Blink to Text-A Real Time Blink Detection and Text Displaying Application

Arnav Sahai
CSE Department
Maharaja Surajmal
Institute of Technology
Janakpuri, New Delhi

Atishay Jain
CSE Department
Maharaja Surajmal
Institute of Technology
Janakpuri, New Delhi

Ayush Rawat
CSE Department
Maharaja Surajmal
Institute of Technology
Janakpuri, New Delhi

Harshit Kumar
CSE Department
Maharaja Surajmal
Institute of Technology
Janakpuri, New Delhi

***Abstract*—This paper looks at the concern of patients suffering from Locked in Syndrome (LIS). Many people suffering from this disease are cognitively aware but are unable to effectively communicate their thoughts or ideas, they have lost their ability to talk, type etc. The goal is to build an application that aids these patients suffering from LIS which is a disease in which the patient loses all his motor abilities except the ability to blink, this application aids in communication by detecting blinks in real time and converting these blinks into text using an effective Graphic User Interface (GUI)**

Keywords—LIS,GUI,Haar Cascade,HOG,SVM,OpenCV,dlib

I. INTRODUCTION

A. The Problem

Certain unfortunate people lose their ability to control muscle function due to the onset of various reasons such as ALS, multiple sclerosis, and other diseases. This condition can exacerbate into LIS or better known as Locked-in-Syndrome which is a form of paralysis affecting nearly all voluntary muscles. People affected by LIS are only left with the movement of eyes and blinking. Their ability to talk, text, move and even communicate is robbed of them. Even though being cognitively aware these people's thoughts remain lost forever in a limbo without being able to be materialised. Their dependence over caretakers for communication takes a major role to interpret and understand their blinking i.e. they need external help for self-expression.

B. The Solution

BlinkToText aims at tackling the above problem and provides a way for self-expression to the people suffering from LIS. The software aims to convert eye blinks to text through an interactive graphical user interface wherein every feature can be controlled through eye movement and blinks. This relieves the patient's dependence upon caretakers for communication and can be independently operated by them. The software can be run on any low-end computer. The software uses computer vision and Histogram of Gradients(HOG) + Support Vector Machine(SVM) to detect eye blinking and convert the motion into text. The software also uses eye-aspect-ratio(Ear) to determine whether eyes are closed or open.

II. KEY OBJECTIVES

A. Allow paralysis victims to communicate independently

Many paralysis victims already use eye blinks as a form of communication. It is common for nurses and caretakers to read a patient's eye blinks and decode the pattern. The ALS association even offers a communication guide that relies on eye blinks. BlinkToText automates this task. The software reads a person's eye blinks and converts them into text. A key feature of the software is that it can be started, paused, and operated entirely with eye blinks. This allows patients to record their thoughts with complete independence. No nurses or caretakers are required to help patients express themselves. Not only does this reduce the financial burden on paralysis patients, but this form of independence can be morally uplifting as well.

B. Be accessible to people with financial contraints

Many companies are developing technologies that are controlled by eye movement. These technologies rely on expensive hardware to track a user`s eyes. While these devices can absolutely help LIS victims, they are only available to people that can afford the technology. BlinkToText focuses on a different demographic that are often ignored. BlinkToText is free and open source. The software runs on wide variety of low-end computers. The only required peripheral is a basic webcam.

Not only is this software accessible to paralyzed people, but paralyzed people of almost all financial classes as well.

III. BACKGROUND STUDY

A. *Face Detection*

The process of face detection however easy for a human mind whether it be the detection of faces in still images, dynamic videos or in real time is quite difficult for computed systems, this difficulty associated with face detection can be attributed to variations in scale, illumination, location, view point, occlusions etc. For the sake of simplicity and to tackle the aforementioned problem a study was conducted as presented in [1] on Viola-Jones face detection method and HOG + SVM detection method to determine which is more suited for our application.

1. Viola -Jones Detection Technique

As well explained in [2] this technique is divided into three main integral parts. The first is the creation of new image representation called the "Integral Image" which allows for better and faster feature extraction. Integral Image also known as a summed area table is an algorithm for quickly and efficiently computing the sum of values in a rectangular subset of pixel grid. The integral image at location *x, y* contains the sum of pixels above and to the left of *x, y.*

$ii\,(x, y) = \sum_{x' \le x, y' \le y} i(x', y')$

where $ii\,(x, y)$ is the integral image and $i\,(x, y)$ is the original image.

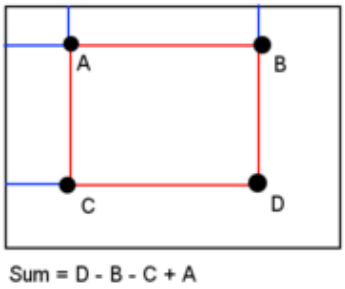


Figure 1: Once the summed-area table has been computed, evaluating the sum of intensities over any rectangular area requires exactly four array references regardless of the area size. That is, the notation in the figure at right, having A= (x_0, y_0), B= (x_1, y_0), C= (x_0, y_1) and D= (x_1, y_1), the sum of $i\,(x, y)$ over the rectangle spanned by A, B, C and D is:

$$\sum_{x_0 \le x \le x_1, y_0 \le y \le y_1} i(x, y) = I(D) + I(A) - I(B) - I(C) \qquad (1)$$

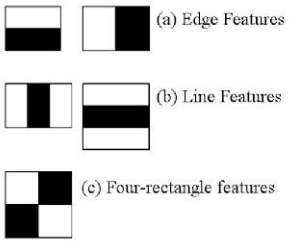


Figure 2:For the creation of rectangle features the sum of the pixels in white area are subtracted from sum of the pixels in the dark area..Rectangle feature two, three and four are depicted by (a),(b) and(c) respectively.

The second integral part is the use of learning algorithm AdaBoost which selects a small number of critical visual features from a larger set and yields efficient classifiers. This is because an image sub window has a plethora of Haar like features and only a few are important or needed to be used to detect a face. The features being talked about in this paper are rectangular features and there are many reasons to use them rather than pixels directly, the most common being that features can act to encode ad-hoc domain knowledge which is difficult to learn using a finite quantity of training data set.

The features that are calculated are basically the following with each having identical regions in shape and size:(See Figure 2) Firstly a two rectangle feature where difference between the sum of the pixels of two rectangular region is computed. Secondly in a three rectangle feature the sum of pixels from outer rectangle subtracted from the middle rectangle region and lastly in four rectangle feature the difference is calculated between diagonal pairs of rectangle.

For detection of faces Ada Boost uses well-founded and tenable features such as the region of eyes is darker than the region of nose and cheeks or the region of eyes are darker than the bridge of nose (See Figure 3). The weak learning algorithm of AdaBoost is designed to select that single rectangle feature which is effectively able to distinguish between positive and negative samples. For each feature that the weak learner encounters it determines the best threshold classification function, so the least number of enables are misclassified. The threshold classification function is as follows:

$$h_j(x) = \begin{cases} 1 \; if \; p_j f_j(x) < p_j \theta_j \\ 0 \; otherwise \end{cases} \qquad (2)$$

Where f_j is feature, p_j is polarity, θ_j is threshold

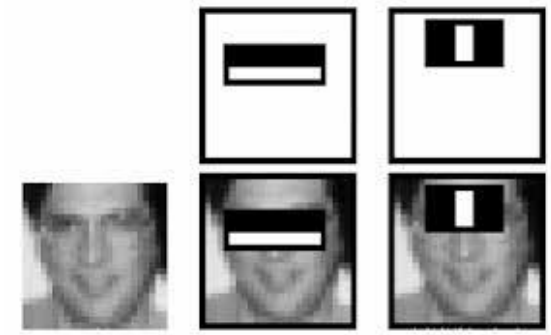


Figure 3: Meaningful features selected by AdaBoost Algorithm.

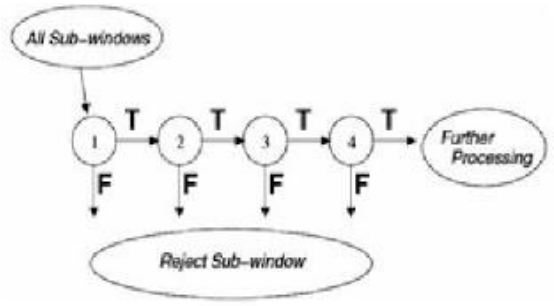


Figure 4: Schematic depiction of the detection cascade. This shows a sub-window is passed through a series of classifiers where large number of negative examples are eliminated by initial classifier which reduces the number of sub-windows drastically after several stages of processing. Also, with the addition of subsequent layers the computational time increases.

The third integral feature of Viola Jones Detection is a method of combining successively more complex classifiers in a "cascade" structure which precipitates the speed of detector by paying attention to only those regions in an image where the probability of finding the object in consideration, in our case faces is highest. This is based on the understanding that small and more efficient classifiers can be constructed which reject many of the negative sub-windows while detecting almost all positive instances. These simpler classifiers reject the majority of sub windows before more complex classifiers can be called to achieve low false positive rates.

The overall detection process is similar to a degenerate decision tree, which is called a "cascade" (See Figure 4). A positive outcome from the first classifier initiates the detection process by second classifier which in turn is tuned to different adjustment so as to achieve high detection rates. A positive response from second classifier similarly triggers a third classifier and so on. A negative outcome at any point leads to the immediate rejection of the sub-window. Each stage in the cascade is constructed by training classifiers using AdaBoost and then adjusting their threshold to minimise false positive rates. The fact that within every image an overwhelming majority of sub windows are negative is effectively reflected by the structure of cascade. Like a decision tree every subsequent classifier are trained on examples which pass through the preceding classifier, this makes these examples much "harder" than the typical examples.

2. HOG + SVM Detection Technique

As well explained in [3] the method evaluated well-normalized histograms of image gradient orientation into a dense grid. The main idea is that local object appearance and shape can be frequently described effectively by the distribution of local intensity gradients or edge directions without even having any knowledge about the corresponding gradient or edge positions. In this technique the image window is divided into small spatial regions("cells"), for each cell accumulating a local 1-D histogram of gradients or edge orientations over the pixels of the cell. These histograms are combined to form a representation. For independence from illumination, shadowing etc. such that they don't affect in detection the local responses are contrast normalized before using them which is done by accumulating a measure of local histogram "energy", over somewhat larger spatial regions ("blocks") and using the results to normalize all the cells in the block. Covering the detection window with a dense and overlapping grid of HOG descriptors along with using combined feature vector in a conventional SVM based window classifier gives the below illustrated chain (See Figure 5).

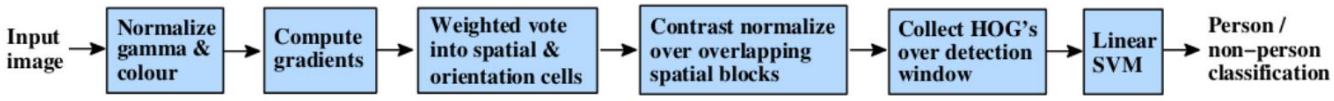


Figure 5: The entire process of HOG Detection where detector window which scans across the image for all position and scales, is covered with a grid of overlapping blocks in which HOG vectors are calculated and extracted. These vectors are then fed into a linear SVM for object/non-object classification.

Table 1. Expression and occlusion variations of 300-W databases

Database	'Neutral'	'Surprise'	'Squint'	'Smile'	'Disgust'	'Scream'	'Occlusion -Yes'	'Occlusion -No'	Landmarks
300-W	37.17%	12.34%	4.84%	29.83%	1.66%	14.16%	29.83%	70.17%	68

B. Facial Feature Detection

The task of facial feature detection was carried out by a pre-trained facial landmark detector included in the dlib library. The model is trained over iBUG 300-W dataset[4] and is an implementation of Face Alignment with an ensemble of Regression Trees [5].

1. iBUG 300-W Dataset

The data set used to train the detector was aimed at determining how effective any current systems or methods are at detecting facial features and are able to handle naturalistic, unconstrained face images. The data set should cover variations like occlusion, image quality, illumination, expression, background, unseen subjects, pose and expression. The data set images should also include images where the subjects can be seen showing other expressions aside from the generic smile. Thus, to incorporate all of this a data set was created consisting of 300 'Indoor' and 300 'Outdoor' images, respectively. The images were further classified into categories as can been seen from Table 1 by assigning tags like 'Neutral', 'Surprise', 'Squint', 'Smile', 'Disgust', 'Scream' and about Occlusion too.

2. Face Alignment with an ensemble of Regression Trees

This technique uses a collection of regression trees to estimate the face's landmark positions directly from a sparse subset of pixel intensities often achieving high performance and quality predictions. A general framework is presented that optimizes the sum of square error loss and naturally handles missing or partially labelled data, which is based on gradient boosting for learning an ensemble of regression trees.
The problem of face alignment is solved with the use of cascade of regression functions where each function in the cascade efficiently estimates the shape from an initial estimate and the intensities of a sparse set of pixels indexed relative to this initial estimate. Two key elements are incorporated into learnt regression functions. The first which revolves around indexing of pixel intensities that is relative to the shape of the face. The extracted features of a face image that are represented in a vector can vary greatly due to shape deformation as well as changes in illumination conditions, which makes accurate shape prediction using these extracted features difficult but to accurately predict the shape we need reliable features and on other hand we need an accurate estimate of shape to extract reliable features. To solve this problem the image is transformed into a normalized coordinate system based on the current estimate of the image from where the features are extracted to predict an update vector for the shape parameters rather than regressing the shape parameters based on features extracted in the global coordinate system of the image. The above-mentioned process is usually repeated until convergence.
The second key element takes into consideration, how to combat the problem of inference/prediction? At test time an alignment algorithm estimates the shape in the form of a high dimensional vector that best agrees with the image data and our model of shape. The problem has many local optima which is solved by assuming that the estimated shape must lie in a linear subspace which can be discovered. For example, finding the principal components of the training shape. The above stated assumption thus greatly reduces the number of potential shapes considered during inference and helps in avoiding local optima.
Accompanied with the above two key factors is efficient regression learning wherein an appropriate loss function is optimized and feature selection is performed in a data driven manner. Each regressor is learned via gradient boosting with a squared error loss function. A combination of gradient boosting algorithm and prior probability on the distance between pairs of input pixels is used to select the sparse pixel set which is used as the regressor's input. The prior distribution allows the boosting algorithm to efficiently explore a large number of relevant features which results in cascade of regressors than can localize the facial landmarks when initialized with a mean face pose.

C. Blink Detection

The algorithm employed as explained in [6] first estimates the landmark positions,extracts a scalar quantity EAR(eye aspect ratio) which is used to detect whether the eye is open or not and finally an SVM classifier in a short temporal window detects eye bilnks as a pattern of EAR values.

$$EAR = \frac{\|p2-p6\|+\|p3-p5\|}{2\|p1-p4\|} \quad (3)$$

Where $p1,\ldots,p6$ are landmark locations, as depicted as depicted in Figure 6.

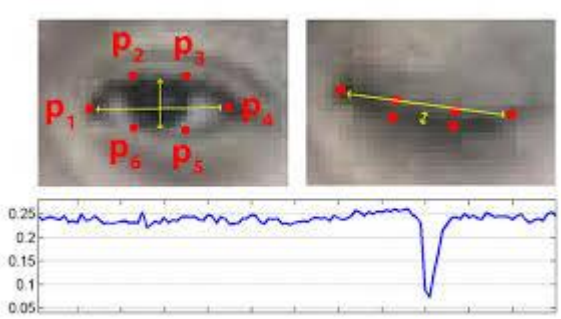


Figure 6: Open and closed eye with landmark detected as p_i along with EAR fluctuation for open and closed eye.

EAR although retains a pretty constant value for all those instances when the eyes are open, but it approaches to zero while closing an eye. Furthermore, it is partially insensitive to head pose orientation and fully in variant to uniform scaling of image and in plane rotation of face. As both the eyes blink concurrently the EAR is averaged.

D. Blink to text Conversion GUI

Electron

The GUI is made using Electron framework. Electron is an open-source framework developed and maintained by GitHub. Electron allows for the development of desktop GUI applications using front and back-end components originally developed for web applications: Node.js runtime for the backend and Chromium for the frontend. The application is written using JavaScript. The GUI application hence created is cross-platform.

Zerorpc

Zerorpc is a light-weight, reliable and language-agnostic library for distributed communication between server-side processes. It builds on top of ZeroMQ and MessagePack. Support for streamed responses - similar to python generators - makes zerorpc more than a typical RPC engine.

IV. WORKING

A. Face Detection

We used HOG detection technique to detect face and generate a bounding box across the face as in Figure 7.

While the use of Viola Jones detection method yields great detection with performance but its application was only limited to frontal faces and was highly variant to illumination fluctuations too. Our application is targeted towards people who are affected from Locked-In-Syndrome and need not have to always be in frontal position, in such cases the HOG method becomes a more viable option as it is invariant to illumination changes and is able to capture geometric features of face that is difficult for linear edge filters such as Haar filters used in Viola-Jones method.

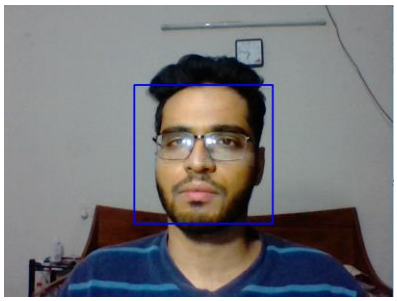


Figure 7: Depiction of face detection by boundary box generation in real time

The HOG method captures gradient structure which is very characteristic of local shape that is done in a local representation with invariant to any kind of transformation: translations and rotations make little difference if they are much smaller than the local spatial or orientation bin size. For human face detection local photometric normalization, orientation sampling and coarse spatial sampling are best and are incorporated in HOG technique because it permits various limbs and body segments to change appearance and move side to side provided they keep a roughly upright position.

B. Facial Feature Detection

With the help of landmark detector trained over iBUG 300 W Dataset[4] inside the dlib library a 68-point (x,y) co-ordinate map is produced to access the various facial features as seen in Figure 8.

- The mouth through points [48, 68].
- The right eyebrow through points [17, 22].
- The left eyebrow through points [22, 27].
- The right eye using [36, 42].
- The left eye with [42, 48].
- The nose using [27, 35].
- And the jaw via [0, 17]

This concept is utilised to map all the facial features in a real time video effectively from which we only extracted the area concerning with eyes.

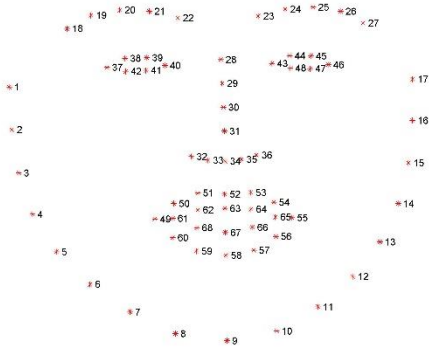


Figure 8: Visualising each facial feature as appoint co-ordinate map



Figure 9: Mapping of facial features

C. *Blink Detection*

Earlier detection techniques were based on motion estimation of the eye region. This motion was estimated from differencing and adaptive thresholding then it was determined whether the eye is closed or not. A major problem for these approaches was that too much importance and dependence were imposed on setup equipment, orientation of head, illumination etc. The eye aspect ratio or EAR is an implementation of facial feature localization using a detector trained in dlib library over [4] data set. This technique was opted by us because of it being invariant to orientation of head and illumination.
As it can be seen from Figure 10 the blink detection technique can easily detect blinks with individuals wearing glasses which better cements our reason for adopting this detection technique.

D. *Blink to text Conversion GUI*

The software has 3 parts that work together as one. The frontend part, that has the UI elements and is written using HTML and CSS. The python backend part, that detects eye blinks and shares that information with the GUI using sockets. This is contained in a python executable file. The JavaScript backend part, that connects the frontend and python backend part and also takes care of the dynamic parts of UI. When the GUI executes, it makes a call to start execution of python executable file. The python executable file setups a TCP server at 127.0.0.1:4242 and also starts the blink detector window. After clicking on “start” from the GUI, the JavaScript backend starts reading the state of blink periodically at 127.0.0.1:4242.

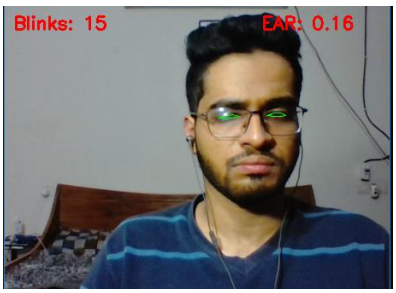


Figure 10 :Detecting blinks

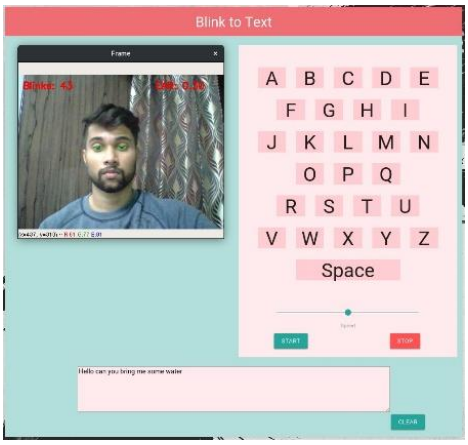


Figure 11: Demonstration of working GUI by demonstrating how to write “Hello can you bring me some water”

Electron allows for the development of desktop GUI applications using front and back-end components originally developed for web applications. This makes the conversion of software to the website easier. Electron applications are cross-platform and hence the user does not require to be using a particular Operating System to use the software. Electron is lightweight and can also be used on raspberry pi.

ACKNOWLEDGMENT

The above work was conducted as a team effort by the aforementioned individuals under the tutelage of Dr. Rinky Dwivedi and support from our families and college.

REFERENCES

[1] Kaiqi Cen, “Study of viola jones real time face detector”, 2016

[2] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Kauai, HI, USA, 2001, pp. I-I.
doi: 10.1109/CVPR.2001.990517

[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 2005, pp. 886-893 vol. 1.
doi: 10.1109/CVPR.2005.177

[4] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, M. Pantic. “300 Faces in-the-Wild Challenge: The first facial landmark localization Challenge.” Proceedings of IEEE Int’l Conf. on Computer Vision (ICCV-W), 300 Faces in-the-Wild Challenge (300-W). Sydney, Australia, December 2013.

[5] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, 2014, pp. 1867-1874.
doi: 10.1109/CVPR.2014.241

[6] T. Soukupová and J. Cech. “Real-Time Eye Blink Detection using Facial Landmarks.” ,2016.