection based on landmarks. Voluntary eye blinks are differentiated 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 predicting next possible words.

## Introduction

Spinal cord injury is one of the most common reasons for paralysis. 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. We programmed a computer-human interface with an understanding of blink detection. It is then applied to the communicating device, namely a virtual keyboard, which is explicitly designed for the physically disabled. They can use the proposed mechanism by merely blinking their eyes, enabling them to communicate with the outside world.

## Haar Cascades

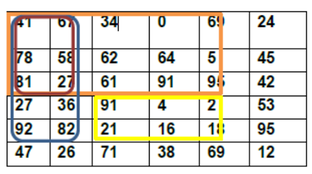
Haar feature-based cascade classifier is a machine learning-based approach that is used for object detection. It is a very effective method that uses a lot of positive and negative images to train a cascade function, which is used to identify objects in images. A Haar feature considers neighbouring rectangular regions in a detection window at a specific location (White and black rectangular regions), add up the intensity of pixels in each rectangular window, and measures their difference. 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.



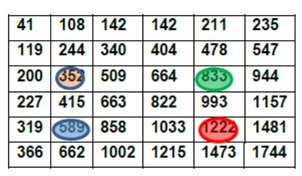
(a) Edge Features

(b) Line Features

(c) Four Rectangle Features



| | | | | | |
|---|---|---|---|---|---|
| 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 |

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

Not all the features we calculated using the above technique are relevant. In the image shown here, Line and Edge features, have been considered. the same type window used for chins or other parts of the face is meaningless and irrelevant. Therefore, we need to select the best features from all the features we have.

## Boosting

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 $(x_1, y_1), \ldots, (x_m, y_m)$ 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 any iteration $t$, the weight of training element $i$ is denoted by $D_t(i)$.

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

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

Error in each iteration can be written as

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

The weights for the next iteration can be updated as

$$D_{t+1}(i) = \frac{D_t(i)\ e^{-\alpha_t y_i h_t(x_i)}}{\sum_{i=1}^{m} D_t(i) e^{-\alpha_t y_i h_t(x_i)}}$$

Updated weight value decreases 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.

After $T$ iterations we get the final hypothesis as

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

## 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. Finding face landmark constitutes of two main tasks:

**Task 1**: Detection of the face in the present video frame.

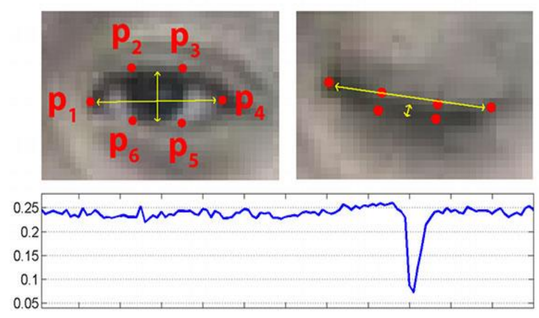
**Task 2**: Detecting principal facial features.

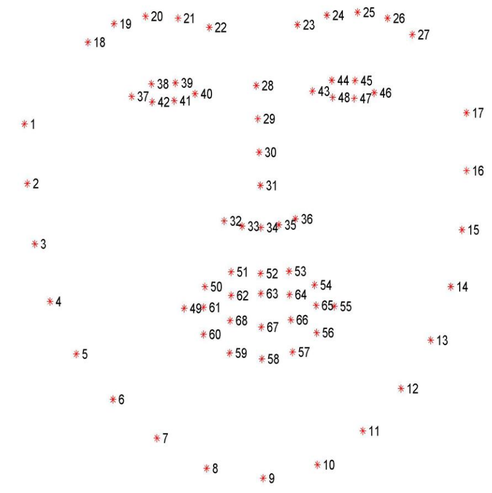
Now that we have got a region that surely includes a face, we can start detecting the features of interest, which is mainly the region around the eye in our case. Facial images that have been manually marked with specific coordinates around different facial structures are used as training images. Positions 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.

## Blink Detection

For the blink detection mechanism, we used the EAR metric, which is also termed as the eye aspect ratio. Blink Detection involves of three main steps - Eye localization, Finding the white parts of the eye, tracking if the white area of your eyes disappears for a period of time which indicates that a person has blinked.

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

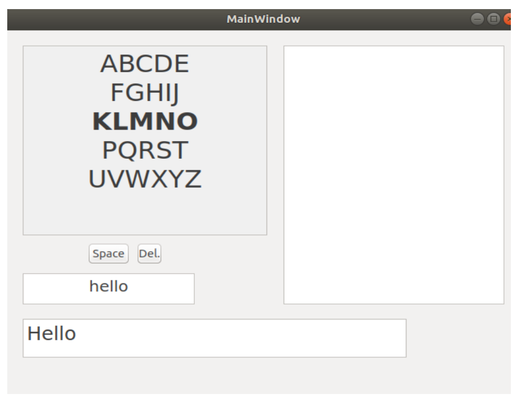


$$\text{EAR} = \frac{||p2 - p6|| + ||p3 - p5||}{2||p1 - p4||}$$

## Communicator Controls

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.




## Conclusion and Future Work

Eyeblink detection using facial landmarks and eye aspect ratio works more accurately than Haar features and cascade classifiers. Using facial landmarks, eyes could be detected even when the face was rotated till 30° from the front plane. Users can write an average of 1.76 words per minute using this application. This software can provide an easy way of communication for disabled people who are not able to move parts of their body voluntarily. Features like sending the written sentence to other devices and many more could be added in the application to make it more user friendly in the future.