Implementing Deep Learning Model in Human Computer Interaction with Face Recognition using Convolutional Neural Network

Sharath B[1], Harshith R[2], Lankesh J[3], Sanjay G R[4], Chandru A S[5]

[1]Sharath B: Student, Dept. of ISE, NIE-IT, Mysore, Karnataka, India
[2]Harshith R: Student, Dept. of ISE, NIE-IT, Mysore, Karnataka, India
[3]Lankesh J: Student, Dept. of ISE, NIE-IT, Mysore, Karnataka, India
[4]Sanjay G R: Student, Dept. of ISE, NIE-IT, Mysore, Karnataka, India
[5]Chandru A S (co-author): Assistant Professor, Dept. of ISE, NIE-IT, Mysore, Karnataka, India

---***---

Abstract – *With the advent of technology in this digital age, there is always room for improvement in the computer field. Hand free computing is needed from today to meet the needs of quadriplegics. This paper introduces the Human Computer Interaction (HCI) system which is very important for amputators and those who have problems using their hands. The system developed is an eye-based communication that acts as a computer mouse to translate eye movements such as blinking, staring and teasing at the actions of a mouse cursor. The program in question uses a simple webcam and its software requirements are Python (3.6), OpenCv, numpy and a few other packages needed for face recognition. The face detector can be constructed using the HOG (Histogram of oriented Gradients) feature and the line filter, as well as the sliding window path. No hands and no external computer hardware or sensors needed.*

Key Words: Python(3.6), OpenCv, Human computer interaction, numpy, face recognition, Histogram of Oriented Gradients (HOG).

1. INTRODUCTION

PEOPLE who are quadriplegic and unable to talk — for example, with cerebral palsy, traumatic brain injury, or stroke — often have difficulty expressing their desires, thoughts, and needs. They use their limited voluntary movements to communicate with family, friends, and other care providers. Some people may shake their heads. Some may voluntarily blink or blink. Others may move their eyes or tongue. Assisted technical devices have been developed to help them use their movements voluntarily to control computers. People with disabilities can communicate through spelling or add-ons. Disable people who cannot move anything without their eyes. For these people eye movement and blinking are the only way to communicate with an external voice through a computer. This study aims to develop a program that can help people with physical disabilities by allowing them to connect to a computer system using only their eyes. Human interactions with computers have become an integral part of our daily lives. Here, eyes as input, user eye movement can provide a simple, natural and high bandwidth input source.

This Computer mouse or finger movement has become the most common way to move the cursor on the screen in modern technology. The system detects any mouse or finger movement so that the map matches the movement of the cursor. Some people who do not have working arms, so-called 'amputated' will not be able to use the current technology to use the mouse. Therefore, if the movement of the eyeball is not tracked and if the direction of the eye towards it can be determined, the movement of the eyeball can be done on a map and the amputated leg will be able to move the cursor at will. 'Eye tracking mouse' will be very useful for the target person. Currently, mouse tracking is not widely available, and only a few companies have developed this technology and made it available. We aim to fix the eye tracking mouse where most of the mouse functions will be found, so that the user can move the cursor using his or her eye. We try to measure the user's 'viewing' direction and move the cursor to where his or her eye is trying to focus. people who are blind actually cannot use a computer so by adding audio output they can listen to the commands and act accordingly. Mouse pointing and clicking action remains common for a long time. However, for some reason a person may find it uncomfortable or if those who are unable to move their hands, there is a need to use these mouse free hands.

In general, eye movements and facial expressions are the basis of a handless mouse.

2. RELATED WORK

This can be achieved in various ways. Camera mouse suggested by Margrit Betke et. al. [1] for people with quadriplegic and non-speech. User movements are tracked using the camera and this can be mapped to the mouse cursor movement that appears on the screen. However, another approach was proposed by Robert Gabriel Lupu, et. al. [2] with the interaction of a personal computer that uses a device mounted on the head to track eye movements and interpret on-screen. Another option for Prof. Prashant salunke et.al [3] introduced eye tracking techniques using Hough transform.

A lot of work is being done to improve the features of the HCI.A paper by Muhammad Usman Ghani, et. Al [4] suggests that eye movements can be read as input and used to help the user access visual cues without using any other hardware device such as a mouse or keyboard [5]. This can be achieved by using image processing algorithms and computer vision. Another way to get eyes is to, by using the Haar cascade feature.

Eyes can be obtained by pairing them with pre-stored templates as suggested by Vaibhav Nangare et. al [6]. For an accurate picture of the iris an IR sensor can be used. A gyroscope can be used to identify the head as suggested by Anamika Mali et. al [7]. Clicking function can be done by 'viewing' or by staring at the screen. Also, by looking at a portion of any part of the screen (top or bottom), the scrolling function can be done as suggested by Zhang et. al [8].

As well as eye movements, it becomes easier when we combine subtle face movements with their parts as well. Real-time eye detection using facial expressions as suggested by Teresa Soukupova and Jan ´ Cech [9] shows how blinking can be detected using facial expressions. This is a great feature as blinking actions are needed to translate click actions. Visualization of the eyes and other parts of the face can be done using openCv and Python with dlib [10]. Like a blink can be detected. [11]. Akshay Chandra [13] proposes the same by attaching a mouse cursor using facial movements.

3. METHODOLOGY

The proposed procedure in this paper works based on the following actions: a) Blinking b) closing both the eyes c) Head movement (pitch and yawning) d) Opening the mouth.

3.1 Requirements:

The requirements for this job are listed and reviewed and analysed. Since the core of this project is part of the software, we only talked about the need for software.

Software requirement –

1) Python:

   Python is the software we used, where the interface is Jupyter Notebook. It is a very common tool, in some basic mathematics, and is one of the easiest tools to use. Some of the libraries used are:

   - Numpy – 1.13.3
   - OpenCV – 3.2.0
   - PyAutoGUI – 0.9.36
   - Dlib – 19.4.0
   - Imutils – 0.4.6

The procedure is as follows:

1) Since the project is based on finding facial features and drawing a cursor, the webcam needs to be accessed first, which means it will be turned on by the webcam. Once the webcam is turned on, the system needs to extract the entire frame from the video. The frame rate of the video is usually equal to 30 frames per second, so the frame every 1/30 second will be used for processing. This framework contains a set of processes before the elements of the framework are detected and mapped in the cursor. And this process continuously occurs throughout the framework as part of the loop.

2) Once the frame has been removed, face circuits need to be detected. Therefore, the frames will face a set of image processing functions to process the frame properly, making it easier for the system to detect features such as eyes, mouth, nose, etc.

The processing techniques are:

i) Resizing:

The image is rotated first over the y axis of y. Next, the image needs to be resized. Resize function refers to setting a new image adjustment to any value as required. For this project, the new resolution is 640 X 480.

ii) BGR to gray:

The data we use to detect different parts of the face requires a grey format image to give more accurate results. Therefore, the image, i.e., the video frame from the webcam needs to go through the process of converting its format from RGB to grayscale.

Once the image has been converted to grayscale format, it can be used to detect faces and identify facial features.

iii) Detection and Prediction of facial features:

For face recognition and features, a pre-built model is used in the project, with available values that can be translated by python to ensure that the face is located in the image. There is a function called ‘detector ()’, which is made available to models, which helps us to find faces.

After face detection, facial features can now be detected using the ‘predictor’ function. The function helps us to score 68 points on any 2D image. These points are accompanied by different facial points near the required parts such as eyes, mouth, etc.

Acquired activity values are in the form of 2D links. Each of the 68 points is the basic number x and y of the connectors, which, when connected, will form a visible surface.

Then they are saved as a list of values so that they can be edited and used in the next step to connect any links and draw a border to represent the required regions of the face.

The four sets of frames are considered to be 4 separate parts of these numbers kept in order, to be kept separately as connectors to represent the required regions, namely: left eye, right eye, nose and mouth.

Once 4 identical members have been prepared, borders or 'concerts' are drawn near the points using 3 of these same members by connecting these points, using the 'drawcontour' function and the shape formed around the two eyes and mouth.

iv) Mouth and Eye aspect ratios: Once the concerts have been planned, it is necessary to have a state reference, which in comparison, provides the program with any information on any action performed by these regions such as blinking, yawning, etc.

These references are understood as measurements, between 2D connectors, and the switch changes, in fact telling us that, parts of the face region have moved away from the normal area and action has been taken.

The system is built to predict local facial features. The pre-built Dlib model assists in fast and accurate facial recognition and 68 points 2D facial features as already mentioned. Here, the Eye-Aspect Ratio (EAR) and the mouth-aspect-ratio (MAR) are used to detect blinking / yawning respectively. These actions are translated into mouse actions.

Eye-Aspect-Ratio (EAR)

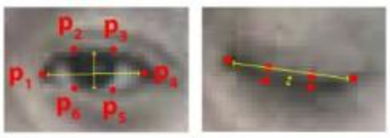

Fig.1: Eye-Aspect-Ratio

$$\text{EAR}=\frac{||P2-P6||+||P3-P5||}{||P1-P4||}$$

The value of EAR decreases significantly when the eye is closed. This feature can be used to click on an action.

Mouth-Aspect-Ratio (MAR)

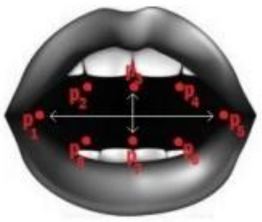

Fig.2: Mouth-Aspect-Ratio

$$\text{MAR}=\frac{||P2-P8||+||P3-P7||+||P4-P6||}{||P1-P5||}$$

MAR rises when the mouth opens. This is used as a starting and ending action on the mouse. For example, if the scale is increased, it could mean that the distances between the points representing the facial region have changed and the action has been man-made. This action should be understood as a person trying to perform a task using a mouse.

Therefore, in order for these functions to be enabled, an 'aspect ratios' needs to be defined, which, if it exceeds the specified limit, translates the action to be performed.

v) Detection of actions performed by the face: After defining the dimensions, the framework can now compare the dimensions of the facial features with the dimensions described in the different actions, of the current process being considered. It is done using the statement 'if'. The actions identified by the program are:

1) To open the mouse: The user needs to 'yaw' which opens his mouth, vertically, and then increases the distance between the corresponding 2D mouth points. The algorithm detects changes in range by making a computer rating, and if this rating exceeds a certain limit, the system is activated and the cursor can be removed. The user needs to position the nose, up, down, left or right rectangle, in order to move the cursor in the opposite direction. The farther away the rectangle, the faster the movement of the cursor

2) Left / Right Click: With a click, you need to close any of the eyes, and make sure you keep one open. The system first checks whether the magnitude of the difference is greater than the limit by using the difference between the two eye measurements, to ensure that the user wants to make a left or right click, and does not want to scroll. (Requires both eyes to clear)

3) Scroll: The user can scroll the mouse, up or down. You need to close your eyes in such a way that the dimension of both eyes is below the set value. In this case, when the user puts his nose out of the rectangle, the mouse does the scrolling work, rather than moving the cursor. He can move his nose or over the rectangle to scroll up, or move it below the rectangle to scroll down.

vi) Multiple Points: Although the current system does not track multiple points, some initial tests were performed to test such tracking. In the future, multi-point tracking may be used, for example, to calculate distances between the nostrils and the pupils for eye recognition. This will result in a special tracking system. The power of the current version is its standard — any feature can be selected for tracking

vii) Voice output: In this project we have added a voice output to people who cannot see the screen using their eyes, the voice output is: right click, left click, right, left, up, down, scroll mode is on, scroll mode is off, input mode is on and input mode is off.

viii) Distance from screen to user: This distance needs to be adjusted correctly so that you can accurately distinguish different areas of the eye and keep track of the center. If this distance is too large then the movement of the eye will be much smaller than the width of the screen so it will be difficult to accurately track the center area. If the distance is too small than the movement of the eyes it will be too large and due to the curvature of the eye.

4. RESULT

The mouse control system will work, and the user can move the cursor or click at will. The value of the cursor position change on any axis can be adjusted according to user requirements. Mouse control is activated by opening the mouth when the amount of MAR crosses a certain limit.

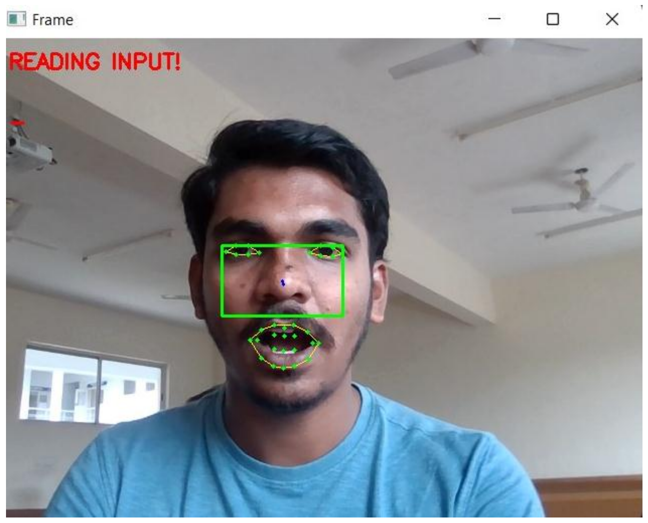

Fig.3: Opening the mouth to open 'input mode'

The cursor is moved by moving the eye right, left, top and down as per the requirement.

Scroll mode is activated by mock operation. Scrolling can be done by moving the head upwards called throwing and, on the sides, called yawning. Scroll mode is stopped by joking again. Clicking action occurs with the blink of an eye. Right blink is accompanied by right blink and left blink is accompanied by left blink.

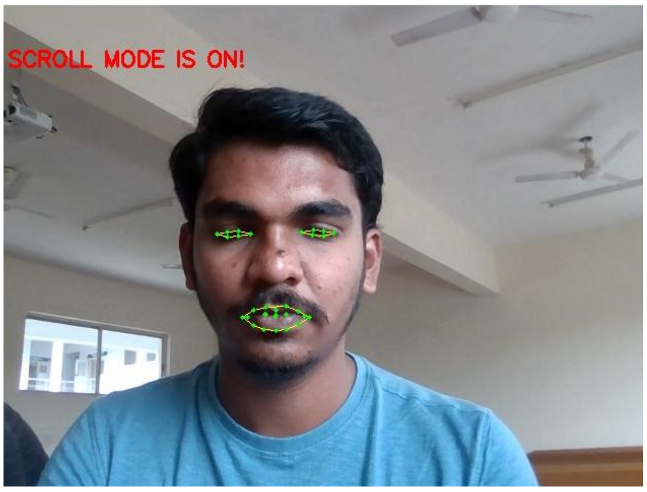

Fig.4: Close your eyes for a few seconds to go to scroll mode

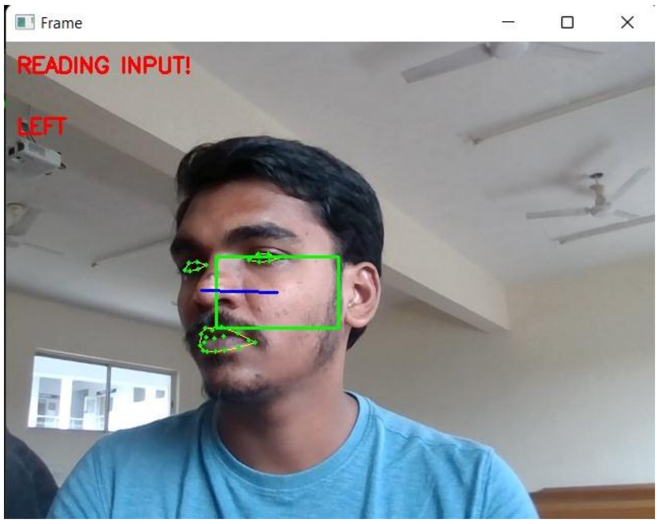

Fig.5: moving face left to move cursor left

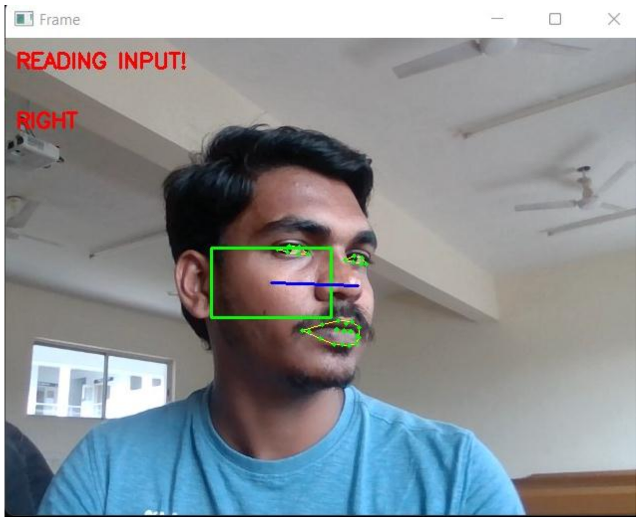

Fig.6: moving face right to move cursor right

Mouse sensitivity can be adjusted according to user needs. Overall, the project works as it should. although comfort is not the same as a hand-controlled mouse, this project can be easily implemented with some practice. We have implemented a testing process to test the usability, accuracy and reliability of the ETM system. Twenty participants, ranging in age from 19 to 27, participated in the study. They all had normal or adjusted vision, had no prior eye tracking experience and needed a little training to use the system.

5. CONCLUSIONS

This function can be extended to improve system speed using better trained models. Also, the system can be made more powerful by making a change in the location of the cursor, in proportion to the number of movements of the user's head, that is, the user can determine, the amount he wants the position of the indicator to change. we have designed the project in such a way that blind people can use a computer by listening to audio output produced by their camera-directed actions. Also, future research work can be done to make the measurement more accurate, as the range of values is the result of the measurement of the feature, which is usually smaller. Therefore, to make the algorithm detect actions more accurately, there may be some modification to the formulas of the aspect ratios used. Also, to make the face detection process easier, some image processing techniques can be used before the model gets the face and facial features.

REFERENCES

[1] Margrit Betke,James Gips, “The Camera Mouse: Visual Tracking of Body Features to Provide Computer Access for People With Severe Disabilities” ,in IEEE transactions on neural systems and rehabilitation engineering, vol.10, no.1, March 2002

[2] Robert Gabriel Lupu, Florina Ungureanu, Valentin Siriteanu, “Eye Tracking Mouse for Human Computer Interaction”, in The 4th IEEE International Conference on E-Health and Bioengineering - EHB 2013.

[3] Prof. Prashant Salunkhe, Miss. Ashwini R. Patil , “A Device Controlled Using Eye Movement”, in International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) - 2016.

[4] Muhammad Usman Ghani, Sarah Chaudhry, Maryam Sohail, Muhammad Nafees Geelani, “GazePointer: A Real Time Mouse Pointer Control Implementation Based On Eye Gaze Tracking”,in INMIC 19-20 Dec. 2013.

[5] Alex Poole and Linden J. Ball, “Eye Tracking in Human-Computer Interaction and Usability Research: Current Status and Future Prospects,” in Encyclopedia of Human Computer Interaction (30 December 2005) Key: citeulike:3431568, 2006, pp. 211-219.

[6] Vaibhav Nangare, Utkarsha Samant, Sonit Rabha, “Controlling Mouse Motions Using Eye Blinks, Head Movements and Voice Recognition”,in International Journal of Scientific and Research Publications, Volume 6, Issue 3, March 2016.

[7] Anamika Mali, Ritisha Chavan, Priyanka Dhanawade, “Optimal System for Manipulating Mouse Pointer through Eyes”,in International Research Journal of Engineering and Technology (IRJET) March 2016.

[8] Xuebai Zhang, Xiaolong Liu, Shyan-Ming Yuan,Shu-Fan Lin, “Eye Tracking Based Control System for Natural Human-Computer Interaction”,in Hindawi Computational Intelligence and Neuroscience Volume 2017, Article ID 5739301, 9 pages.

[9] Tereza Soukupova´ and Jan Cˇ ech. “Real-Time Eye Blink Detection using Facial Landmarks.” In 21st Computer Vision Winter Workshop, February 2016.

[10] Adrian Rosebrock. “Detect eyes, nose, lips, and jaw” with dlib, OpenCV, and Python.”

[11] Adrian Rosebrock. Eye blink detection with OpenCV, Python, and dlib.

[12] 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