2022 3rd Int. Conf. on Innovations in Science, Engineering and Technology (ICISET)
26-27 February 2022, Chittagong, Bangladesh

A Robust Webcam-based Eye Gaze Estimation System for Human-Computer Interaction

Koushik Roy[1] and Dibaloke Chanda[2]
Department of Electrical, Electronic and Communication Engineering (EECE)
Military Institute of Science & Technology (MIST), Dhaka - 1216, Bangladesh
rkoushikroy2@gmail.com[1]; dibaloke66@gmail.com[2]

***Abstract*—Precise eye gaze detection has a multitude of real-life use cases such as the input mechanism for physically disabled persons, driver's attention detection in vehicles, cheating detection in online exam, augmented reality, medical research and so on. Most of the applications need to support real-time functionality, thus the need for a fast and reliable method for eye gaze detection can be justified. In this research work, we propose a non-wearable and webcam-based eye-gaze detection method that offers multiple benefits in terms of accuracy, robustness, and reliability over existing solutions. We approached gaze detection as a multiclass classification problem, this reduced the complexity of the method and allows the solution to be implemented in a way that allows free head movement of the user. We leveraged the latest innovation and breakthroughs in deep learning to construct a novel eye-gaze detection method that works using the live video feed from any modern webcam with acceptable frame rates for proper real-time applications. We achieved 99% validation accuracy in gaze prediction and 20 FPS on average in real-time applications such as mouse pointer control and scrolling.**

***Index Terms*—Eye Gaze Detection, Mediapipe, Convolutional Neural Network, Human-Computer Interaction**

I. Introduction

Eye-tracking is a method of determining where a person is looking (point of gaze), what he or she is looking at, and how their attention moves and stops in space by watching their eye movements. Eye-tracking software allows a computer to scan and record a person's gaze movements on a screen in real-time and with great accuracy. It is a natural and simple interface between a human and an external technology that offers people with physical limitations a powerful way of communication [1]. Patients with diverse types of chronic and progressive neurological disorders that impair their ability to move can use eye-tracking technology instead of traditional input devices [2].

Other than patient communication, eye-tracking technology has a wide range of uses. Because gaze motion reflects cognitive processes, it is used in medical and psychological research as a tool for recording and researching human visual activity [3]. Monitoring eye attention can be useful in product design and marketing where users can implicitly point out the salient features in a product [4]. The resolution and frames per second(FPS) of video games as well as augmented reality applications can be improved by capitalizing on the eye-tracking technology [5].

Even though different iterations of eye-tracking devices have existed for quite some time, the usage has been limited to laboratory usage instead of using it for Human-Computer Interaction(HCI), because of the huge cost associated with these devices [6]. Fast forward to recent years, the technological advancements bring us better and cheaper components for eye tracking, thus making it much more likely to integrate with everyday usage. Replacing traditional input mechanisms such as a mouse, keyboard, or joystick with an eye-tracking device comes with its own set of strengths and weaknesses. For instance, the eye is considered a crucial real-time input medium, which is responsible for obtaining 80-90% of outside world information [7]. Thus, HCI using the eye can be vital for people with motor disabilities to become more independent. On the other hand, the weaknesses include the drop in performance for different eye shapes, noise in input images, low image resolution, head pose and distance from the camera, special wearable equipment, etc. Recent research works including ours have been trying to eliminate these shortcomings and make eye-tracking-based HCI accessible to everyone with little to no additional costs.

In this research work, we aim to make eye tracking based HCI technology more affordable, intuitive, and accessible to more people. The proposed method utilizes deep learning to facilitate an accurate and robust eye gaze estimation technique to provide a high bandwidth source of additional user input. Our approach doesn't require expensive eye-tracking devices to work instead it only requires a low-cost webcam. We used state-of-the-art face mesh technology(Mediapipe) for face detection and eye localization thus making our approach suitable for mobile applications with high FPS. Furthermore, the gaze estimation deep learning model is lightweight and provides useable FPS (on average 20) in moderate hardware. Even with shifting lighting conditions and random background, the system's performance is adequate. It can even identify the eye gaze of a person who is wearing spectacles. Our approach prioritizes the user's convenience by providing accurate eye gaze estimation and eye blinks detection with unconstrained head movements and background. We used the custom normalization method to compensate for the user's distance from the camera, thus making the approach much more intuitive and flexible in an unpredictable situation. We applied our model for mouse movement control, mouse click, and scrolling assistance.

A. Existing Literature

In 2013, Ghani et al. [8] implemented an eye gaze tracking method for real-time mouse pointer control applications using input image frames from a webcam. They used a feature extraction-based approach to facilitate eye and pupil detection.

Deepika et al. [9] in 2015, presented an eye movement-based HCI system by converting the input RGB image into a gray image and applying thresholding techniques as well as Hough Transformation to locate the center point of the iris. Shimata et al. [10] proposed a technique for HCI using an infrared camera placed very close to the eye. They used image processing techniques such as thresholding along with intensity frequency distribution for accurate localization of pupil. Mange et al. [11] used very inexpensive IR sensors and IR-led based apparatus connected to a processing unit to create an eye gaze and blinking based HCI system. Banaeeyan et al. [12] presented a non-intrusive eye gaze tracking method using a single eye image for HCI usages. The techniques they used employed user-specific geometric and trigonometric relationships with respect to a user's point of regard for final mapping onto a user interface.

Fatima et al. [13] in 2016, proposed some new applications of eye gaze such as effortless scrolling and enriched reading,that are aimed at general users. They investigated various eye gaze estimation techniques and algorithms. In 2017, Hegde et al. [14] developed an eye gaze determination method using Gaussian filtering, Haar detection, and equalization of highlighted edges aimed at aiding amputees and paralyzed persons. Bozomitu et al. [15] experimented with ways to improve control in an eye-tracking based human-computer interface. Pasarica et al. [16] presented a filtering method to extract the pupil location from the eye image and used this to create an HCI interface. Zhang et al. [7] presented a control method for mouse, keyboard using industrially marketed eye-tracking devices such as Tobii EyeX and Eye Tribe.

In 2018, Zhang et al. [17] proposes a rapid eye-tracking method, to respond to a situation that requires a high processing speed but less accuracy. They used a webcam with a low resolution of 640 × 480, which decreased the cost of devices considerably. They employed a low-resolution webcam. To compensate for the error introduced by the usage of low-resolution images, they used a modifier color intensity-based approach which offers low computational complexity.

In 2019, Phothisonothai et al. [18] used wearable EEG Headset and non-wearable Tobii Eye Tracker 4C and proposed a user-friendly virtual keyboard typing in the Thai language. Scalera et al. [19] in 2021, present a novel architecture for controlling an industrial robot via an eye-tracking interface for artistic purposes.

B. Limitations of Existing Literature and Motivation

Even though the existing literature uses different methods for eye gaze estimation with high accuracy, the majority of them are not suitable for real time application as real time application requires high FPS. Moreover, recent applications of eye gaze estimation are done on mobile devices. This is one of the major limiting factor for the existing methods as they will achieve high accuracy but at the cost of high computational complexity.

Even if the computational complexity is reduced by using optimization in the software, these methods still lack robustness, meaning they will work perfectly for specific orientations of the head, but will perform badly for other orientations.

Another aspect to consider is how convenient it is for the user to use the system i.e. user-friendliness. To get a precise result and high enough accuracy many of the existing methods require wearable devices. Its obvious that such devices will constrain the user's movement. Moreover, these wearable devices are costly and exclusive to different industries, which makes it more harder for ordinary users to obtain them.

On top of these drawbacks, another major drawback of the existing systems is how these methods handle the gaze estimation process. In most of the cases, the system provides a continuous gaze output which adds complexity and jittery movements. It is obvious that a discrete approach where the system would detect only some specific classes of gaze will reduce the computational complexity by a huge amount and provide a stable output.

C. Major Contributions

- HCI system is capable of localizing eye in real time with high FPS
- Webcam-based which is cost effective and convenient for the user
- Prediction is class-based and discrete, which makes it computationally inexpensive
- Highly Robust as Mediapipe is able to localize eye from webcam input despite pose variation or illumination variation
- HCI system is capable of running on mobile devices

D. Paper Organization

The rest of the study is arranged as follows: In Section II, the design and detailed process of the system's functions will be outlined. Section III presents the suggested system's experimental study and assessments. Section IV describes the application of eye gaze estimation in detail. Section V presents the future research scope. Finally, Section VI concludes the article.

II. METHODOLOGY

As seen in the system block diagram in Fig. 1 the implementation consists of several stages. At first, we made our own initial dataset consisting of images with full faces, captured using a webcam in HD resolution. The images were taken with random backgrounds, varying lighting conditions, and distance from the webcam. After that, we used state-of-the-art face mesh method to localize the face area and detect 468 landmark points [20]. We used these detected landmark points to extract the eye portion of the image and created our own dataset using a custom lightweight convolutional neural net(CNN). The dataset was divided into 9 discrete classes(such as Up, Down, Left, Right, and so on). Before training, the dataset was processed and normalized. After our dataset collection and preprocessing stage were done, we trained and tuned the neural net models for eye gaze estimation. Then we applied our gaze estimation model in HCI applications(Mouse Pointer Control, On-screen keyboard, and Scroll Assistance) using the necessary packages of Python Programming language in the Windows operating system. Our training and application method is easily reproducible and an initialization step can be aggregated to make this approach work for people with different ethnicity, races, and age.

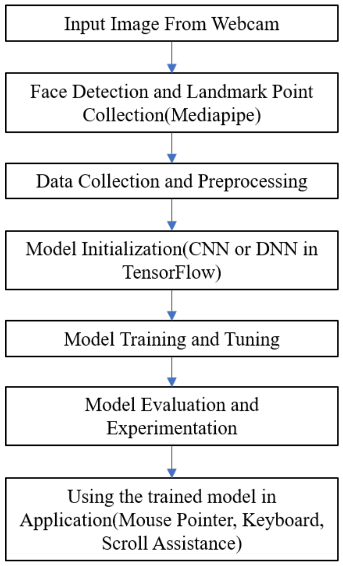

Input Image From Webcam
Face Detection and Landmark Point Collection(Mediapipe)
Data Collection and Preprocessing
Model Initialization(CNN or DNN in TensorFlow)
Model Training and Tuning
Model Evaluation and Experimentation
Using the trained model in Application(Mouse Pointer, Keyboard, Scroll Assistance)

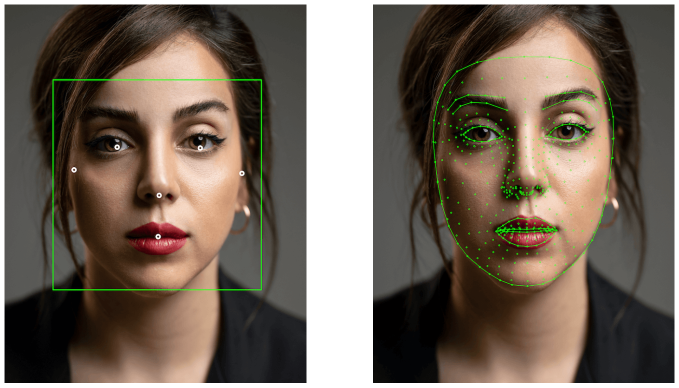
Fig. 1: Overall System Block Diagram

A. *Face Detection and Landmark Point Collection*

Face Mesh by MediaPipe [20] is a real-time facial geometry solution that calculates 478 3D face landmarks on mobile devices. It uses machine learning (ML) to infer 3D surface geometry from single-camera input, eliminating the requirement for a separate depth sensor. The approach achieves real-time performance for live experience by combining lightweight model design with GPU acceleration across the pipeline. Fig. 2 shows an example of face bounding box detection and facial landmark point detection using OpenCV-python and MediaPipe packages in Python 3.9, where the detection confidence was 91%. These landmark points will be used in future steps of data collection and data preprocessing. Having a reliable and robust face landmark detector is crucial for eye gaze estimation as all HCI applications based on the eye gaze output will be in real-time.

B. *CNN Based Gaze Estimation*

In the Convolutional Neural Net(CNN) based gaze estimation method, we used our custom dataset of cropped eye images with discrete class output(9 classes) as input to the neural net model. At first, we had to collect the cropped eye images from our baseline dataset that contains face images.



(a) Face bounding box(in green) with 6 key point landmarks
(b) 468 landmark points detected from the localized face image

Fig. 2: Facial Landmark points detection using MediaPipe [20]

After that, we made a custom method for train-test split as the cropped eye images in our dataset remain in their respective class directory. Then we used Keras and TensorFlow for data preprocessing, which includes image rescaling and augmentation. When all the previous steps are done, we focus on model making, training, evaluating, and optimizing. Finally, we use our trained model in the real-time application and use the gaze estimation data as an input mechanism for HCI applications.

1) Data Collection: The first stage in the CNN-based gaze estimation is data collection. Our baseline dataset contains 9000 face images of one person, belonging to 9 classes of eye gaze(Down: 0, Left: 1, Lower Left: 2, Lower Right: 3, Middle: 4, Right: 5, Up: 6, Upper Left: 7, Upper Right: 8). Each class contains exactly a thousand image frames. To make the input dataset for the CNN model we need to extract the eye part from the face images.

To get a robust and reliable method for eye area localization, we used MediaPipe face landmarks. The steps to acquire cropped images are given below:

1) Getting landmark points in the face image frames.
2) Isolating the necessary landmark points to localize the eye area.
3) Transforming the landmark points for the left eye to obtain coordinates necessary for the cropping method in OpenCV, which is presented in Fig. 3.
4) Cropping and saving images in their respective directory.

Step 1 uses MediaPipe facemesh API to obtain the facial landmark points from the input images. In step 2, we use two landmark points for each eye that are situated near the two corners of the eye. The landmark points are 55, 31 for the left eye, and 285, 261 for the right eye. We experimented with different landmark points and found out that using these specific points gave us a more uniform result. In step 3, a slight modification for the left eye landmark position was required as the OpenCV cropping function takes the upper left and lower right corner locations of a rectangle shape to work properly. The landmark position of the left eye we get from the

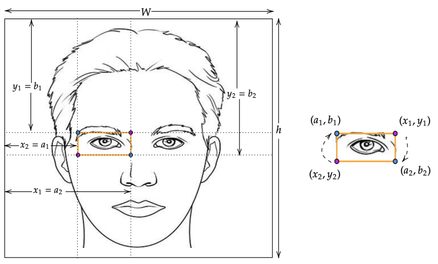


Fig. 3: Localizing eye position based on predefined landmark coordinates

previous step gives us the location of the opposite positions of the required corner points. Therefore, we formulated a simple 2d coordinate transformation equation to obtain the required corners' coordinates. And finally, in step 4, we cropped the images and saved them in their respective directories. After all steps are done, we are left with 18000 eye images belonging to 9 eye gaze classes.

2) *Data Preprocessing:* We used image data generator from Keras preprocessing, which is integrated with TensorFlow Core v2.5.0 using Python. Image data generator takes the original input images and augment them on the fly, thus eliminating the need to save the images in storage and consequently make the training process faster. Newer version of Tensorflow supports augmentation using the Graphics Processing Unit (GPU), which can be vital in saving the overall training time. The following augmentation parameters were used: Rescale, Rotation, Width Shift, Height Shift, Sheer and Zoom.

3) *Model Training:* We started with a simple and lightweight Keras sequential model, which is constructed with multiple convolutional and max-pooling layer arrays. The model was terminated with a flattened layer followed by a dropout layer and the output dense layer. We experimented with the model size and different hyperparameters and decided to go with the final version of the model which is presented in Fig. 4. In our experimentation with the model, we found this version as a sweet spot that provides both the robustness to be applied in a real-time scenario as well as good accuracy to work as a sophisticated input mechanism for HCI application.

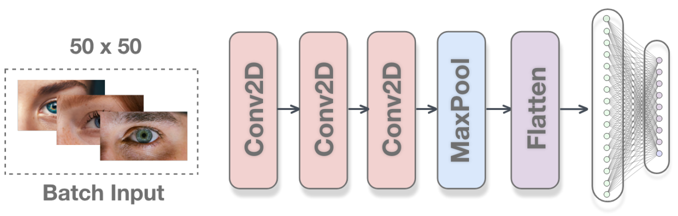


Fig. 4: Custom CNN architecture with an input image

Our sequential CNN model uses the conv2D layer provided by Keras that uses Rectified Linear Unit(RELU) as the activation function. And the 512 unit dense layer, which resides before the final layer, also uses a RELU as activation. But the final dense layer uses SoftMax as activation, which is generally common practice for categorical class outputs.

Hyperparameters used for training the model are presented in Table I. For model compilation, categorical cross-entropy loss was given as a parameter, the formula for which is presented in Equation 1.

TABLE I: Training Parameters for CNN based gaze estimation method

Parameter	Value
Base Architecture	Convolutional Neural Network(CNN) based image classifier
Classes	9
Batch Size	100
Number of Epochs Trained	25
Input Size	(50, 50, 3)
Model Type	Multiclass Classification
Hidden Layer Actication	Rectified Linear Unit (ReLU)
Output Layer Activation	Softmax
Optimizer	RMS prop
Learning Rate	0.001
Loss	Categorical Crossentropy
Model Training Time	9 Minutes

The optimizer used in this case was Root Mean Square Propagation (RMS prop) which is a gradient-based optimization technique. We also added the precision, recall, and accuracy metrics for further evaluation of the model performance. The model was trained for 25 epochs, steps per epoch were 144 (training set/batch size), and the validation steps were 36. The final dense layer employs softmax activation function, the representation of which is given in Fig. 5.

$$Loss = \sum_{outputsize} -t_i log(y_i) \tag{1}$$

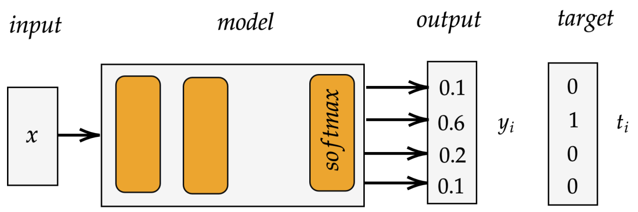


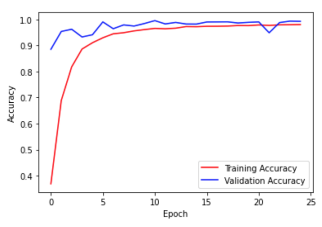
Fig. 5: Diagram representing softmax activation function

III. Experimental Results

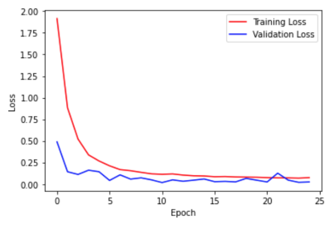
We recorded all our model's training progressing with the necessary evaluation matrices. The model was trained for 25 epochs in total and the execution took about 9 minutes. We can see the categorical cross-entropy loss value started at 1.91 and went to as low as 0.07 in the 25th epoch for the training set. In the case of the validation set, the loss value reaches 0.02 in the last epoch, which is very close to the training loss, thus shows that the chance of overfitting is very minimal. In terms of accuracy, the training accuracy started at 36% and goes up to 97%, and in the case of validation, it goes up to 99%. The observation that the validation results seem to be better than the training set can seem a little counter-intuitive but it may be due to the fact the number of images in the validation set is fairly low compared to the training set. The model progression result in every 5 epochs is presented in Table II.

TABLE II: Training progression and evaluation metrics in every 5 epoch

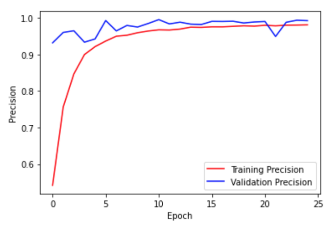
No. of Epoch	Time(s)	Loss	Precision	Recall	Accuracy
1/25	52	1.9152	0.5417	0.1469	0.3693
5/25	19	0.2674	0.9211	0.9015	0.9101
10/25	18	0.1186	0.9638	0.959	0.9606
15/25	18	0.0936	0.9738	0.9704	0.9716
20/25	18	0.0791	0.9772	0.9745	0.9758
25/25	18	0.0745	0.9808	0.9785	0.9799



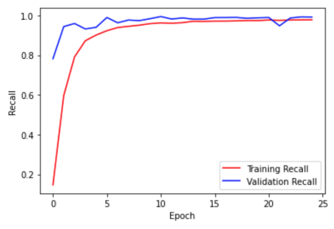
(a) Accuracy vs Epoch



(b) Loss vs Epoch



(c) Precision vs Epoch



(d) Recall vs Epoch

Fig. 6: Training and Validation Evaluation Curves

From Fig. 6, it is apparent that all the metrics increased with increase in number of epoches and with a satisfactory rate. The smoothness of the training curve indicates that the the model did not face any problem during training phase. But with validation dataset, the fluctuation in the curve indicates the model did face some difficulty during the validation phase. Also, the higher value for the validation dataset compared to the training dataset signifies the model did not overfit and able to generalise.

In Fig. 7, the confusion matrix summarizes the prediction capability of the model for 9 different classes which is generated for validation dataset. The model is able to predict all the 9 classes with very high precision and there is very minimal chance of false detection.

IV. Application of Eye Gaze Estimation

A. Mouse Pointer Control

The eye gaze prediction model was used to track the orientation of the eye in real time. With the help of "mouse"

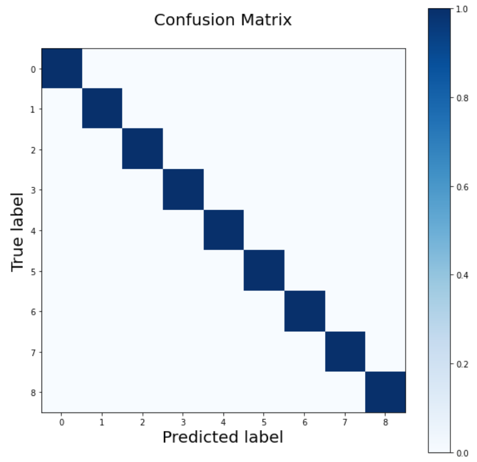


Fig. 7: Diagram representing Confusion Matrix of the Model for Validation Dataset

library, the mouse pointer can be moved across the screen in a specified direction with a constant speed. For instance, if predicted class output was "upper left" the mouse pointer would move in that direction at a constant speed. We selected the "Middle" class as trigger to stop the movement of the mouse pointer. As there are 9 classes, it provides the user with a lot of flexibility with the direction of movement. The speed of the mouse pointer movement is correlated with preciseness of the control. If the speed is set to a large value, the user would be able to move the mouse pointer quickly but will have less precise movement. On the other hand, with a small speed value, the pointer would move with high precision but it would take a considerable amount of time to move the pointer from one end of the screen to another. After some trial and error we found an optimized speed. The most significant contribution is this system works in real time in a relatively low spec system with high accuracy.

B. Mouse Click Control

Introducing click control was easy as it requires only detection of eye blinks. Left eye blink was used to trigger the left mouse click and right eye blink was used to trigger the right mouse click.

The eye blink is detected by calculating the distance between two landmark points, one from on top of the eye, another from bottom of the eye. The system can successfully distinguish between open eye state and close eye state, as it only need to keep track of the distance between two specific landmarks at a point in time.

C. Scroll Assist

Implementing the scroll assist requires some kind of trigger that would change the mode of control. Initially, we tried to use two eye blink as the trigger, but that was being misinterpreted by the system as a mouse click control. Moreover, multiple false triggers can happen, as a normal person can blink up to 15-20 times per minute.

To avoid these false triggers, a close eye state for 3 seconds was used as a trigger for going into scroll mode. After going into the scroll mode, the user can scroll the mouse at a constant speed. The speed vs precision trade-off mentioned in the mouse pointer section is also applicable in scroll mode.

To implement scrolling, 6 class output was mapped to 2 class output. Upper left, up, upper right was mapped to Up and lower left, lower, lower right was mapped to the lower class. Therefore, after going into the scrolling mode, the user can look upwards in any direction to scroll up and look downwards in any direction to scroll down. Keeping the eye gaze in the middle, left or right direction would stop the scrolling.

V. Future Research Scope

Each year a lot of accidents happen, due to the lack of attention from drivers. Eye gaze detection system keeps track of if the user is looking at a particular direction. Warning system can be developed which will warn the driver if it detects the driver is not paying attention for more than 30s.

This system can be extended to online exam cheat detection. Some gaze classes can be predefined as valid class. If it is a computer-based exam, the only valid class should be left, middle, right.

A webcam can track the eye gaze of the user in real time and find out which part of the screen the user is focusing on in that time instant. Then, the resolution of that particular area can be increased or decreased depending on what they are looking at.

VI. Conclusion

Eye gaze estimation is an intuitive and convenient input method which has many applications in real life and academic research. In this research work, a robust eye gaze estimation system which has been implemented that has several benefits over the existing solutions. The proposed system achieves 99% validation accuracy using dataset created with low cost web cam, whereas many of the existing solutions require a high cost wearable device to achieve such high accuracy. Moreover, the prediction is in real time for the gaze class output which provides smooth gaze output. The multiclass prediction approach reduced the computational complexity of the system and enabled the system to work in real-time.

A HCI system that provides functionality like mouse control, mouse click, and scrolling has been successfully implemented. With 9 class output, the user has a lot of flexibility and degrees of freedom to work with the HCI system. The system can easily distinguish between an open eye state and a close eye state, which can be used as effective triggers for mouse clicks or toggling between different modes of operations. This system was tested on relatively low spec hardware, which indicates it can be easily be commercialized without requiring any additional resources.

References

[1] P. Majaranta and A. Bulling, "Eye tracking and eye-based human–computer interaction," in *Advances in physiological computing*. Springer, 2014, pp. 39–65.

[2] E. Pasqualotto, T. Matuz, S. Federici, C. A. Ruf, M. Bartl, M. Olivetti Belardinelli, N. Birbaumer, and S. Halder, "Usability and workload of access technology for people with severe motor impairment: a comparison of brain-computer interfacing and eye tracking," *Neurorehabilitation and neural repair*, vol. 29, no. 10, pp. 950–957, 2015.

[3] J. L. Orquin and K. Holmqvist, "Threats to the validity of eye-movement research in psychology," *Behavior research methods*, vol. 50, no. 4, pp. 1645–1656, 2018.

[4] R. d. O. J. dos Santos, J. H. C. de Oliveira, J. B. Rocha, and J. d. M. E. Giraldi, "Eye tracking in neuromarketing: a research agenda for marketing studies," *International journal of psychological studies*, vol. 7, no. 1, p. 32, 2015.

[5] P. M. Corcoran, F. Nanu, S. Petrescu, and P. Bigioi, "Real-time eye gaze tracking for gaming design and consumer electronics systems," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 2, pp. 347–355, 2012.

[6] R. J. Jacob and K. S. Karn, "Eye tracking in human-computer interaction and usability research: Ready to deliver the promises," in *The mind's eye*. Elsevier, 2003, pp. 573–605.

[7] X. Zhang, X. Liu, S.-M. Yuan, and S.-F. Lin, "Eye tracking based control system for natural human-computer interaction," *Computational intelligence and neuroscience*, vol. 2017, 2017.

[8] M. U. Ghani, S. Chaudhry, M. Sohail, and M. N. Geelani, "Gazepointer: A real time mouse pointer control implementation based on eye gaze tracking," in *INMIC*. IEEE, 2013, pp. 154–159.

[9] S. Deepika and G. Murugesan, "A novel approach for human computer interface based on eye movements for disabled people," in *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. IEEE, 2015, pp. 1–3.

[10] R. Shimata, Y. Mitani, and T. Ochiai, "A study of pupil detection and tracking by image processing techniques for a human eye-computer interaction system," in *2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. IEEE, 2015, pp. 1–4.

[11] A. A. Mange, A. V. Choudhari, and S. Prasad, "Gaze and blinking base human machine interaction system," in *2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*. IEEE, 2015, pp. 1–4.

[12] R. Banaeeyan, A. A. Halin, and M. Bahari, "Nonintrusive eye gaze tracking using a single eye image," in *2015 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*. IEEE, 2015, pp. 139–144.

[13] R. Fatima, A. Usmani, and Z. Zaheer, "Eye movement based human computer interaction," in *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*. IEEE, 2016, pp. 489–494.

[14] V. N. Hegde, R. S. Ullagaddimath, and S. Kumuda, "Low cost eye based human computer interface system (eye controlled mouse)," in *2016 IEEE Annual India Conference (INDICON)*. IEEE, 2016, pp. 1–6.

[15] R. Bozomitu, A. Păsărică, V. Cehan, C. Rotariu, and H. Costin, "Methods of control improvement in an eye tracking based human-computer interface," in *2017 IEEE 23rd International Symposium for Design and Technology in Electronic Packaging (SIITME)*. IEEE, 2017, pp. 300–303.

[16] A. Pasarica, R. G. Bozomitu, H. Costin, C. Miron, and C. Rotariu, "Human-computer interface based on eye tracking with dwell time selection," in *2017 IEEE 23rd International Symposium for Design and Technology in Electronic Packaging (SIITME)*. IEEE, 2017, pp. 375–378.

[17] C. Zheng and T. Usagawa, "A rapid webcam-based eye tracking method for human computer interaction," in *2018 International Conference on Control, Automation and Information Sciences (ICCAIS)*. IEEE, 2018, pp. 133–136.

[18] M. Phothisonothai and S. Tantisatirapong, "Integrated human-machine interaction system: Erp-ssvep and eye tracking based technologies," in *2019 11th International Conference on Knowledge and Smart Technology (KST)*. IEEE, 2019, pp. 244–248.

[19] L. Scalera, S. Seriani, P. Gallina, M. Lentini, and A. Gasparetto, "Human–robot interaction through eye tracking for artistic drawing," *Robotics*, vol. 10, no. 2, p. 54, 2021.

[20] I. Grishchenko, A. Ablavatski, Y. Kartynnik, K. Raveendran, and M. Grundmann, "Attention mesh: High-fidelity face mesh prediction in real-time," *arXiv preprint arXiv:2006.10962*, 2020.