VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
STUDY PROGRAM: INFORMATICS

EYE - MOUSE AS A COMPUTER MOUSE USING MEDIAPIPE FRAMEWORK

(Akimis valdoma kompiuterio pelė naudojant Mediapipe sistemą)

Master‘s thesis

By: Bhumika Shiradanahalli Dhananjaya
VU email:bhumika.shiradanahalli@mif.stud.vu.lt

Supervisor: prof. Dr. Aistis Raudys
Reviewer: prof. Dr. Olga Kurasova

Vilnius
2023

Summary

This master's thesis proposes an approach for Patients having neuro-motor disabilities who may occasionally lose their capacity to communicate with others as their condition deteriorates. This thesis presents a solution that makes it easier for these individuals to interact with others based on eye movements, supported by Google MediaPipe. Google MediaPipe has not been used so far by researchers for mouse interaction.

Python was used for the coding, and Google Mediapipe, a framework for building machine learning pipelines for processing time series data. This cross-platform framework works on desktops, androids. This project tracked the landmarks of the eye. After getting the landmarks, it is used to move the cursor of the mouse using the Pyautogui module. Finally, the survey is conducted to determine the proposed system's effectiveness. Furthermore, the test users tested the system in the absence of spectacles, and this system can detect the iris and pupil better than existing methods. The solution was compared with the existing solutions for mouse control. GPU/CPU speed (343.63 fps & 235.34 fps) and iris/pupil detection were both found to be superior to those of competing technologies. The average precision, however, was lower than that of the competing solutions. More research is needed to refine and enhance this system, and finally, some areas to consider in the future are presented.

Keywords: Google MediaPipe, Eye-Mouse, OpenCV, Pyautogui, Python, Machine learning, Eyeblink, Mouse action

Santrauka

Šiame magistro darbe siūlomas metodas, skirtas neuro-motorinę negalią turintiems pacientams, kurie kartais gali prarasti gebėjimą bendrauti su kitais, nes jų būklė blogėja. Šiame darbe pateikiamas sprendimas, kuris palengvina šių asmenų bendravimą su kitais asmenimis pagal akių judesius ir kurį palaiko "Google MediaPipe". Mokslininkai iki šiol nenaudojo Google MediaPipe sąveikai su pele.

Kodavimui naudotas "Python" ir "Google Mediapipe" - sistema, skirta mašininio mokymosi vamzdynams kurti ir laiko eilučių duomenims apdoroti. Ši tarpplatforminė sistema veikia staliniuose kompiuteriuose, androiduose. Šiuo projektu buvo stebimi akies orientyrai. Gavus orientyrus, jie naudojami pelės žymekliui judinti naudojant modulį Pyautogui. Galiausiai, siekiant nustatyti siūlomos sistemos veiksmingumą, atliekamas tyrimas. Be to, bandomieji naudotojai išbandė sistemą be akinių, ir ši sistema gali nustatyti rainelę ir vyzdį geriau nei esami metodai. Sprendimas buvo palygintas su esamais pelės valdymo sprendimais. Nustatyta, kad tiek GPU/CPU greitis (343,63 fps ir 235,34 fps), tiek rainelės ir (arba) vyzdžio aptikimo greitis yra geresni nei

konkuruojančių technologijų. Tačiau vidutinis tikslumas buvo mažesnis nei konkuruojančių sprendimų. Šiai sistemai tobulinti ir gerinti reikia atlikti daugiau tyrimų, o galiausiai pateikiamos kelios sritys, kurias reikėtų apsvarstyti ateityje.

Reikšminiai žodžiai: Google MediaPipe, Eye-Mouse, OpenCV, Pyautogui, Python, Mašininis mokymasis, Eyeblink, Pelės veiksmas

Table of Contents

Introduction

New adaptive solutions that can increase the independence of persons with disabilities can be proposed, thanks to assistive technology. The use of assistive technology enables persons with disabilities to carry out basic everyday tasks that are required for living, working, and communicating with friends and family. Caregivers and technological solutions face significant challenges with various disabilities, such as neuro locomotor disability or amyotrophic lateral sclerosis [CBH15; LRF14]. In order to communicate with the outside world, patients with severe speech and motor disability who are unable to talk or utilize sign language need specialized human-computer interfaces [RSN09]. Communication devices must be customized to the user's needs based on the type of disability, ranging from the adaption of current devices (such as a mouse or joystick) to the development of technology (such as brain-machine interfaces for patients who are locked), [CXS17]. Disabled people who can regulate their gaze can use their eyes as a form of communication (for example, to control a wheelchair) [PRL14; EMM19]. Iris detection and tracking is a key component of human-computer interaction and has recently gained interest among researchers. Applications that require accurate and exact iris landmark detection include virtual reality, augmented reality, gaze recognition for customer behavior, computer control and handheld embedded devices. Iris detection and tracking has improved significantly so far. However, it is still difficult and computationally expensive to accurately detect iris landmarks in real-time. A lack of publicly accessible databases of annotated iris landmarks is another issue [AST22]. This study proposes an eye tracking-based control system in order to enable users to interact with computers in a natural and convenient way by using Mediapipe Iris, a Google product. The system integrates both and mouse functionality so that users can use their system to perform practically all computer inputs without the need for traditional input devices.

In real-world systems like computational photography and augmented reality, iris tracking has been widely used. Effective iris tracking on portable devices is still difficult because of limited computing, shifting lighting conditions, and occlusions like squinting or hair. In order to give precise iris estimation without the use of depth sensors, MediaPipe Iris, a cutting-edge machine learning model, has been developed by a group of Google AI researchers as a solution [Syn20]. The technology can measure the distance between the camera lens and the operator with a relative error rate equivalent to approaches that use depth sensors, according to experiments. The suggested model can record real-time landmarks affecting the iris, pupil, and eye features using a solitary RGB camera without the need for specialist hardware, drawing on Google AI's prior work with

MediaPipe Face Mesh. The method is based on iris size since, throughout a large population; the horizontal iris diameter of human eyes is generally consistent around 11.7±0.5 mm.

Over the past 20 years, much research has been done to create devices that can help persons with movement disabilities who also struggle with verbal communication (e.g., [ZJ05]; [DLS11]). Particularly in the digital era, the application of eye tracking and head movement recognition in empowering persons with disabilities has remained an intriguing topic. Eye tracking typically involves manipulation by measuring eye movements or watching what the eyes are doing. There is evidence from numerous studies using various eye tracking techniques that the science has relevance for society as a whole and for people with disabilities in particular.

However, certain people with severe disabilities, including those caused by amyotrophic lateral sclerosis or severe cerebral palsy, are unable to utilize their hands and legs or even their face muscles. Due to their severe disabilities, many common assistive technology systems are both expensive and ineffective for them [HGT11]. In these situations, an eye tracking device may be an alternative solution for those with severe disabilities who are able to move just their eyes. This study proposes a simple method that only requires a camera and Google's MediaPipe Iris software.

According to the figures provided by the National Family Health Survey (India), some 10 million people in India are disabled [Ane22]. These people have several handicaps, including mental retardation, mental illness, multiple disabilities, hearing, speech, or, relevant to this research, motor impairments. The scientific world is paying increasingly more attention to these problems, as a growing number of solutions are being put out to help such people live more independent lives. For their daily tasks, some of these people will require support from others. There are fewer people available to provide care in our society and in our rapidly expanding global population. Many members of these groups are living in social alienation, according to research trends, and it can be assumed that this number will rise. As a result, there will be a greater need for nurses and care providers. The problem is that there are not enough people to offer assistance.

In the past ten years, a lot of research has gone into creating intelligent software that can help these individuals. Several prototypes have been created throughout the last ten years, however there are few researchers who are still pushing for increased support. Driven by the needs of such people with motor disabilities for independency in their daily lives, a system is proposed in this study by which the disabled who have lost the use of their limbs can communicate using Computer mouse operated by eye movements.

Goal of the study

This study's goal is to develop a solution that makes it easier for patients having neuro-motor disabilities to interact with others on the basis of eye movements using a mouse, supported by Google MediaPipe

Objectives of the Study

The following are the objectives of the study:

1. To review the existing innovations in eye-tracking devices that help patients having neuro-motor disabilities to communicate, and identify the shortcomings in such devices.
2. To design a system that allows the disabled to communicate through the computer by eye movements using Google MediaPipe.
3. To implement this system and compare the results with existing methods.

Significance of study

Very few investigations have been carried out utilizing participants with movement impairments, according to the current literature in the field of brain computer interface for using information and communication technology and communicating with computers. From this perspective, this study is one of the few instances that can considerably help us understand the variables that can affect the suitability of a technology for touchless computer interaction, especially for users with motor impairments. With the use of this device, those with the condition can do mouse operation without contacting the physical mouse to the computer.

1 Theoretical Background Review of Literature

1.1 Background

With the development of IoT, Mark Weiser's [Wei91] dream of smart environments is swiftly becoming a realization. By 2023, there will likely be 50 billion connected gadgets, according to estimates [Eva11]. We will reap many benefits from this future, some of which are now evident. Systems that regulate our home's lighting by automatically switching the lights on and off when necessary lessen the effect on the environment and make life easier. Smart home sensors [Zwa22] can improve security and lower the likelihood of burglaries. One of the most well known manufacturers of home electronics in the world, Samsung, has promised that by 2020, all of its goods will be intelligent [SAM+22]. The proliferation of electronics around us raises new issues and concerns, such as how we will interact with them all. Independent user interfaces for each device or system, whether they are standalone or included in another device, would rapidly become too much to handle. It might make using these systems more difficult than it would be if we did not have them. Context-aware systems can be developed as a potential answer to this issue. Such systems can perceive and infer many environmental scenarios; hence, they do not need a user's express input to respond. Although there has been a ton of prior research in the field of context-aware technologies, context-recognition is still an extremely difficult task. To be able to understand the current condition, numerous sensors must cooperate and gather a significant amount of various data points. Context-aware systems may be crucial to achieving the goal of smart environments and promoting the IoT if this issue can be resolved [SDO17].

The ability of the eyes to convey a wealth of information about non-physical aspects and activities, like high cognitive loading [BGA08; RWK15], activity detection, like reading [JK17; KUS+13], and potentially emotion detection [BFA13], is very promising. However, eye trackers are not without their own drawbacks. For instance, head-mounted eye tracking devices are still intrusive and sensitive to movement. Remote eye trackers are subject to the fact that they may not necessarily have a direct line of sight towards the eyes and are only capable of a very narrow detection range, like the Tobii eye [Tob22] tracker. Yet, as eye trackers advance and become more precise, compact, and affordable, it is conceivable that in the future, it will be feasible to put these in smart glasses to render them entirely undetectable. This has recently been demonstrated by devices like Google Glass [JMP15] and Tobii eye trackers. Eye tracking technologies are employed in both commercial applications, such as smartphone apps [ZKM17] and driving assistance technologies [YSF17] as well as research projects [SB15]. Today's eye trackers are

much more advanced than those from just ten years ago. It serves as one of the main input channels for people with disabilities, too [Lan00; SR15]. Still, there is a lot of scope for improving further.

1.2 Evolution of the Technologies

The development of a PC camera-based eye-tracking and virtual mouse control system was developed by Kocejko [KBW08] an interface that people with disabilities may use in instead of a mouse. Pupil detection and screen to head position detection are two detection techniques that have been created and brought into use. The algorithm takes the arbitrary circular pupil shape for reference. The pupil's longest vertical and horizontal lines are then identified. The algorithm for screen detection is used to adjust all possible head movements in relation to the screen. A system developed by Missimer and Betke [MB10] extends the capabilities of earlier camera-based binary-switch systems developed by Grauman [GBL$^+$03] and Chau and Betke [CB05]. This gives users access to a more flexible system. The technology gives a user mouse pointer control on a level with a conventional mouse by analyzing the movements of three face regions, including both eyes. Further, research found that during system development, the system was more accurate in categorizing the eye state during head turns when employing both an open-eye and a closed-eye template, instead of one template. The implementation of an eye-based HCI approach and system that uses pupil knowledge has been effective. Without requiring any prior calibration, it demonstrates more than 96% success rates for controlling, selecting, and determining key and mouse events [AM11]. For those with severe physical limitations like amyotrophic lateral sclerosis (ALS), an eye-gaze input system was created by Abe[AOO11]. To detect eye-gaze in environments with natural lighting, the system makes use of a personal computer and a home video camera. Cecotti [Cec16] adopted a limbus tracking method, which is also the widely used technique for horizontal eye-gaze detection, to improve the resolution of vertical eye-gaze detection. In the evaluation of the system, 18 adults participated under three settings that corresponded to three modalities: direct selection using a mouse, eye-tracking with a switch to pick the appropriate command, and solely eye-tracking. The speed and information transfer rate (ITR) at both the command and application levels were used to evaluate how well their virtual mouse performed. The typical speed among subjects was 9.30 letters/min with the eye-tracker alone, 15.26 letters/min with the eye-tracker and the switch, and 18.43 letters/min with the mouse alone. A customizable and adaptive mouse emulation known as ETM (Emulator Mouse) was created by Henzen and Nohama [HN16] as part of an application. The software adapted the contacts of the buttons on a joystick to capture trigger signals connected to the ze(Universal Serial Bus). It sent mouse commands for any Windows application using a command processor and a

key layout customized to user needs. Their tests, which were carried out with volunteers who had ALS and cerebral palsy in the project participants' schools, produced satisfactory results. A revolutionary eye control system proposed by Zhang [ZKM17] integrated both mouse functionalities in order to increase the accuracy, mobility, and usability of eye tracking technology in user-computer interaction. Their suggested solution focused on offering a simple and practical interactive mode that only requires the user's eyes. To compare the suggested eye control tool with an existing system, two interactive activities of varying difficulty (article searching and surfing multimedia websites) were used in their study. Currently few script-specific virtual mouse optimization solutions include multimodal input access. An analogue frontend was created in De Lucena and Conzelmann's [DC19] study to evaluate EOG (electrooculogram) signals in a microcontroller and transform them into changes in computer mouse position. The baseline drift, which is the most frequent issue with EOG readings, is fixed via a drift correction method. Different methods for using the mouse have been studied, and the most effective way involved controlling it by issuing discrete orders that generate defined movements. The model proposed by Bazarevsky [BKV+19] can serve as the first step in any face-related computer vision application, such as 2D/3D face key points, contour or surface geometry estimation, running on full image or video frames, facial features or expression classification and facial recognition segmentation. Their contributions include a MobileNetV1/V2-inspired lightweight feature extraction network, a more GPU-friendly anchor technique from single shot multi-box detectors (SSD), and a better tie resolution strategy of non-minimum suppression. Pupil tracking from live video on mobile devices was proposed by Ablavatski [AVG+20]. To execute the neural network on mobile devices, Lee [LCI+19] employed TensorFlow Lite with GPU backend coupled with MediaPipe [LCJ+19], a framework for building perception pipelines. This method extends a state-of-the-art face mesh detector with two new components: a tiny neural network that predicts positions of the pupils in 2D, and a displacement-based estimation of the pupil blend shape coefficients. Their technique can be used to control the pupil movements of a virtual puppet accurately, and lends liveliness and energy to it. Their proposed approach runs at over 50 FPS on modern phones, and enables its usage in any real-time puppeteering pipeline.

1.3 Existing methods

1.3.1 Eye tracking algorithms

Eye tracking has been used with a variety of different methods. Existing eye tracking algorithms include both model- and appearance-based methods. Prior to eye detection, some projects used facial detection. Only the eye blinking was recorded; neither the face recording nor

the gaze recording has been done. A way for improving gaze-tracking system performance using a huge screen at a distance was presented [LHP13]. This method can be evaluated in a variety of settings, including gaze detection on a small display. Praglin and Tan [PT14] first performed color-based eye detection before masking off non-face regions until only two are left, and finally computing the centroid of related regions. After detecting the eyes, the face is located by computing mean color mark pixels in the image's mean color space. The homography model, RANSAC, and SIFT are used to estimate gaze. Then, Wood [WBM16] built a geometric model for eyes to predict the corresponding gaze, including both 2D and 3D models that used near infrared (NIR) illumination to create corneal reflections to estimate the gaze vector. Some new applications of eye gaze such as effortless scrolling and enriched reading are aimed at general users. Fatima [FUZ16] investigated various eye gaze estimation techniques and algorithms. The latter directly maps the raw pixels to the gaze angles [PSH18]. Appearance-based methods, in general, have surpassed model-based ones for eye tracking, especially when being equipped with advanced deep learning methods. Different deep neural network (DNN) structures have been proposed to enhance the performance of gaze estimation. For example, Zhang [ZSF15] proposed the first DNN model for gaze estimation and Park [PSH18] further proposed a hybrid network integrating both Hourglass [NYD16] and DenseNet [HLV+17] to leverage auxiliary supervision based on the gaze-map. Cheng [CZ18] introduced asymmetric regression-evaluation network (ARE-Net), which consists of two smaller modules to first find directions from each eye individually and then estimate the reliability of each eye, respectively. Zhu and Deng [ZD17] defined two convolutional neural networks (CNNs) to predict head and gaze angles, respectively. In parallel, different processing pipelines have been developed with diverse focuses on the input features. For example, Zhang [ZSF15] utilized minimal context by only using grayscale eye images and head poses as inputs. A multi-model CNN to extract information from two single eye images, including face image and face grid, was developed by Krafka [KKK+16], for aiding the following gaze estimation. Fischer [FCD18] built an ensemble on top of the features extracted by two eye patches, and head pose vectors and achieved superior performance on several datasets, as did Sugano [SMS14]. Accurate eye segmentation can improve eye-gaze estimation and support interactive computing based on visual attention; however, existing eye segmentation methods suffer from issues such as person-dependent accuracy, lack of robustness, and an inability to run in real time. The RITnet model proposed is a deep neural network that combines U-Net and DenseNet. An unsupervised boosting strategy was suggested by Hosp [HEM+20] for pupil detection that would only compute the most crucial locations on the circle outline in order to achieve an incredibly quick (0.78 ms) detection. The system created by Sharma was based on

facial and eye-gaze detection using attention-based gaze estimation (AGE-Net) network (as per [Bis21]) and hand detection using MediaPipe (Google MediaPipe[1], 2021), but it can be easily expanded to track other body parts for various tasks. They used MediaPipe output directly in their simulation and user study, but to build a buffer zone around the hand, a greater separation distance was required. By replacing a different object identification method for the MediaPipe algorithm, the algorithm can be modified to avoid various body parts or objects. They have a four-way and point-to-point control interface superimposed on their video see-through interface that allows them to move the robotic arm to any random location inside its field of vision. Users can move the robotic arm to any location on the screen by moving the mouse pointer, and the UI had one language-agnostic screen. As demonstrated by Souza [SAM+22], the AIT2-UX framework can assist in making an evaluation of UX by including several tracking approaches. They created a system that uses artificial intelligence algorithms, eye and mouse tracking techniques, mouse input, self-assessment questionnaires, and ways for evaluating user experience to classify users according to performance profiles. The outputs generated by this framework areartifactst that may be utilized to facilitate website UI (User Interface) customization.

1.3.2 Landmark detection

A learning-based strategy for eye region landmark localization that makes older appearance-based techniques competitive with more modern ones was proposed by Park [PZB18]. Their approach outperforms state of the art for iris localization and eye shape registration on real-world imagery, despite having only been trained on synthetic data. The algorithm recognizes eyes based on 68 face landmarks that must be found on every face, including the eyes, nose, lips, eyebrows, and face area. These landmarks are used to predict the form of the face [ACR22]. The hybrid approach (supervised and unsupervised approach) used by Laddi [LP19] involved pupil detection in each eye patch using an image gradients algorithm and supervised regression-based eye patch detection using discovered landmark points. The use of 68 facial landmarks detectors was used for eye detection by Islam [IRS21]. The face landmark localization was modified by Roy and Chanda [RoC22] to localize the eye points because it typically localizes 68 points on the face. These 12 spots on the face were located using a customized Dlib shape predictor. By measuring the distance between two landmark points—one from the top of the eye and another from the bottom—the eye blink was recognized. Their system successfully distinguished between open eye state and close eye state, as it only needed to keep track of the distance between two specific landmarks at a point

[1] Google MediaPipe, Available at https://google.github.io/mediapipe/solutions/hands.html

in time. Radmehr [RAM21] employed the 468-landmark solution of Lugaresi [LCJ+19] in their experimental investigation on the imitation of the human head-and-eye position with the 3-degree of freedom (DOF) spherical parallel mechanism applied with ROS and MediaPipe Framework. The MediaPipe face landmark solution uses geometry to estimate landmarks in three dimensions, including the face, eye, and mouth. A two-dimensional landmark was used together with a simple linear regression model to create outputs that are fuzzy and range from -10 to 10. A benchmark dataset was introd,uce usedd a robust frameworkwa provided for the localization of key landmark points to extract the iris with better accuracy by Adnan [AST22]. A number of training sessions were conducted for MobileNetV2, ResNet50, VGG16, and VGG19 over an iris landmarks dataset, and ImageNet weights were for model initialization.

1.3.3 Eye control-based mouse

A system developed by Ghani [GCS+13] uses a laptop's built-in webcam to capture live image frames, which the GazePointer GUI then analyses to determine the user's Point of Gaze (PoG). The extracted eye patch from the collected frames is displayed in the first area, while the results of the Hough circle transform are also shown. The area under it is the "Test Area" for mouse pointer movement. ETM system was provided by Henzen and Nohama [HN16]. The mouse emulation are similar. The user chooses a line, after which he chooses the button that will carry out the desired action. To make using the mouse easier, the mouse simulator includes a number of buttons for various functions. For instance, a button performs two operations simultaneously to avoid the user's need to select a button for clicking the mouse and then select a different button for moving the mouse, or "click and dr."". A successful method based on MATLAB was created and tested by Nasor [NRZ+18] for face detection, ocular region extraction, iris tracking, and cursor control. The computer screen's test GUI, which consists of nine boxes, was used to verify the cursor movement. However, the movement up and down was not very reliable, and the mechanism has to be improved. Eye tracking can be a useful tool for accessing social media, however, traditional methods or eye mouse emulation cannot address the complex design issues of eye-controlled interfaces. Kumar [KMS17] developed a gaze-adapted interface that enables non-invasive gaze-based Twitter participation. They compared the proposed interface to OptiKey, a popular method for operating computer applications using simulated mouse input. Najla and Jayashree, [NJ18] proposed an approach which involves use a camera to track a real-time video sequence of a human face in order to extract the eye region. Then, identify the face features to create the eye region and obtain the gaze point. Next, for each gaze point, a cursor movement is made on the computer screen. Their central focus is the use of edge strength and intensity energy

to locate the eye corner detector, which in turn locates the iris center and eye vector. Eye tracking was employed in this method to solve the Midas touch issue for persons with severe disabilities. They compared the following three conditions on 10 participants to assess performance: gaze detection with mouth switch, gaze detection with dwell duration while allowing for the distance to the closest command, and gaze detection within the command box's surface. A workload using the NASA-TLX test was conducted and the results showed that it had a faster typing speed (36.6 ± 8.4 letters/min). A vision-based computer mouse controlled by head movements and eye blinks was developed by Arslan [ABA19] to help people with spinal cord injuries operate computers. A sensor-based solution was first used for head mouse systems, but these are expensive, uncomfortable and provide limited data for future computing. Thus, they developed a system that could be used without any hardware components to overcome these difficulties and also allow disabled persons to operate computers. Meena [MCW+19], in their expe,riment compared finding and choosing characters on a touch screen to using a mouse. When using a mouse, the user must click on the desired object, whereas when using a touch screen, the user must simply touch the desired item. The mouse-only condition was included to test how well the GUI worked without a touch screen. The search for a target item is accomplished through gaze detection, and the selection can happen using a dwell time, soft-switch, or gesture detection using surface electromyography in asynchronous mode. In synchronous mode, both the search and selection can be accomplished with just the eye-tracker. Zheng and Usagawa [ZU20] used a webcam with a resolution of 640×480. By checking the color intensity of the eye areas, first create an initial pupil pattern. Then, based on the matched pupil position in the subsequent frames, estimate the gaze. The estimate result is then calibrated using mouse clicks from computer users, and the pupil's position is then noted to train the weight map of the pupil pattern. The finding showed that their approach could reduce the impact of bright spots while also improving the accuracy of pupil identification and eye tracking. A raspberry pi is used in the proposed study to measure eye movement and control the cursor. Khare [KKS19] created a system using Raspberry Pi, a singsingle-boardlication that enables users to move the mouse and select a certain folder using only their eye. Their suggested method shows accuracy and speed, which are sufficient for some real-time applications and allow handicapped persons to recognize a variety of computer tasks. In addition, these newly developed assistive multimodal electrical devices may be used efficiently without the user using their hands, and they can still operate a computer even if their fingers or arms are inactive. The recognition method illustrated by Abiyev and Arslan [AA20] is based on CNN, which makes use of low-quality images taken by a computer’s camera. The CNN converts head movements into the real mouse coordinates. The system’s architecture enables persons with disabilities to blink and move

their heads to control mouse buttons and the mouse cursor. With the help of this device, persons with disabilities can operate mouse cursors and buttons without any assistance. Meena [MCW+19] developed an algorithm that enables persons with physical disabilities to utilize a mouse by moving their face and eyes. They can scroll up and down, left and right-click, and move the pointer left and right. The system requires only the minimum requirements, including a webcam, NumPy, dlib, and a few other basic libraries. In order to translate eye motions like blinking, staring, and squinting onto mouse cursor actions, Vasisht [VJS19] developed a system that functions as an eye-based interface. The software for the system, which uses a basic webcam, includes Python (3.6), OpenCv, numpy, and a other libraries required for face recognition. A face detector can be created by utilizing the HOG (Histogram of Oriented Gradients) feature, a linear classifier, and the sliding window method. It is hands-free and requires no extra hardware or sensors. In order to control the mouse without the usage of additional hardware, Kastrati [KPW+21] technique can automatically determine head direction and eye states from camera images. They created a new dataset and standard to advance study to examine the relationship between brain activity and eye movement. Participants must indicate if one of the two target symbols appears among the five search symbols for each row by clicking with the mouse on the 'YES' or 'NO' symbol. The EEGEyeNet dataset consists of simultaneous electroencephalography (EEG) and eye-tracking (ET) recordings from 356 distinct subjects from three different experimental paradigms. They provided a benchmark to analyze gaze prediction from EEG measurements using this dataset. The benchmark consists of three tasks left-right, angle-amplitude, and absolute position each of which is getting increasingly difficult. The system is designed to work across a broad range of contexts with motion inputs; for example, in a hospital using complex user interfaces, playing games, exercising, for rehabilitation, and for creative tasks. Webcam eye image and a lightweight CNN were both used by Huang [HZX+21] to accomplish the appearance-based gaze estimation. To replace the traditional hardware used for human-computer interaction, such as the mouse , eye control mouse and character input methods are developed using the results of gaze estimation to perform accurate interactive operations with the computer. The head movement module developed by Kummen [KLH+21] is built using Dlib[2]'s prebuilt model5, which quickly detects faces and correctly predicts 68 2D facial markers. Specific activities and facial expressions can be recognized and matched to particular mouse events using these predicted facial landmarks. For example, a yawn or a blink can be identified by calculating the mouth-aspthe ect-ratio, and the left/right profile of a face can be used to initiate mouse click events. A Bluetooth mouse or a wireless mouse still uses devices

2 Davis King, 2015, dilb-models [Source code] https://github.com/davisking/dlib-models

and is not entirely device-free because it requires a dongle to connect to the PC and a battery to power it. This limitation is solved by Shriram [SN+21] using an AI virtual mouse system that records hand gestures using a webcam or built-in camera and recognizes fingertips using computer vision and deep learning algorithms. Latent features are encoded using GRUs, and converted into keypress probabilities by a fully connected layer. Peresunko [PPS+22] proposed a mouse for computer vision-based cursor control. Users of this software can quickly set the active region, which affects how the cursor moves on the screen. The assistant chooses the area that is now active and enables for reference point movement. The user can navigate the GUI and mouse to open files, launch program, browse the Internet. As a result, this tool enables interaction between people with disabilities and their environment. Prameela [PLM+22] used facial gestures to construct a virtual mouse. In the technological age, using a computer requires interaction. Using a mouse is quite challenging for people with physical limitations; however, it is now feasible. They can only use their face to control the mouse's functions. The blink detection mechanism can only be used if the eye is not moving. The face will be used to move the cursor on the screen rather than the mouse, and the blink will be used to click on the eyes.

1.4 Research Gap

The following are the research gaps:

- A new prototype for voice commands and head movements working together to control a computer was developed by Ismail [IHH11]. Their system is a multimodal interface designed for people with severe disabilities that is used to control computers. However, when a dark background that should be blank is present, the system fails to function properly.

- Based on eye gaze monitoring, [GCS+13] developed a real-time mouse pointer control. Their system has various limitations, such as the requirement that the user's head be at the same altitude as the webcam and that the distance between the user's eyes and the webcam be between 20 and 75 cm. It is not possible to utilize this device with wearing glasses, and sufficient illumination and head stillness are required. The system is only tested for frontal faces.

- An open-source Matlab toolkit that is used to interact with the Tobii EyeX device for gaze recording in behavioural research was developed and tested by Gibaldi [GVB+16]. However, the Tobii EyeX's applicability for tracking micro-saccadic eye movements or for

in-the-moment gaze-contingent stimulus control is hindered by the comparatively low sample rate and moderate precision.

- In order to develop design standards for eye-controlled systems that take into account the possibility that accuracy and precision can fluctuate significantly between various tracking settings, it is necessary to be aware of the characteristics and limitations of eye tracking [FWT+17].

- Zhang [ZYC18] designed the CNN model for the 9-direction gaze estimate. They use the nine key T9 input mechanism, which is popular in mobile, based on the nine-direction gaze. The range of the gaze angle is wider in the off-screen mode than it is in the screen mode, where it is limited to the screen area. But its capabilities were limited, and it was not very flexible.

- A significant improvement has been made so far in iris detection and tracking. However, iris landmarks detection in real-time with high accuracy is still a challenge and a computationally expensive task [AST+22].

- Salunkhe and Patil [SP16] analyzed three techniques with their drawbacks includes the limbus tracking, pupil tracking, and EOG. As the eyelid frequently hides all or part of the limbus, limbus tracking technique is negatively impacted. As a result, only horizontal tracking is possible with it. The use of infrared light is typically not a part of this technique [AOO11]. In contrast to limbus tracking, x-y tracking (pupil tracking) is more practical because the pupil is never truly covered by the eyelid. The drawback is that border identification is more challenging because there is less contrast between the pupil and iris than between the iris and sclera [KKS09].

- The main drawback of EOG is its poor video tracker accuracy in determining gaze direction. The main problem that can restrict someone from using a gaze-based system is if they have nystagmus, an involuntary eye movement disorder that can impair vision [SC18]. Disorders of the central nervous system, diseases (such as brain tumours), and toxic factors (such as alcohol intoxication) can all contribute to it (e.g., cerebral ataxia).

- When considering an eye-controlled computer mouse, the image of a system that places the pointer at the exact location of the user's eyes on the screen and makes it follow the user's eye movements is likely to appear. De Lucena and Conzelmann [DC19] work was based on this vision and an attempt to determine the eyes' exact positions and gaze

directions. It became exactly evident that using the EOG signal would require various limitations, such as a fixed head and low resolution. Due to these limitations, a different approach was used that is centered on the concept of a keyboard. Eye movements can be used to provide predefined commands, creating a unique pattern in the EOG signal that is recognized by software and it causes a corresponding movement of the mouse cursor.

- Kim [KYY19] employed a palm tablet rather than a smartphone with a more widely used screen size. Detecting eye movements when the target viewing area is as small as a smartphone screen remains a technical challenge despite advances in eye-tracking technologies.

- AI virtual mouse has some limitations such as small decrease in accuracy of the right click mouse function and also the model has some difficulties in executing clicking and dragging to select the text [SN+21]. One limitation of system is the gaze estimation technique in combination with the camera setup [HEM+20]. It is considered that (1) because they did not employ a chin rest to create an environment with as little restrictions as possible, gaze estimation error is influenced by head movements; and (2) because adjusting 13 unknown variables at once is a very challenging task, the 3D eye-model parameter optimization procedure is prone to errors. A lower resolution results in a larger scene coverage for each pixel and, consequently, a higher sensitivity to changes in illumination (light). As a result, even little changes in the shape (contour) of the features appear to have a significant impact on the precision with the chosen resolution.

- In addition, the camera's range of view is limited, so it is possible that users mistakenly step outside the recording region. Furthermore, the accuracy of the tracking may be hampered depending on the proximity and angle of the camera to the user [HB+22]. Additionally, because a remote working environment is the use case for their proposed pipeline, the users' distance from and position in relation to their PCs are likely to be relatively constant throughout the day. Thus, the eye tracking system needs to be improved into a system that is more comfortable for the disabled.

The present work attempts to address these shortcomings of these previous studies.

2 Methodology

2.1 Overview

The model proposed by Bazarevsky [BKV+19] was the first step in any face-related computer vision application, such as 2D/3D face key points, contour or surface geometry estimation, running on full image or video frames, facial features or expression classification and facial recognition segmentation. Their contributions include a MobileNetV1/V2-inspired lightweight feature extraction network, a more GPU-friendly anchor technique from single shot multi-box detectors (SSD), and a better tie resolution strategy of non-minimum suppression. Pupil tracking from live video on mobile devices was proposed by Ablavatski [AVG+20]. To execute the neural network on mobile devices, they employed TensorFlow Lite with GPU backend [LCI+19] coupled with MediaPipe [LJH+19] a framework for building perception pipelines. This method extends a state-of-the-art face mesh detector with two new components: a tiny neural network that predicts positions of the pupils in 2D, and a displacement-based estimation of the pupil blend shape coefficients. Their technique can be used to control the pupil movements of a virtual puppet accurately, and lends liveliness and energy to it. The proposed approach runs at over 50 FPS on modern phones, and enables its usage in any real-time puppeteering pipeline. On the other hand, it controls eye movements even in people with brain injuries.

It is possible to think of the human eye as a hardware replacement for computers. An Internet protocol camera was used by Khare [KKS19] to capture an image of an eye frame for cursor movement. In this regard, the eye's function is first considered. A Raspberry Pi, which also controls the computer's mouse cursor is used to identify the pupils. By using the Python programming language's Open-Source Computer Vision module, it is possible to derive an Eye Aspect Ratio (EAR) for this task that correlates to the eye's blinks (left or right). Their main goal was to improve the computer experience for people with physical disabilities by assisting them in overcoming challenges like using a mouse. The hand-tracking algorithm of MediaPipe consists of two models that work dependently, i.e., the palm detection model and the landmark model [NUS+22]. The palm detection model gives a perfectly cropped image of the palm and then gives this image to the landmark model. This process diminishes data augmentation, which is required in deep learning-based landmark localization.

Monitoring eye movements can be a useful tool for generating control signals for wheelchairs. The Motion Input v2.0 application was developed by Kummen [KHG+21] which accepts the video stream from a normal RGB webcam as input by using additional gesture

definitions defined and accessible through open-source frameworks. It transforms human motion gestures into input for already-existing applications and video games. The user can choose from a range of motion types, such as single and bimodal hand gestures, repetitive exercises involving the entire body or only the extremities, head and facial movements, eye tracking, and combinations of the above. Along with monitoring of facial expressions such lip movements, blinking, and head direction with rotation, gestures for idle states, auto calibration, depth acquisition from a 2D RGB webcam stream, and gestures for idle states are also included. They further provided a number of unique gesture detection classes as DirectInput triggers. User can map gestures to specific mouse events by developing a collection of gestures that the program can recognize. The proposed system by Lavanya [LBS+22] utilizing OpenCV, the computer's cursor movement is controlled by eye movement. Their system combines keyboard and mouse operations so that users can use their system to perform practically all computer inputs without the usage of conventional input devices. A system that depends on facial expressions including eye and mouth movement was presented by Bharath [Bmr22]. The system recognizes the region of the face, eyes, and mouth are detected, and they are extracted for processing by using the Haar classifier. It allows for hands-free contact between people and computers and controls the operation of the virtual mouse. It enables disabled persons to perform cursor movement using a virtual mouse, including up, down, left, and right scrolling.

2.1.1 MediaPipe framework

MediaPipe allows a developer to prototype a pipeline incrementally. A pipeline is defined as a directed graph of components where each component is a Calculator. The graph is specified using a GraphConfig protocol buffer and then run using a Graph object. In the graph, the calculators are connected by data Streams. Each stream represents a time-series of data Packets. Together, the calculators and streams define a data-flow graph. The packets that flow across the graph are collated by their timestamps within the time-series. The main components of MediaPipe framework are described in the following sub-sections.

2.1.1.1 Graph

Processing takes place inside a graph, which defines the flow paths of packets between nodes. A graph can have any number of input and outputs, and branch or merge data. In a graph, source nodes that do not have input streams but still produce packets can be the source of data flow (e.g., by reading from a file). A graph input stream, which enables an application to feed

packets into a graph, is another source of data flow (e.g., passing in camera texture obtained from the operating system) [LJH+19].

2.1.1.2 Calculators

Every node in the graph is a “calculator” when it is implemented. The majority of the graph's activity occurs at nodes. Many in and output ports are defined by each node's interface. Calculators are customizable; they all originate from the same base Calculator class and contain the following four fundamental methods: Open(), GetContract(), Process(), and Close() [LJH+19].

2.1.1.3 Packet

The basic data flow unit is called a “packet.” It consists of a numeric timestamp and a shared pointer to an immutable payload. Any C++ type may be used for the payload, which is also known as the packet's type. Packets can be inexpensively copied and are handled as value classes. With reference-counting semantics, each copy equally owns the payload. Every copy has a unique timestamp [LJH+19].

2.1.1.4 Streams

Each node in the graph has a “stream” connecting it to every other node. A stream is a connection between two nodes that carries a sequence of packets with increasing timestamps. Any number of the same sort of input streams can be connected to a single output stream. Each input stream receives a unique copy of the packets from an output stream and keeps track of these packets in a separate queue so that the receiving node can process them at its own pace [LJH+19].

2.1.1.5 Side packet

A single packet with an undefined timestamp travels between nodes on a side-packet connection. In contrast to streams, which indicate a flow of data that varies over time, it can be used to offer some data that will remain constant. For example, a side packet can be used to feed a node the string defining the file path for an ML model [LJH+19].

2.1.1.6 Solutions

MediaPipe can be used for various application such as face detection, iris detection, hand position detection, holistic detection, and more. Face Detection [GAK20; RAM21] using MediaPipe enables accurate face mesh prediction in real time. Iris detection [AVG+20] can be done, and real-time pupil tracking from live video on mobile devices is made possible by Mediapipe. Hand position (landmark) detection was examined in some studies using MediaPipe [ZBV20; IRA+22]. Holistic detection was studied by Singh [SKA21] and Agrawal [ACR22], in

which the pose, facial, and hand landmark models from MediaPipe Pose, MediaPipe Face Mesh, and MediaPipe Hands were used.

MediaPipe[3] depends on OpenCV for video and FFMPEG for audio data handling. It also has other dependencies like OpenGL/Metal, Tensorflow, Eigen, etc.

2.2 Data collection

GazeCapture: There is significant variation in illumination, head pose, appearance, and background [KKK+16]. This variation allows us to learn robust models that generalize well to novel faces. Simulated and real datasets of eyes looking in different directions are available at Eye Gaze. The synthetic dataset has the gaze information generated by UnityEyes with a predefined look vector[4]. The overview notebook covers what this vector means and how each component can be interpreted. It would be very useful to have a simple, quick network for automatically generating this look vector from an image [SPT+17].

2.3 Proposed Method

A physically disabled person may find it difficult to operate the mouse. Mouse cursor control utilizing eye movements is proposed as a potential alternative for those who are unable to use a mouse physically. By using eye movements to control the mouse, eye gazing is an alternate method of using a computer. Eye gazing is a possible approach that enables a user to manage their computer by blinking their eyes movement of eyes for those who find touchscreens and mouse to be inaccessible. For those with physical disabilities, eye movement might be seen as a crucial real-time input medium for human-computer communication. An eye-controlled mouse using MediaPipe is proposed in this system using Webcam and without needing any additional hardware in order to enhance the dependability, portability, and usability of eye tracking technology in user-computer communication. The suggested system is focused on offering a simple and effective interactive mode that only requires the user's vision. The proposed system's usage flow is created to mimic human natural tendencies completely. The proposed approach outlines the implementation of both iris and cursor movement in response to iris position, which may be used to control the cursor on the screen while utilizing a webcam and MediaPipe in Python.

[3] Available at: https://learnopencv.com/introduction-to-mediapipe/
[4]Eye Gaze dataset available at: https://www.kaggle.com/datasets/4quant/eye-gaze

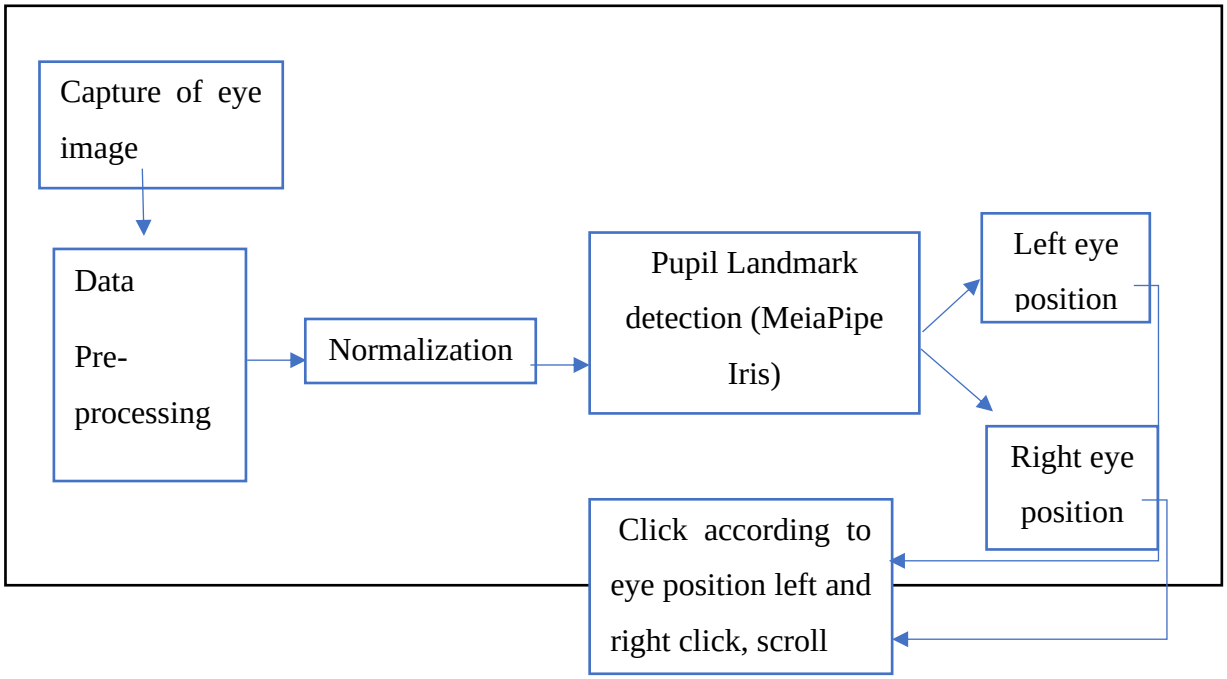

Figure 1: Overall process of landmark detection using MediaPipe

In the proposed system, the mouse is controlled by eye movement using OpenaCv. Camera detects the eyeball movement, which can be processed in python. By this, mouse cursor can be controlled. The user has to sit in front of camera or display screen of computer or laptop a video camera established on screen to capture eye movement of user. The laptop constantly analyses the video image of attention and determines where user is looking at a display screen. To pick out a key, user looks at a key for a period of time, and to "press" a key, the user just blinks the eye. On this device, calibration procedure is not required. For this system, no expensive hardware is required. To detect face landmarks for selecting points around specific facial features (eye, iris, lips, face boundary), MediaPipe used with Python and OpenCV package. The camera gets inputs from streaming as it is taken as frames. After receiving frames, it will check for lights conditions. The captured frames, which are already in RGB format, are transformed into binarized (black and white form) images. Images (frames) focusing the eye are analyzed for iris detection (middle of eye). The iris model takes an image patch of the eye region and estimates both the eye landmarks (along the eyelid) and iris landmarks (along the iris contour). The overall process is described in the following sub-sections.

2.3.1 Data pre-processing

The dataset should be normalized before giving landmarks to the model [RAM21] because redundant information affects the results, such as distance from the face to the camera or even the

size of the face. In addition, landmarks extracted from the dataset should be in the center of the frame because the face position in the frame does not affect its angulation.

Data acquisition

This is one of the important factors for obtaining a good result. The existing iris databases internationally available are used. These samples are available in raw form and pre-processed form. The pre-processed samples are used for testing the algorithm. A good and clear image eliminates the process of noise removal and also helps in avoiding errors in calculation. In this case, computational errors are avoided due to absence of reflections, and because the images have been taken from close proximity. The Infrared light can be used for illuminating the eye, to avoid any secular reflections [AT10]. Using a web camera, the image of the eye and head movement can be captured. The position of the pupil's center is determined using OpenCV code by focusing on the eye in the image. Then, distinct values of x and y coordinates will be specified for a certain command, using the pupil's center location as a reference.

Normalization

Data normalization is the process of assigning the same value to each input parameter or pixel. Because each input pixel in the image data has a value between 0 and 255, the pixel values are normalized to values between 0 and 1.

2.3.2 Landmarks detection

In order to extract eyes, landmarks of eyes are needed. Since MediaPipe provides us with the landmarks in normalized values, they need to be converted into pixels, or coordinates relative to the image plane. The iris model takes an image patch of the eye region and estimates both the eye landmarks (along the eyelid) and iris landmarks (along this iris contour).

In the Face Mesh, 468 landmarks are obtained. Looping through each landmark is necessary. There will be x and y values, and for conversion purpose width is multiplied with x, and height with y. The results would be pixel coordinates, storing them in the list of tuples (x, y) coordinates of each landmark. MediaPipe Iris detects 71 eye landmarks (or key points) and 5 pupil key points. This simple function does nothing but turns normalized into pixel coordinates.

2.3.3 Detecting motion

The easiest approach possible is taken by focusing on the pupil. By converting the image into grayscale format, it is seen that the pupil is always darker than the rest of the eye, no matter

where the eye is looking at and no matter what color is the sclera of the person. First, conversion to grayscale is carried out and then the threshold to extract only the pupil is found.

2.3.4 Mouse click and selection

In this part, one portion of the screen is selected by using eye gaze. For eye movements to control the mouse, however, the program primarily relies on the camera and the design of virtual screens known as surfaces. With use of the data from current gaze at the world (instead at the screen) cursor is moved to the appropriate coordinates on the screen with respect to that gaze. For this, the gaze ratio is calculated from the left and right eyes. It will also examine the change in eye aspect ratio while reading, and if it falls below the threshold for either eye, that particular click is made (right-click, left-click).

2.4 Implementation

The system implementation process includes hardware and software requirement which is discussed in this section.

2.4.1 Hardware and software requirement

2.4.1.1 Minimum system requirement

- Windows 8 and above OS
- CPU: one or more 32/64-bit CPUs, with a minimum clock frequency of 1.5Ghz and it is recommended to have a 2Ghz or faster multicore CPU.
- RAM: 2GB of RAM is required as a minimum.
- Disk Space: Locally attached storage with 16 GB of disk space as a minimum 60 GB of disk space is recommended.

2.4.1.2 Hardware Component

- **Web Camera**

The video stream is captured by an internal built-in webcam and used for processing afterwards.

2.4.1.3 Software component

Python and libraries: Python[5] programming language (Python 3.11.0) is used in web development, machine learning applications, along with all cutting-edge technology in Software Industry.

[5] Python available at: https://www.python.org/

- **Mediapipe** is a cross-platform library developed by Google that provides amazing ready-to-use ML solutions for computer vision tasks.

- **OpenCV** (Open-source Computer Vision) library in python is a computer vision library. OpenCV library[6] provides privileges to play with different images and video streams and also helps in end-to-end projects like object detection, face detection, object tracking, etc.

- **TensorFlow**[7] is an open-source software library for high-speed numerical computing. Because of its adaptable design, computing may be easily deployed across a range of platforms (CPUs, GPUs, and TPUs), from desktop computers to server clusters to mobile and edge devices. Its flexible numerical processing core is used across many other scientific areas, and it has significant support for machine learning and deep learning. It was originally created by researchers and engineers from the Google Brain team within Google's AI division. PyAutoGUI[8] enables Python scripts to automate interactions with other apps by taking control of the mouse. PyAutoGUI has features such as (i) Using the mouse to navigate between various application windows and click, (ii) taking screenshots to locate an image on the screen, such as a button or checkbox, and capture it, and (iii) users can move, resize, maximize, minimize, or close an application's window by finding it (Windows-only, currently).

The domain of using NumPy includes linear algebra, Fourier transform, and matrices. Python also uses the win32api library (PyWin32) which enables us to access to various the Windows APIs from Python.

2.5 Comparison with existing models

The following models were used to compare the achieved results using the proposed model:

2.5.1 Haar Cascades model

Viola and Jones (2001) proposed the use of this strategy. Like the straightforward CNN, it pulls a lot of characteristics from photographs and is exceptionally quick to use. Adaboost is then used to choose the best features. As a result, just 6100 features remain from the original 165000+ features. However, it will still take a long time to implement all of these functionalities in a sliding window. The features are then bundled in a Cascade of Classifiers, which was then introduced.

[6] OpenCv available at: https://opencv.org/
[7] Tensorflow available at: https://pypi.org/project/tensorflow/
[8] PyAutoGUI available at: https://pyautogui.readthedocs.io/en/latest/

These subsequent characteristics in that cascade are not analyzed if a window fails at the initial level. If it succeeds, the process is repeated for the following feature. A window is categorized as a face region if it can pass all the characteristics. The trained XML files as well as the OpenCV library are both included with the Haar cascades, which require a large number of positive and negative images for training. The crucial advancement in facial recognition was made possible by Haar Cascades. Although it greatly increased the speed and precision of the detections, it has certain constraints and was unable to successfully find faces in noisy imagery. Numerous advancements have been made over time. In addition to Face Detection, the Haar Cascade technique was also employed for Eye Recognition, License Plate Recognition, and other tasks.

The classifier examines the pixel intensities and looks for several specified features in the image (Figure 2). It will know there is an object if it gets enough matching for a particular area.

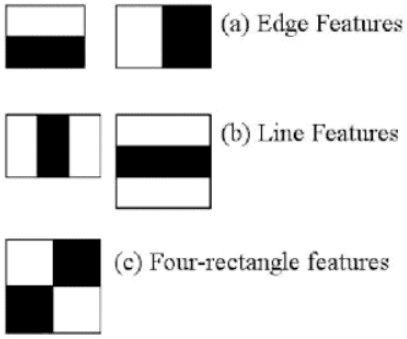

Figure 2: Features of Haar Cascades

2.5.2 DILB

Dlib is a C++ toolbox with machine learning techniques that can be used to address practical issues. Despite being built in C++, it can be executed in Python thanks to python bindings. In addition, a real-time gaze tracking system can be created using its effective facial landmark keypoint analyzer. The pre-trained model is used to initialize the face landmark detector. Since the algorithm will predict continuous values, ensemble regression trees are the foundation of the model. The iBUG-300 W dataset, which includes images and their matching 70 facial landmark points, served as the basis for training this model. The nose, eyes, mouth, and the border of a face are often these landmarks.

2.5.3 Single Shot-Multibox Detector (SSD)

In order to provide a set of default boxes with various aspect ratios and scales for each feature map location, SSD breaks down the output space of the box boundaries. The network creates scores at predicting time for the existence of each object type in every default box and modifies the box

to better fit the contours of the object. In order to naturally manage objects of diverse sizes, the network also incorporates predictions from many feature maps that have various resolutions. Due to the fact that SSD fully removes proposal generation and the following pixel or feature resampling phases and incorporates all computing in a single network, it is simpler than approaches that call for object proposals. As a result, SSD is simple to train and simple to incorporate into systems that call for a detection component. SSD has competitive performance to approaches that include an extra object proposal phase, is significantly faster, and offers an integrated platform for both training and inference, according to results from experiments on the PASCAL VOC and ILSVRC datasets. This model's usage of multi-scale convolutional boundary box outputs coupled to numerous feature mappings at the network's top is a significant component. We can effectively model the space with prospective box shapes using this approach.

2.5.4 MultiTask Cascaded Convolutional Neural Network (MTCNN)

MTCNN is a cutting-edge face detection technique that uses a 3-stage neural network analyzer. To identify faces of various sizes, the image is first scaled many times. The P-network (Proposal), which performs first detection, scans the images after that. Even following NMS (Non-Maximum Suppression), it maintains a low threshold for identification and so frequently identifies false positives. However, this is intended. The second system, the R-network (Refine), receives the recommended regions—which have a high proportion of false positives—as input. This network refines detections—also using NMS—to produce extremely accurate bounding boxes. The O-network (Output) executes the last tweak of the bounding boxes during the last stage. In this manner, not just are faces identified but also highly accurate and precise bounding boxes. Identifying facial landmarks, such as the corners of the mouth, the nose, and the eyes, is an optional function of MTCNN. It manages 13 FPS on full HD videos and, with a few tweaks, potentially up to 45 FPS when rescaled. Using and installing it is also quite simple. Real-time processing is definitely feasible with MTCNN as I've also managed 6–8 FPS on the CPU for full HD. MTCNN is very reliable and accurate. Even with varying sizes, lighting conditions, and forceful rotations, it correctly recognises faces. With GPU, it is just slightly slower than the Viola-Jones detector. Given that CNNs obtain RGB images as input, color data is also used.

2.5.5 System requirement for comparison tests

- OS – Windows
- Processor- intel evo CPU
- RAM -16GB

- SPEED – 3.20GHZ
- For GPU AND CPU Google colab environment

2.6 System Performance Evaluation

2.6.1 Evalution of click action of mouse

Dwell is used for the evalution of the click action of the mouse. Dwell time, which detects how long a user's gaze stays on a certain button or region of the screen until a click is detected, is one popular technique. Longer dwell periods typically indicate more deliberate clicks, whereas lower dwell times could indicate inadvertent or unintentional clicks.

2.6.2 Evalution of blink detection

A method called blink detection scans for patterns in the individual's eye movements, particularly as they blink. For clicking a button one time, or to double-click, a user only needs to blink their eyes once. The precision of click detection can be increased through blink detection.

2.6.3 Average Precision

The term Average Precision has evolved with time. The performance of a model is condensed into one metric using the area under the curve. When assessing the effectiveness of various models, it is crucial. Occasionally, a model with a high AUC **(**Area under the ROC Curve) may perform worse in a particular area than a model with a lower AUC. But in actual use, the AUC works well as a broad indicator of prediction accuracy.

2.6.4 Procedure and feedback

Five different participants are involved in this test procedure (3 female and 2male). All participants need to do both test with glasses without glasses for click performance. They are asked to move the cursor and do click operation with the eye. Eye tracking with dwell time for click is evalutated with time and errors .

The feedbacks links are sent to the participant to collect the feedback . In the feedback, there are two parts. One is for test participant information like name , age, using glasses or not, do they have knowledge of computer. Second, after performing the eye tracking, each test participant was asked to review the system for the overall experience, and how stressful for eye’s while doing eye tracking.

3 Analysis of Mediapipe Eye Mouse

3.1 Overview

A computer vision algorithm-based solution is used in this study. The idea of developing a low cost, real-time eye gaze monitoring system has been addressed by Nasor [NRZ+18], Vasisht [VJG19], and Roy and Chanda [RoC22]. The current study attempted to employ MediaPipe to control a mouse using eyes, which will be useful for those who are unable to move their hands due to illness, accident, or disability.

Eye gaze tracking has several uses, such as in human computer interaction (HCI) [YcK14; TPP19], appliance control [HLJ+14; JTP18; MSJ19], usability testing [KDK+07; JSS+11; ASR19] and advertising efficacy [Bre11; PND16]. The quality of the image and the lighting affect how accurately features are extracted. Algorithm performance drops down in poor lighting environment. Computer vision techniques are used for detecting features; however, they fail in low lighting. When detections are accurate, point of gaze estimation (PoG) can be determined with accuracy. Video quality needs to be improved in order to improve the projection result. A better resolution quality would improve the accuracy of computer vision algorithms. Advanced pre-processing methods (such as thresholding, RGB to gray-scale image) should be used to adjust for variations in lighting, and the resolution of the web camera should also be enhanced to improve eye tracking and make the mouse cursor more sensitive while hovering over the eyes. It will be beneficial to introduce the idea of gaze estimation along with gaze projection because it will significantly enhance gaze projections. The concept of gaze estimation offers to predict gaze projections by learning from usage statistics. In the current study, MediaPipe [LJH+19], a framework for creating perception pipelines in Python, was used to create an eye movement-controlled mouse system while keeping the above aspects in consideration. This chapter describes the concepts and steps taken in implementing the eye mouse control system, containing detailed description. It also shows the results obtained from each step leading to the final result.

3.2 Experimental setup

The proposed model was implemented in Python 3.11.0 with NumPy, OpenCV, TensorFlow, and win32api.

3.2.1 Input from the eye

Eye movements are now being used in real time. The input from the eye moves much more quickly than other forms of input. The user typically considers the location to which they want to

move before using any mechanical pointing device. As a result, the user's goal can be determined from their eye movement before activating any other input device. In fact, the eye is much faster than a fast cursor-positioning device. However, it might be challenging to identify the whole circular shape of the pupil due to the upper and lower eyelids, among other factors [WCD13].

3.2.2 Video Capture

This is the input taken from the camera. The input represents the current frame taken from the video stream waiting to be processed using later steps. To capture video, one must create a 'VideoCapture' object. The index 0 refers to the default camera (built-in webcam).

Challenges of Webcam

The implemented VideoCapture() function should contain parameter 0 if the camera is built into or incorporated into a laptop. The parameter passed must be 1, else the video would not be detected, if the webcam is an external device [PJK+22].

3.2.3 Face and eyes detection

A real-time facial geometry system called Face Mesh by MediaPipe [GAK20] computes 478 3D face landmarks on mobile devices and laptops/PCs. The usage of a separate depth sensor is not necessary because machine learning (ML) is used to estimate 3D surface geometry from data from a single camera. The method combines pipeline-wide GPU acceleration with lightweight model design to achieve real-time performance for live experience. Using the OpenCV-python and MediaPipe libraries in Python 3.9, face bounding box detection and facial landmark point detection is possible, with a detection confidence of 91%. These landmark points will be employed in next steps of data collecting and data preparation. Due to the fact that all HCI applications dependent on the eye gaze output will be in real-time, having a stable and reliable facial landmark detector is essential [RoC22].

3.2.4 Landmark detection

There are 468 landmarks in the Face Mesh, each one must be looped through. For conversion purposes, multipulication of the width by x and the height by y is done. The resulting values are pixel coordinates i.e. (x,y) coordinates of each landmark. About 468 normalized landmarks provide by mediapipe.

A MediaPipe graph segments the portrait and detects face landmarks. Applying the tasks to two separate subsets of frames is one way to lessen the computing load required to conduct both

tasks simultaneously. Using a demultiplexing node in MediaPipe, which divides the input stream's packets into interleaving subsets of packets and outputs each subset as a distinct stream, makes this simple to accomplish. The landmarks and segmentation masks are temporally interpolated across frames to generate the identified landmarks and segmentation masks on all frames. All incoming frames' timestamps are used as the target timestamps for interpolation [LTN+19b;LTN19a]. TensorFlow Lite employed (TensorFlow) with GPU backend coupled with MediaPipe [LTN19a], a framework for building perception pipelines.

3.2.5 Euclidean distance

The Euclidean distance is the distance between two points in coordinate geometry. It is necessary to measure the length of a segment that connects two points in order to locate them on a plane. The Euclidean distance formula is derived using the Pythagoras theorem (Cohen, 2004).

The Euclidean distance formula derived as:

$$d(x, y) = \sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2} \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

where, (x_1, y_1) are the coordinates of one point; (x_2, y_2) are the coordinates of the other point and d is the distance between (x_1, y_1) and (x_2, y_2).

3.2.6 Mouse clicks control

It is simple to implement click control because it simply requires the detection of eye blinks. The left mouse click is triggered by the left eye blink and the right mouse click by the right eye blink. By measuring the distance between two landmark points—one from the top of the eye and another from the bottom—the eye blink can be recognized. Because it only needs to keep track of the distance between two distinct landmarks at a given moment in time, the system can correctly distinguish between open eye state and close eye state [RoC22].

The win32api[9] module offers a variety of libraries and objects used to interact with the Windows system's Application Programming Interface (API). The win32api module in Python is made available by the PyWin32 library, which is already a part of the Python extension (PyWin32). The mouse cursor can be located or moved using PyWin32.

[9] See for details, https://www.delftstack.com/howto/python/python-win32api/

First, the SetCursorPos((x, y)) moves the cursor to the position of (x, y). Then, the x and y positions where the mouse event will be executed for the parameters is entered. The mouse cursor consequently changes to the location (x, y).

3.2.7 Eye blinking and eye extractors

Using landmark, horizontal and vertical lines are drawn for right indices and left indices. The eye blink ratio is obtained based on Euclidean distance for left eye, right eye, top, and bottom. Then, left and right eye distance is determined. For detecting the eyeballs in the image, a mask is created that shows only the eyes. By using a mask, only the part of the image is extracted that is, the eye, ignoring the rest.

Focusing on the eyes shows that when they are open, the width and height of the eye landmarks are at their maximum, but when they are closed, there is no change in the width of the eye, but there are changes occurring in the height of the eye, and they will be targeted in order to detect an eye blink. The Euclidean distance is calculated using equation (1) rather than height and breadth instead. One reason for the Euclidean distance is that when heads are rotated, the width and height of the eyes change proportionally. To avoid this, the distance between two points must be known.

3.2.8 Converting color image to scale image

OpenCV offers more than 150 different color-space conversion techniques. However, two of the most popular are BGR Gray and BGR HSV. The function cv.cvtColor (input image, flag), where the flag specifies the type of conversion, is used to convert colors.

3.2.9 Create a mask

Using the OpenCV function fillPoly(), a filled polygon is created in this code. The polygon's ends and an image are inputted into the function.

Sometimes it is necessary to manipulate the provided image or extract specific portions of the provided image. In these situations, bitwise operators in OpenCV are used. When the elements of the arrays corresponding to the provided two images need to be combined bit-wise, an operator is used in OpenCV called the but wise and operator. This operator allows the arrays corresponding to the two images to be combined, merging them bit-wise.

Parameters

source1_array is the array corresponding to the first input image on which bitwise and operation is to be performed,

source2_array is the array corresponding to the second input image on which bitwise and operation is to be performed,

destination_array is the resulting array by performing a bitwise operation on the array corresponding to the first input image and the array corresponding to the second input image and

mask is the mask operation to be performed on the resulting image, and it is optional.

By using the bitwise operator and providing the two input images as parameters, the merged image is returned and shown as the output image on the screen as ,shown in Figure 3.

3.2.10 Eyes portion Estimator

To get the height and widththe of image, and then extract the exact position of the eye. Here, image-smoothing techniques, such as Gaussian blur, Median blur in OpenCV, are used to remove noise from the image.

3.2.10.1 Gaussian blur

A Gaussian blur, sometimes referred to as Gaussian smoothing, is the outcome of blurring an image with a Gaussian function in image processing. The image is convolved with a Gaussian filter rather than a box filter in a gaussian blur process. A low-pass filter called the Gaussian filter removes high-frequency components [MNA08; FWW10]. The gaussian blur is a type of image processing that uses a filter on an image. This filter returns a single number that is computed using a weighted average based on the normal distribution from the surrounding pixels, the number of which is defined by the filter's size. An easy and widely used image processing technique is smoothing, often known as blurring [Sze22]. In order to create the output array for gaussian filtering, each point in the input array is convolved with a gaussian kernel before being added together.

2D Gaussian can be represented as:

$$G_0(x,y) = A_e \frac{-(x-\mu_x)^2}{2\sigma_x^2} + \frac{-(y-\mu_y)^2}{2\sigma_y^2} \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

where μ is the mean (the peak) and σ^2represents the variance (per each of the variables x and y).

For instance, to create the blurred image, width 9 and height of 9 is used.

In order to reduce picture clarity, make images stand out, remove noise from the images, or reduce details from the visuals, gaussian blurring[10] uses a function named Gaussian Blur(). The function applies the chosen Gaussian kernel to the source image. Filtering in-place is supported.

Parameters

Src input image - the image may include any number of independent channels that are processed, but the depth must be one of the following: CV 8U, CV 16U, CV 16S, CV 32F, or CV 64F.

dst - the same size and type of the output image as the source.

Ksize - size of the Gaussian kernel. ksize.width and ksize.height Although heights might vary, they must both be odd and positive. Or, they may be zeros, in which case sigma would be used to compute them.

sigmaX - Gaussian kernel X-axis standard deviation

sigmaY - Standard deviation of the Gaussian kernel in the Y direction; if sigmaY is zero, it is set to be equal to sigmaX; if both sigmas are zeros, they are computed from, respectively, ksize.width and ksize.height (see getGaussianKernel for details); to fully control the result regardless of potential future modifications to all of this semantics, it is advised to specify all of ksize, sigmaX, and sigmaY.

borderType - method of pixel extrapolation.

3.2.10.2 Median blur

The median blur operation resembles other averaging techniques. Here, the kernel area's median of all the pixels takes the place of the image's central component. This procedure smoothens the edges while minimizing noise. Since median smoothing can sometimes preserve edges while reducing noise, it is frequently employed in edge detection methods. The median filter with the ksize×ksize aperture is used to smooth an image by the function. A multi-channel image is processed independently for each channel. It is possible to operate in place.

Parameters

Src input 1-, 3-, or 4-channel image; when ksize is 3 or 5, the image depth should be CV_8U, CV_16U, or CV_32F, for larger aperture sizes, it can only be CV_8U.

[10] Gaussian blur details see, http://www.ruanyifeng.com/blog/2012/11/gaussian_blur.html

Dst same-sized destination array and type as src.

ksize[11] linear aperture size; it must be odd and more than 1, for instance: 3, 5, 7...

The median filter passes over each component of the signal (in this example, the image) and swaps out each pixel with the median of its surrounding pixels (located in a square neighborhood around the evaluated pixel).

The median filter is significantly superior to gaussian blur at removing noise while maintaining edges for small to moderate levels of gaussian noise for a specific, set window size [ArD09]. While it performs similarly to Gaussian blur for high noise levels, it performs particularly well for speckle noise and salt-and-pepper noise (impulsive noise) [Arc05].

3.2.10.3 Thresholding

The assignment of pixel values in relation to the supplied threshold value is known as thresholding [OtN79;SeS04]. This is a technique used in OpenCV. Each pixel's value is compared to the threshold value during thresholding. The pixel value is set to 0 if it is less than the threshold; otherwise, it is set to the maximum value (generally 255). Thresholding is a segmentation method that is frequently used to distinguish between foreground and background objects. A threshold is a value that has the areas below the threshold and above the threshold on either side of it. This thresholding method is used for grayscale images in computer vision. Therefore, the image must first be transformed to a grayscale colour space.

Binary thresholding

The simple thresholding method is binary thresholding. The same threshold value is used for each pixel. The pixel value is set to 0 if it is below the threshold; otherwise, it is set to the maximum value.

$$dst = \begin{cases} maxval & if src(x,y) > thresh \\ 0, & otherwise \end{cases} \quad \text{……..(3)}$$

The simple thresholding technique in Python is that if pixel intensity is greater than the set threshold, value set to 255, else set to 0 (black).

In our case, ret, threshold = cv.thresh (med_bl, 140, 355, cv.THRESHOLD_BINARY)

[11]See https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html#ga564869aa33e58769b4469101aac458f9

The source image, which must be a grayscale image, is the first argument. The threshold value, which is used to categorise the pixel values, is the second input. The maximum value that is assigned to pixel values that are higher than the threshold is the third argument.

The next step in this section is to divide the eyes into the right, center, and left pieces. Then, the left eye, right eye, center, and closed eye are determined by counting the black pixels in each area and using the index of the list's maximum values.

3.2.10.4 Pixel counter function

This function is used to count pixels of the right eye, left eye, center, and closed eye.

Then the time of each frame is set from the video loop. These frames are counted for eye position continuously while the camera capturing the frame from the video.

3.2.10.5 Resizing frame

First, the frame counter is started, which continuously counting the frames as soon as video capturing is started.

cv2.flip() method is used to flip a 2D array. The function cv::flip flips a 2D array around vertical, horizontal, or both axes.

A flag that specifies the axis of the array to rotate; a value of 0 rotates the array around the x-axis, while a positive value (such as 1) rotates it around the y-axis. A negative value (such as 1) results in the two axes being flipped.

BGR format

The image file is read using the OpenCV method imread(), and the colors are in the BGR order (blue, green, red). Thus, the BGR image format is used by OpenCV. As a result, by default, cv2.imread() analyses images while reading them in BGR format. In this case, cv2.COLOR RGB2BGR is used.

Using the cvtColor() method, a BGR image can be converted into RGB (red, green, blue) and vice versa.

There are several reasons to convert an image from BGR to RGB and vice versa, but one of them is that different image processing libraries have different pixel orderings.

3.2.11 Frame per second

The frequency (rate) at which consecutive images (frames) are recorded or displayed is called frame rate and is expressed in frames per second (FPS). The term is equally applicable to computer graphics, motion capture technologies, and film and video cameras. The frame frequency, commonly known as frame rate, is measured in hertz. In electronic camera specifications, the term "frame rate" may refer to the highest possible rate; however, in reality, various parameters (such expas osure time) may cause the frequency to be lower [WSe21].

Graphs can run OpenGL in many GL contexts due to MediaPipe. This is particularly helpful in graphs that combine slower GPU inference paths (like those at 10 FPS) with faster GPU rendering paths (such as at 30 FPS) because using the same GL context for both operations will increase the rendering frame rate [LJH+19]. The frame per second (FPS) value is calculated by dividing the frame counter by the end time. The frame rate obtained between 5.1 fps to 7.3 fps (see, Figure 4). These values are much lower than the current video standard frame speed of 30fps.

3.2.12 Accelerating TensorFlow Lite with XNNPACK

With the use of the new XNNPACK[12] delegate, TensorFlow Lite[13] can now provide excellent x86 performance, sometimes outperforming Intel's OpenVino package. The main disadvantage of XNNPACK is that it can only be used for floating-point calculations. 8-bit model quantization can easily lead to a >2x speed boost, with an even larger increase when used with the new Intel Cascade Lake CPUs that offer AVX-512 VNNI instructions.

3.3 Results of eye-mouse

When the program executes, a window will open where it will detect the face and eye. It will then crop the eye in a three-stage mask, draw the eyes, and open the frame. (See Figure 3 and Figure 4)

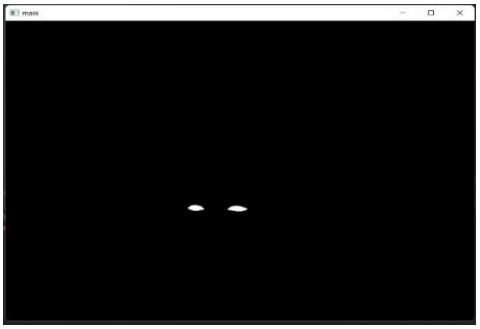

Figure 3: Mask

[12] XNNPACK available at: https://www.private-ai.com/2020/06/12/accelerating-tensorflow-lite-with-xnnpack/
[13] TensorFlow Lite available at: https://tensorflow.google.cn/lite/guide/build_cmake_pip?hl=zh-cn

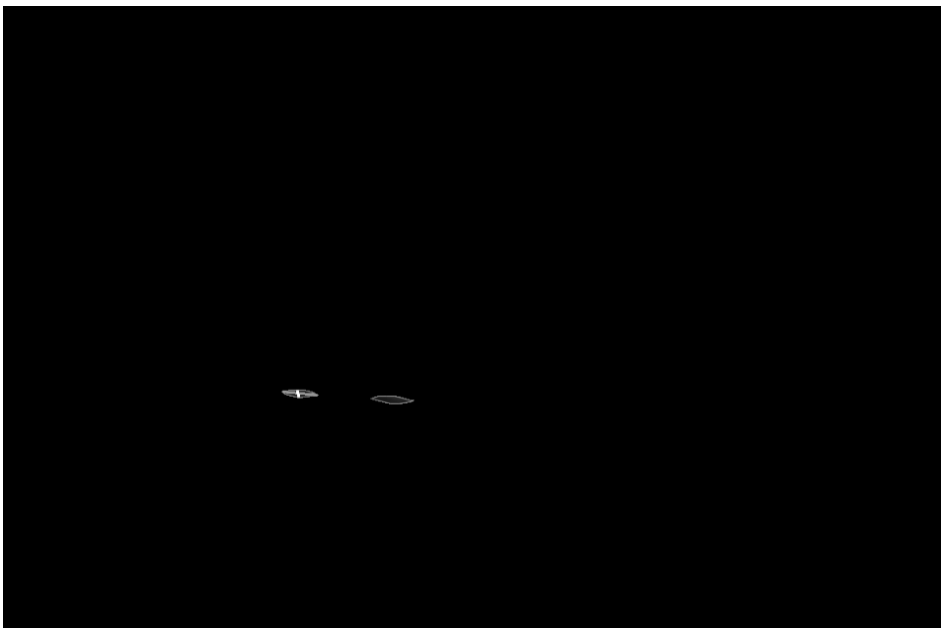

Figure 4: Detected eyes

It will calculate the blink ratio, determining the eye's direction of movement (left, right, or center), and displaying the ratio in the frame as shown in Figure 5. Then mouse cursor will start to move according to eye movement, mouse clicks occur when user eye blinks. When the cursor movesa on particular icon or menu othe n screen and the eye blinks, thenthe application is opened.

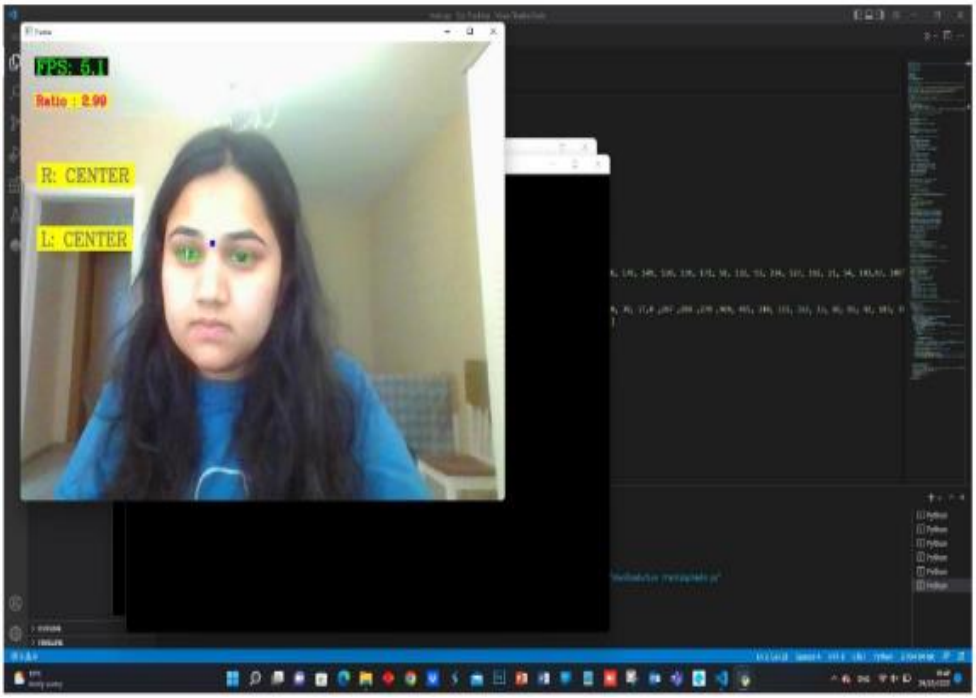

Figure 5: Output of frame per second and eye center movement

4 Results and Discussion

4.1 Comparison Results

The proposed MediaPipe solution was compared with Dlib, Haar cascade, SSD, and MTCNN solutions.

4.1.1 Performace Comparison of different face detector models

The speed results are presented in Table 1 and Figure 6. CPU and GPU speeds are shown. It is seen that MediaPipe solution speeds of GPU and CPU are the highest, especially the GPU speed, followed by the rest.

Table 1: CPU and GPU speeds(Frame per second) of the various models solutions

Model	**Speed in GPU(fps)**	**Speed in CPU (fps)**
Dlib	-	18.85
Haar cascade	-	38.15
SSD	18.8	16.32
MTCNN	3.12	1.91
MediaPipe	**343.63**	**235.34**

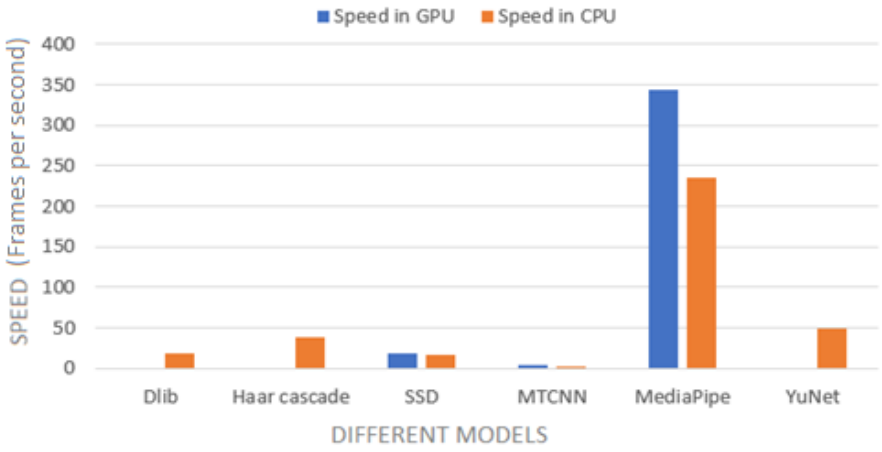

Figure 6: Model comparison in terms of frames/second

4.1.2 Average precision results

The average precision results are presented in Table 2 and Figure 7. It is seen that though the competitors outperformed MediaPipe, the margin is not much.

Table 2: Comparison of average precision of the different models

Model	**Average precision**
MediaPipe	**0.8**
Single Shot Multibox Detector	0.851
Multi-Task Cascaded CNN	0.815
YuNet	0.915

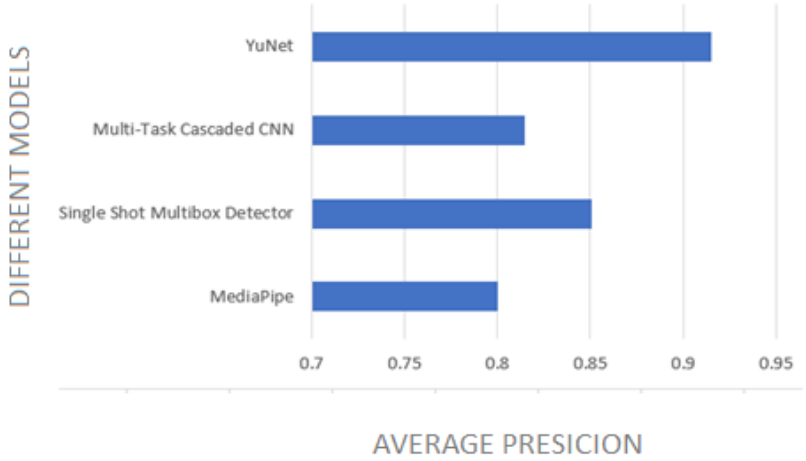

Figure 7: Average precision comparison of different models

Thus, the proposed MediaPipe model is the best model for face and eye landmark detection.

4.1.3 Results of evaluation of dwell time and mouse clicks

Figure 8 indicates the results of the evaluation of dwell time and mouse clicks. It can be seen that results are better without glasses (15.893–20.987s), i.e., mouse click responses are better without glasses.

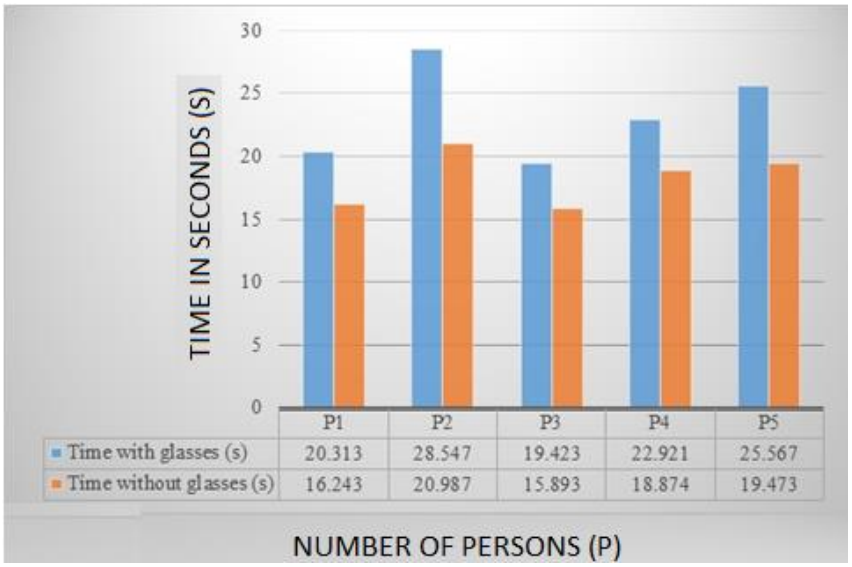

Figure 8: Dwell time with and without glasses

4.1.4 Iris detection results

MediaPipe's iris detection was compared with other popular iris detection approaches (Table X). MediaPipe showed the best results in terms of speed and accuracy. Though its robustness of occlusion is medium, it has good robustness to light and is easy to implement, thus indicating its superiority.

Table 3: Comparison of iris/pupil detection with different models

Method	**Speed (FPS)**	**Accuracy (%)**	**Robustness to light**	**Robustness of occlusion**	**Ease of Implementation**
MediaPipe Iris Detection [VaV20]	94	30	High	Medium	Easy
Swirski Pupil Detection [SwD13]	90	28	High	Low	Medium
Starburst Pupil Detection [Gon09]	88	32	Medium	Medium	Complex
ExCuSe Pupil Detection [TiB11]	92	25	Medium	High	Easy
EISe Pupil Detection [TiB11]	91	27	Medium	Medium	Easy

4.1.5 Survey results

Figure x shows the survey results of the five respondents. It can be seen that the respondents are satisfied with the Eye-mouse experience.

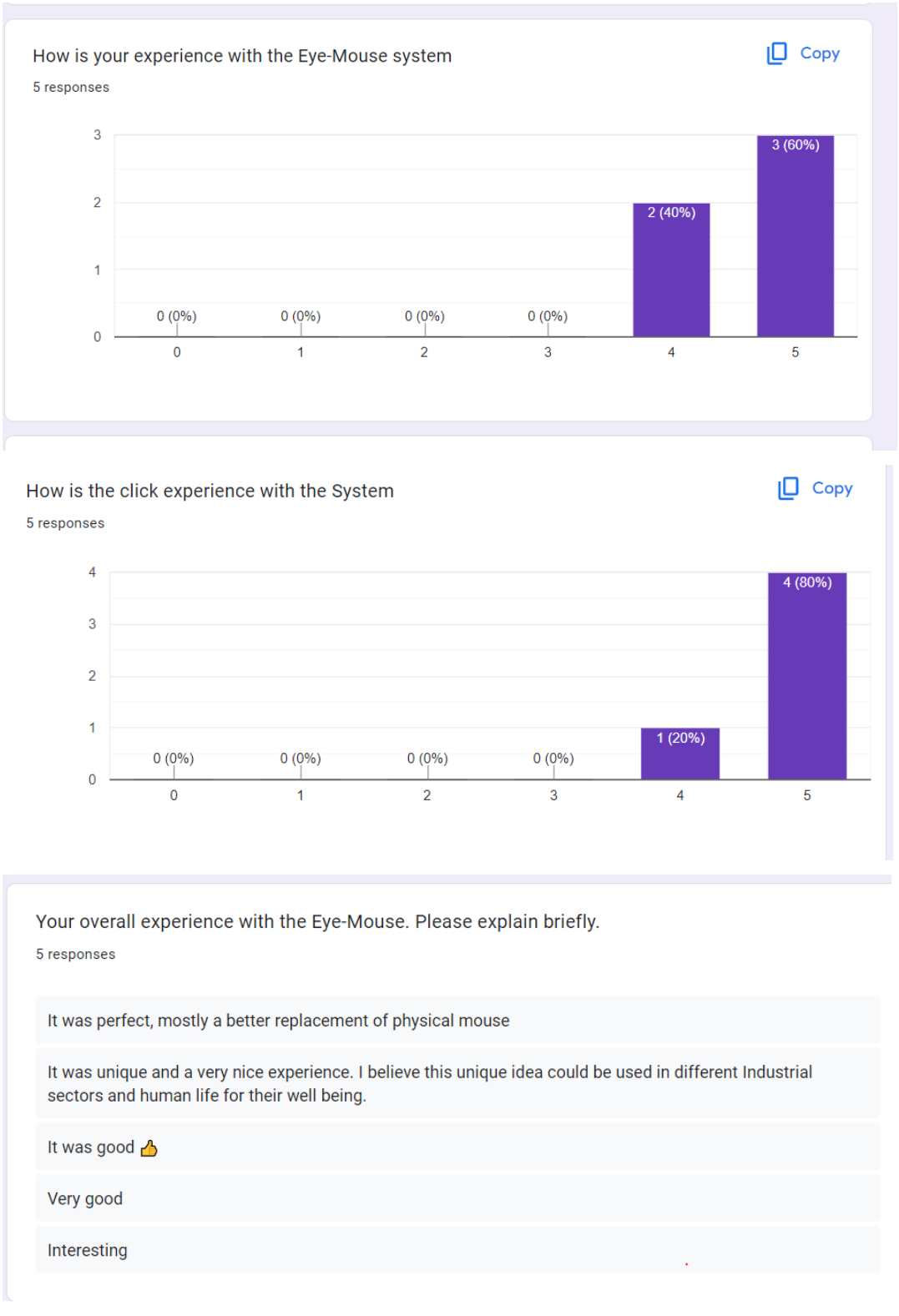

Figure 9: Survey results of experience in using the Eye-mouse solution

4.2 Discussion

An HCI system has been successfully constructed that offers mouse control, mouse click, and scrolling features. The user has a lot of flexibility and a degree of freedom while interacting with the HCI system because of the 9-class output. The system is able to clearly distinguish between an open-eye state and a closed-eye state, making these states useful as triggers for mouse clicks or switching between various operating modes [RoC22]. This system was tested on relatively low-spec hardware, which indicates it can be easily commercialized without requiring any additional resources. Using eye blinks to control mouse actions was suggested by Sanjay [GGP16]. An eye blink and head movement-based humhuman-computerterface is presented. This system converts them into mouse pointer movements on the screen and detects user's eye blinks, which it then converts into mouse click events on the computer screen. The proposed system is successfully implemented, and mouse movement is controlled by eye moveme,nt and mouse clicking is completed by the user's eyes blinking at the appropriate sensitivity level [NAN^{+}22]. In the same way, our current system was successful in detecting and determining the location of the eye's pupil, which allowed it to move the cursor on the screen to the desired gaze zone. This resulted in a system that can detect where a user is looking reliably and accurately even when their head is not moving. When the user's head moves, accuracy decr,eases and incorrect detections may happen. The tracking of the pupil location is relied on doing the detection frequently, which is a significant contributing factor to the low precision. Using predictive filters does not require high calculations; Since it is a real-time system with low delay, it makes a trade-off between accuracy and delay. The current detected position's coordinates were to be stored, and the future detected position's coordinates were to be scaled to fit the dimensions of the screen by calculating the difference between the two. The above method was implemented in the code, but because scaling was not possible, the cursor could only move in a limited area of the screen. Since the capabilities and specifications of the available resources are limited, there was a delay since image processing demands heavy computations at a high camera frame rate. The system must identify both the location of the face and the user's eyes when the user's head is moving, which causes a substantial delay. If the head is stable, however, it only needs to detect the user's eyes, which results in a smaller delay. Because the system can only move in a small area of the screen, it cannot stop the eye mouse as long as a face is recognized; therefore, stopping or existing window will take some time. The user must be close enough to the computer (less than half a meter away from the screen) for the user's iris to be clearly visible in the camera's field of view in order for the system to function properly.

5 Conclusions

This study suggested a solution that will make it easier for these people to interact with others by using a mouse. The findings based on the three objectives are summarized in the following sub-sections.

- Key shortcomings based on the review are summarized as follows:
  Systems for controlling a computer using voice commands and head movements were created. These systems control computers using a multimodal interface developed for people with severe disabilities. The systems, however, fail to function properly when a dark background that should be blank is present. A real-time mouse pointer control for eye gaze evaluation has been created by researchers. Their technology has several constraints, such as requiring the user's head to be at the same altitude as the webcam and a distance between the user's eyes and the webcam of 20-75 cm.Tobii EyeX, a gadget for recording gaze, was created. The Tobii EyeX's application for tracking micro-saccadic eye movements or in-the-moment gaze-contingent stimulus control, on the other hand, is limited by its low sample rate and modest precision.Iris detection and tracking solutions have been created. However, detecting iris landmarks in real-time with high accuracy remains a problem, and it is a computationally expensive process.Eye-controlled computer mouse-based systems have sought to discern the precise locations and gaze orientations of the eyes. However, the EOG signal has some limitations, including a fixed head and poor resolution.Thus, the main issues are concerned with locational restrictions, accuracy, and cost.
- The system that allows the disabled to communicate through the computer by eye movements was designed using Google MediaPipe.In the absence of spectacles, this system can detect iris and pupil better than existing methods.The system does not require expensive cameras, so overall costs will be lesser. Mediapipe, which has so far not been used for eye movement detection, is a very useful tool for such applications. With the use of spectacles, the system response becomes slower.Average precision of the solution is lesser than other compared methods.
- The system was implemented as detailed in section 3.2 and was compared with the existing solutions for mouse control. It was seen that the solution outperformed other solutions in terms of GPU/CPU speed (343.63 Mbps & 235.34 Mbps) and iris/pupil detection. However, average precision was lesser than the other solutions, though not by a big margin. This needs to be further investigated and improved.

6 Future work

The following may be considered for future work:

- High-resolution photos present a promising avenue for future development in this study. Time-of-flight (ToF) cameras, also referred to as depth cameras, use lasers or light-emitting diodes (LEDs) to estimate the distance between the camera and the image's source. Robots in automated manufacturing employ ToF technologies for picking up objects, but it has numerous other applications as well. It is commonly used in tandem with computer algorithms to isolate the subject in an image and blur the background.
- In order to boost performance, a number of obstacles must be removed. These include interference from head and muscle motion, signal drifting, and channel cross talking. A small amount of head movement is always there, regardless of whether the user is actively engaged in a task or not. The effects of head movement are minimal in tasks that require only a low level of detail, such as wheel-chair driving or casually perusing a computer screen. On the other hand, head tracking can be implemented in different contexts to account for the user's natural head movement.
- Future research in this technology may enable paralyzed people to perform daily tasks including operating household appliances, engaging in video games, and interacting with interactive toys. Studies can be done to expand the employment options available to the disabled, improving their quality of life.

References

[AA20] Abiyev, R. H., & Arslan, M. (2020). Head mouse control system for people with disabilities. Expert Systems, 37(1), e12398.Ablavatski, A., Vakunov, A., Grishchenko, I., Raveendran, K., & Zhdanovich, M. (2020). Real-time Pupil Tracking from Monocular Video for Digital Puppetry. arXiv preprint arXiv:2006.11341.

[ABA19] Arslan, M., Bush, I.J., Abiyev, R.H. (2019). Head Movement Mouse Control Using Convolutional Neural Network for People with Disabilities. In: Aliev, R., Kacprzyk, J., Pedrycz, W., Jamshidi, M., Sadikoglu, F. (eds) 13th *International Conference on Theory and Application of Fuzzy Systems and Soft Computing — ICAFS-2018. ICAFS 2018. Advances in Intelligent Systems and Computing*, vol 896. Springer, Cham. https://doi.org/10.1007/978-3-030-04164-9_33

[ACR22] Agrawal, A. S., Chakraborty, A., & Rajalakshmi, M. (2022). Real-Time Hand Gesture Recognition System Using MediaPipe and LSTM. *International Journal of Research Publication and Reviews.*3(04), pp 2509-2515. Available at: www. ijrpr.com ISSN, 2582, 7421

[AM11] Arai, K., & Mardiyanto, R. (2011, April). Eye-based HCI with full specification of mouse and keyboard using pupil knowledge in the gaze estimation. In 2011 Eighth International Conference on Information Technology: New Generations. pp. 423-428. IEEE.

[Ane22] Anecca, A. (2022). The gaps in counting India's disabled population. Scroll.in. https://scroll.in/article/1028665/the-gaps-in-counting-indias-disabled-population#:~:text=If%20data%20from%20the%20latest,Sample%20Survey%20Report%20in%202018.

[AOO11] Abe, K., Ohi, S., & Ohyama, M. (2011, July). Eye-gaze detection by image analysis under natural light. In International Conference on Human-Computer Interaction (pp. 176-184). Springer, Berlin, Heidelberg.

[Arc05] Arce, G. R. (2005). Nonlinear signal processing: a statistical approach. John Wiley & Sons.

[AST+22] Adnan, M., Sardaraz, M., Tahir, M., Dar, M. N., Alduailij, M., & Alduailij, M. (2022). A Robust Framework for Real-Time Iris Landmarks Detection Using Deep Learning. Applied Sciences, 12(11), 5700.

[ArD09] Arias-Castro, E., & Donoho, D. L. (2009). Does median filtering truly preserve edges better than linear filtering?. The Annals of Statistics, 37(3), 1172-1206.

[ASR19] Aviz, I. L., Souza, K. E., Ribeiro, E., de Mello Junior, H., & Seruffo, M. C. D. R. (2019, October). Comparative study of user experience evaluation techniques based on mouse and gaze tracking. In *Proceedings of the 25th Brazillian symposium on multimedia and the Web*. pp. 53-56.

[AT10] Agarkar, P. M., & Talbar, S. N. (2010, February). Algorithm for iris code organization and searching for iris recognition system. In Proceedings of the International Conference and Workshop on Emerging Trends in Technology (pp. 544-547).

[AVG+20] Ablavatski, A., Vakunov, A., Grishchenko, I., Raveendran, K., & Zhdanovich, M. (2020). Real-time Pupil Tracking from Monocular Video for Digital Puppetry. arXiv preprint arXiv:2006.11341.

[BFA13] Babiker, A., Faye, I., & Malik, A. (2013, October). Pupillary behavior in positive and negative emotions. In 2013 IEEE International Conference on Signal and Image Processing Applications (pp. 379-383). IEEE.

[BGA08] Barreto, A., Gao, Y., & Adjouadi, M. (2008, October). Pupil diameter measurements: Untapped potential to enhance computer interaction for eye tracker users? In Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility (pp. 269-270).

[Bis21] Biswas, P. (2021). Appearance-based gaze estimation using attention and difference mechanism. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 3143-3152).

[BKV+19] Bazarevsky, V., Kartynnik, Y., Vakunov, A., Raveendran, K., & Grundmann, M. (2019). Blazeface: Sub-millisecond neural face detection on mobile gpus. arXiv preprint arXiv:1907.05047.

[Bmr22] Bharath, M. R. R. (2022). Controlling Mouse and Virtual Keyboard using Eye-Tracking by Computer Vision. *JOURNAL OF ALGEBRAIC STATISTICS*, 13(3), 3354-3368.

[Bre11] Bremmers, L. (2011). What behavior predicts banner effectiveness?: a mouse and eye tracking study.

[CB05] Chau, M., & Betke, M. (2005). Real time eye tracking and blink detection with usb cameras. Boston University Computer Science Department.

[CBH15] Caraiman, S., Stan, A., Botezatu, N., Herghelegiu, P., Lupu, R. G., & Moldoveanu, A. (2015, May). Architectural design of a real-time augmented feedback system for neuromotor rehabilitation. In *2015 20th International Conference on Control Systems and Computer Science* (pp. 850-855). IEEE.

[CD004] Cohen, David (2004), Precalculus: A Problems-Oriented Approach (6th ed.), Cengage Learning, p. 698, ISBN 978-0-534-40212-9

[CXS17] Chaudhary, U., Xia, B., Silvoni, S., Cohen, L. G., & Birbaumer, N. (2017). Brain–Computer Interface–Based Communication in the Completely Locked-In State. PLOS Biology, 15(1), e1002593. doi:10.1371/journal.pbio.1002593 Chaudhary, U., Xia, B., Silvoni, S., Cohen, L. G., & Birbaumer, N. (2017). Brain–Computer Interface–Based Communication in the Completely Locked-In State. PLOS Biology, 15(1), e1002593. doi:10.1371/journal.pbio.1002593

[CZ18] Cheng, Y., Lu, F., & Zhang, X. (2018). Appearance-based gaze estimation via evaluation-guided asymmetric regression. In Proceedings of the European Conference on Computer Vision (ECCV). pp. 100-115.

[DC19] De Lucena, S. E., & Conzelmann, P. (2019, August). Computer USB-Mouse Emulation Using EOG. In 2019 4th International Symposium on Instrumentation Systems, Circuits and Transducers (INSCIT). pp. 1-5. IEEE. https://doi.org/10.1109/inscit.2019.8868754

[DLS11] De Santi, L., Lanzafame, P., Spano, B., D'Aleo, G., Bramanti, A., Bramanti, P., & Marino, S. (2011). Pursuit ocular movements in multiple sclerosis: a video-based eye-tracking study. Neurological Sciences, 32(1), 67-71.

[EMM19] Elliott, M. A., Malvar, H., Maassel, L. L., Campbell, J., Kulkarni, H., Spiridonova, I., ... & Scanlan, J. M. (2019). Eye-controlled, power wheelchair performs well for ALS patients. Muscle & nerve, 60(5), 513-519.

[Eva11] Evans, D. (2011). The internet of things. how the next evolution of the internet is changing everything, whitepaper. Cisco Internet Business Solutions Group (IBSG).

[FCD18] Fischer, T., Chang, H. J., & Demiris, Y. (2018). Rt-gene: Real-time eye gaze estimation in natural environments. In Proceedings of the European conference on computer vision (ECCV). pp. 334-352.

[FUZ16] Fatima, R., Usmani, A., & Zaheer, Z. (2016, March). Eye movement based human computer interaction. In 2016 3rd International Conference on Recent Advances in Information Technology (RAIT). pp. 489-494. IEEE.

[FWT+17] Feit, A. M., Williams, S., Toledo, A., Paradiso, A., Kulkarni, H., Kane, S., & Morris, M. R. (2017, May). Toward everyday gaze input: Accuracy and precision of eye tracking and implications for design. In *Proceedings of the 2017 Chi conference on human factors in computing systems.* pp. 1118-1130.

[GBL+03] Grauman, K., Betke, M., Lombardi, J., Gips, J., & Bradski, G. R. (2003). Communication via eye blinks and eyebrow raises: Video-based human-computer interfaces. Universal Access in the Information Society, 2(4), pp. 359-373, 2003.He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770-778.

[GCS+13] Ghani, M. U., Chaudhry, S., Sohail, M., & Geelani, M. N. (2013, December). GazePointer: A real time mouse pointer control implementation based on eye gaze tracking. In INMIC. pp. 154-159. IEEE. https://doi.org/10.1109/inmic.2013.6731342

[GGP16] Gantyala, S., Godad, W., & Phadnis, N. (2016). Controlling mouse events using eye blink. *International Journal of Advanced Research in Computer and Communication Engineering*, 5(3), 12.

[GAK20] Grishchenko, I., Ablavatski, A., Kartynnik, Y., Raveendran, K., & Grundmann, M. (2020). Attention mesh: High-fidelity face mesh prediction in real-time. arXiv preprint arXiv:2006.10962

[GVB+16] Gibaldi, A., Vanegas, M., Bex, P. J., & Maiello, G. (2016). Evaluation of the Tobii EyeX Eye tracking controller and Matlab toolkit for research. Behavior Research Methods, 49(3), pp. 923–946. https://doi.org/10.3758/s13428-016-0762-9

[HB+22] Heimerl, A., Becker, L., Schiller, D., Baur, T., Wildgrube, F., Rohleder, N., & Andre, E. (2022, June). We've never been eye to eye: A Pupillometry Pipeline for the Detection of Stress and Negative Affect in Remote Working Scenarios. In Proceedings of the 15th International Conference on PErvasive Technologies Related to Assistive Environments (pp. 486-493).

[HEM+20] Hosp, B., Eivazi, S., Maurer, M., Fuhl, W., Geisler, D., & Kasneci, E. (2020). Remoteeye: An open-source high-speed remote eye tracker. *Behavior research methods*, *52*(3), 1387-1401.

[HGT11] Hervet, G., Guérard, K., Tremblay, S., & Chtourou, M. S. (2011). Is banner blindness genuine? Eye tracking internet text advertising. Applied cognitive psychology, 25(5), 708-716.

[HLJ+14] Heo, H., Lee, J. M., Jung, D., Lee, J. W., & Park, K. R. (2014). Nonwearable gaze tracking system for controlling home appliance. *The Scientific World Journal*, 2014.

[HLV+17] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700-4708.

[HN16] Henzen, A., & Nohama, P. (2016). Adaptable virtual keyboard and mouse for people with special needs. 2016 Future Technologies Conference (FTC). http://doi.org/10.1109/ftc.2016.7821782

[HZX+21] Huang, J., Zhang, Z., Xie, G., & He, H. (2021). Real-time precise human-computer interaction system based on gaze estimation and tracking. Wireless Communications and Mobile Computing, 2021.

[IHH11] Ismail, A., Hajjar, A.E., & Hajjar, M. (2011). A Prototype System for Controlling a Computer by Head Movements and Voice Commands. ArXiv, abs/1109.1454.

[IRA+22] Islam, M. R., Rahman, R., Ahmed, A., & Jany, R. (2022). NFS: A Hand Gesture Recognition Based Game Using MediaPipe and PyGame. arXiv preprint arXiv:2204.11119.

[IRS21] Islam, R., Rahman, S., & Sarkar, A. (2021, July). Computer Vision Based Eye Gaze Controlled Virtual Keyboard for People with Quadriplegia. In 2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI). pp. 1-6. IEEE.

[JK17] Jian, Y. C., & Ko, H. W. (2017). Influences of text difficulty and reading ability on learning illustrated science texts for children: An eye movement study. *Computers & Education*, 113, pp. 263-279.

[JMP15] Jalaliniya, S., Mardanbegi, D., & Pederson, T. (2015, September). MAGIC pointing for eyewear computers. In Proceedings of the 2015 ACM International Symposium on Wearable Computers. pp. 155-158.

[JTP18] Juhong, A., Treebupachatsakul, T., & Pintavirooj, C. (2018, January). Smart eye-tracking system. In *2018 International Workshop on Advanced Image Technology (IWAIT)*. pp. 1-4. IEEE.

[JSS+11] Johansen, S. A., San Agustin, J., Skovsgaard, H., Hansen, J. P., & Tall, M. (2011). Low cost vs. high-end eye tracking for usability testing. In CHI'11 Extended Abstracts on Human Factors in Computing Systems. pp. 1177-1182.

[KBW08] Kocejko, T., Bujnowski, A., & Wtorek, J. (2008). Eye mouse for disabled. 2008 Conference on Human System Interactions. https://doi.org/10.1109/hsi.2008.4581433

[KDK+07] Kim, B., Dong, Y., Kim, S., & Lee, K. P. (2007, July). Development of integrated analysis system and tool of perception, recognition, and behavior for web usability test: with emphasis on eye-tracking, mouse-tracking, and retrospective think aloud. In International Conference on Usability and Internationalization. pp. 113-121. Springer, Berlin, Heidelberg.

[KHG+21] Kummen, A., Li, G., Hassan, A., Ganeva, T., Lu, Q., Shaw, R., ... & Mohamedally, D. (2021). MotionInput v2. 0 supporting DirectX: A modular library of open-source gesture-based machine learning and computer vision methods for interacting and controlling existing software with a webcam. arXiv preprint arXiv:2108.04357.

[KKK+16] Krafka, K., Khosla, A., Kellnhofer, P., Kannan, H., Bhandarkar, S., Matusik, W., & Torralba, A. (2016). Eye tracking for everyone. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2176-2184).

[KKS19] Khare, V., Krishna, S. G., & Sanisetty, S. K. (2019, March). Cursor control using eye ball movement. In *2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)*. Vol. 1, pp. 232-235. IEEE.

[KKS09] Kumar, N., Kohlbecher, S., & Schneider, E. (2009, October). A novel approach to video-based pupil tracking. In *2009 IEEE International Conference on Systems, Man and Cybernetics.* pp. 1255-1262. IEEE.

[KLH+21] Kummen, A., Li, G., Hassan, A., Ganeva, T., Lu, Q., Shaw, R., ... & Mohamedally, D. (2021). MotionInput v2. 0 supporting DirectX: A modular library of open-source gesture-based machine learning and computer vision methods for interacting and controlling existing software with a webcam. arXiv preprint arXiv:2108.04357.

[KMS17] Kumar, C., Menges, R., & Staab, S. (2017, June). Assessing the usability of gaze-adapted interface against conventional eye-based input emulation. In 2017 IEEE 30th International Symposium on Computer-Based Medical Systems (CBMS). pp. 793-798. IEEE.

[KPW+21] Kastrati, A., Płomecka, M. M. B., Pascual, D., Wolf, L., Gillioz, V., Wattenhofer, R., & Langer, N. (2021). EEGEyeNet: a Simultaneous Electroencephalography and Eye-tracking Dataset and Benchmark for Eye Movement Prediction. arXiv preprint arXiv:2111.05100.

[KUS+13] Kunze, K., Utsumi, Y., Shiga, Y., Kise, K., & Bulling, A. (2013, September). I know what you are reading: recognition of document types using mobile eye tracking. In Proceedings of the 2013 international symposium on wearable computers (pp. 113-116).

[KYY19] Kim, H., Yi, S., & Yoon, S. Y. (2019). Exploring touch feedback display of virtual keyboards for reduced eye movements. Displays, 56, 38-48.

[Lan00] Lankford, C. (2000, November). Effective eye-gaze input into windows. In Proceedings of the 2000 symposium on Eye tracking research & applications (pp. 23-27).

[LBS+22] Lavanya, M. M., Bhargavi, S. V. S., Srikanth, B., Guravaiah, M. K., Deepthi, P. N., & Siva, V. (2022). Eyeball Based Cursor Movement Using Opencv. *European Journal of Molecular & Clinical Medicine*. 9(03). pp. 10748-10755.

[LCI+19] Lee, J., Chirkov, N., Ignasheva, E., Pisarchyk, Y., Shieh, M., Riccardi, F., ... & Grundmann, M. (2019). On-device neural net inference with mobile gpus. arXiv preprint arXiv:1907.01989.

[LCJ+19] Lugaresi, Camillo, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang et al. Mediapipe: A framework for building perception pipelines. arXiv preprint arXiv:1906.08172 (2019).

[LHP13] Lee, J. W., Heo, H., & Park, K. R. (2013). A novel gaze tracking method based on the generation of virtual calibration points. Sensors, 13(8), 10802-10822.

[LJH+19] Lugaresi, Camillo, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang et al. "Mediapipe: A framework for building perception pipelines." arXiv preprint arXiv:1906.08172 (2019).

[LP19] Laddi, A., & Prakash, N. R. (2019). Eye gaze tracking based directional control interface for interactive applications. Multimedia Tools and Applications, 78(22), pp. 31215-31230.

[LRF14] Lopez, A., Rodriguez, I., Ferrero, F. J., Valledor, M., & Campo, J. C. (2014). Low-cost system based on electro-oculography for communication of disabled people. *2014 IEEE 11th International Multi-Conference on Systems, Signals & Devices (SSD14)*. doi:10.1109/ssd.2014.6808755

[LTN19a] Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., ... & Grundmann, M. (2019a). Mediapipe: A framework for building perception pipelines. arXiv preprint arXiv:1906.08172.

[LTN+19b] Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., ... & Grundmann, M. (2019b, June). Mediapipe: A framework for perceiving and processing reality. In Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR) (Vol. 2019).

[MB10] Missimer, E., & Betke, M. (2010, June). Blink and wink detection for mouse pointer control. In Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments (pp. 1-8).

[MCW+19] Meena, Y. K., Cecotti, H., Wong-Lin, K., & Prasad, G. (2019). Design and evaluation of a time adaptive multimodal virtual keyboard. *Journal on Multimodal User Interfaces*, 13(4), pp. 343-361.

[MKJ20] Meena, K., Kumar, M., & Jangra, M. (2020, May). Controlling mouse motions using eye tracking using computer vision. In 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS). pp. 1001-1005. IEEE. http://doi.org/10.1109/ICICCS48265.2020.9121137

[MNA08] Mark, S. (2008). Nixon and Alberto S. Aguado. Feature Extraction and Image Processing. In Academic Press (p. 88).

[MP+11] Mehrubeoglu, M., Pham, L. M., Le, H. T., Muddu, R., & Ryu, D. (2011, October). Real-time eye tracking using a smart camera. In 2011 IEEE Applied Imagery Pattern Recognition Workshop (AIPR). pp. 1-7. IEEE. https://doi.org/10.1109/AIPR.2011.6176373

[MSJ19] Mathew, S., Sreeshma, A., Jaison, T. A., Pradeep, V., & Jabarani, S. S. (2019). Eye Movement Based Cursor Control and Home Automation for Disabled People. 2019 International Conference on Communication and Electronics Systems (ICCES). http://doi.org/10.1109/icces45898.2019.9002325

[MRH+15] Membarth, R., Reiche, O., Hannig, F., Teich, J., Körner, M., & Eckert, W. (2015). Hipa cc: A domain-specific language and compiler for image processing. IEEE Transactions on Parallel and Distributed Systems, 27(1), 210-224.

[NAN+22] Nalinipriya, G., Aishwarya, T., Nandinisree, V. S., Sekkappan, K. N. K., Anand, S. G., & Shankar, S. S. (2022). An Innovative Rapid Cursor Control System Using Eye Movement for a Physically Challenged Person. In Modern Approaches in Machine Learning & Cognitive Science: A Walkthrough (pp. 241-247). Springer, Cham.

[NJ18] Najla P, R., & Jayasree T, C. (2018). Mouse Control Via Live Eye Gaze Tracking. International journal of engineering research and technology, 4.

[NRZ+18] Nasor, M., Rahman, K. M., Zubair, M. M., Ansari, H., & Mohamed, F. (2018). Eye-controlled mouse cursor for physically disabled individual. In *2018 Advances in Science and Engineering Technology International Conferences (ASET)*. pp. 1-4. IEEE.

[NUS+22] Naseer, F., Ullah, G., Siddiqui, M. A., Khan, M. J., Hong, K. S., & Naseer, N. (2022, May). Deep Learning-Based Unmanned Aerial Vehicle Control with Hand Gesture and Computer Vision. In *2022 13th Asian Control Conference (ASCC)*. pp. 1-6. IEEE.

[NYD16] Newell, A., Yang, K., & Deng, J. (2016, October). Stacked hourglass networks for human pose estimation. In *European conference on computer vision.* pp. 483-499. Springer, Cham.

[OtN79] Otsu, N. (1979). A threshold selection method from gray-level histograms. IEEE transactions on systems, man, and cybernetics, 9(1), 62-66.

[PJK+22] Pardeshi, S., Jagtap, S., Kathar, S., Giri, S., & Kapare, S. (May, 2022) AI Virtual Mouse and Keyboard. *International Journal of Advances in Engineering and Management (IJAEM).* 4(5), pp: 1925-1936. www.ijaem.net ISSN: 2395-5252

[PLM+22] Prameela, J., Lakshmi, K. V., Manju, K., & Devi, M. S.(2022). Mouse handling using facial gesture. 4(05). pp. 468-475.

[PND16] Puškarević, I., Nedeljković, U., Dimovski, V., & Možina, K. (2016). An eye tracking study of attention to print advertisements: Effects of typeface figuration. *Journal of eye movement research*, 9(5), 1-18.

[PPS+22] Peresunko, P., Pleshkova, E., Semizorova, A., & Kovalev, I. (2022). MLI. Mouse: a new computer vision-based cursor control software.

[PRL14] Plesnick, S., Repice, D., & Loughnane, P. (2014). Eye-controlled wheelchair. 2014 IEEE Canada International Humanitarian Technology Conference - (IHTC). doi:10.1109/ihtc.2014.7147553

[PSH18] Park, S., Spurr, A., & Hilliges, O. (2018a). Deep pictorial gaze estimation. In Proceedings of the European conference on computer vision (ECCV). pp. 721-738.

[PT14] Praglin, M., & Tan, B. (2014). Eye detection and gaze estimation. Eye, 1.

[PZB18] Park, S., Zhang, X., Bulling, A., & Hilliges, O. (2018b, June). Learning to find eye region landmarks for remote gaze estimation in unconstrained settings. In Proceedings of the 2018 ACM symposium on eye tracking research & applications. pp. 1-10.

[RAM21] Radmehr, A., Asgari, M., & Masouleh, M. T. (2021, November). Experimental Study on the Imitation of the Human Head-and-Eye Pose Using the 3-DOF Agile Eye Parallel Robot with ROS and Mediapipe Framework. In 2021 9th RSI International Conference on Robotics and Mechatronics (ICRoM) (pp. 472-478). IEEE. https://doi.org/10.1109/ICRoM54204.2021.9663445

[RWK15] Rafiqi, S., Wangwiwattana, C., Kim, J., Fernandez, E., Nair, S., & Larson, E. C. (2015, July). PupilWare: towards pervasive cognitive load measurement using commodity devices. In Proceedings of the 8th ACM International Conference on PErvasive Technologies Related to Assistive Environments (pp. 1-8).

[RoC22] Roy, K., & Chanda, D. (2022, February). A Robust Webcam-based Eye Gaze Estimation System for Human-Computer Interaction. In *2022 International Conference on Innovations in Science, Engineering and Technology (ICISET)*. pp. 146-151. IEEE.

[RSN09] Raudonis, V., Simutis, R., & Narvydas, G. (2009, November). Discrete eye tracking for medical applications. In *2009 2nd International Symposium on Applied Sciences in Biomedical and Communication Technologies* (pp. 1-6). IEEE.

[SAM+22] Souza, K. E. S. D., Aviz, I. L. D., Mello, H. D. D., Figueiredo, K., Vellasco, M. M. B. R., Costa, F. A. R., & Seruffo, M. C. D. R. (2022). An Evaluation Framework for User Experience Using Eye Tracking, Mouse Tracking, Keyboard Input, and Artificial Intelligence: A Case Study. International Journal of Human–Computer Interaction, 38(7), 646-660.

[SB15] Steil, J., & Bulling, A. (2015, September). Discovery of everyday human activities from long-term visual behaviour using topic models. In Proceedings of the 2015 acm international joint conference on pervasive and ubiquitous computing (pp. 75-85).

[SC18] Soundarajan, S., & Cecotti, H. (2018, July). A gaze-based virtual keyboard using a mouth switch for command selection. In 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). pp. 3334-3337. IEEE.

[SDO17] Sezer, O. B., Dogdu, E., & Ozbayoglu, A. M. (2017). Context-aware computing, learning, and big data in internet of things: a survey. IEEE Internet of Things Journal, 5(1), 1-27.

[SeS04] Sezgin, M., & Sankur, B. (2004). Survey over image thresholding techniques and quantitative performance evaluation. Journal of Electronic imaging, 13(1), 146-165.

[SKA21] Singh, A. K., Kumbhare, V. A., & Arthi, K. (2021, June). Real-Time Human Pose Detection and Recognition Using MediaPipe. In International Conference on Soft Computing and Signal Processing (pp. 145-154). Springer, Singapore.

[SMS14] Sugano, Y., Matsushita, Y., & Sato, Y. (2014). Learning-by-synthesis for appearance-based 3d gaze estimation. In Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1821-1828.

[SN+21] Shriram, S., Nagaraj, B., Jaya, J., Shankar, S., & Ajay, P. (2021). Deep learning-based real-time AI virtual mouse system using computer vision to avoid COVID-19 spread. Journal of healthcare engineering, 2021.

[SP16] Salunkhe, P., & Patil, A. R. (2016). A device controlled using eye movement. *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*. http://doi.org/10.1109/iceeot.2016.7754779.

[SPT+17] Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., & Webb, R. (2017). Learning from simulated and unsupervised images through adversarial training. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2107-2116).

[SR15] Sambrekar, U., & Ramdasi, D. (2015, December). Human computer interaction for disabled using eye motion tracking. In 2015 International Conference on Information Processing (ICIP) (pp. 745-750). IEEE.

[SwD13] Swirski, L., & Dodgson, N. (2013). A fully-automatic, temporal approach to single camera, glint-free 3D eye model fitting. *Proc. PETMEI*, 1-11.

[Syn20] Synced (2020). Eyes on Me: Google AI 'MediaPipe Iris' Improves Iris Tracking and Distance Estimation. https://syncedreview.com/2020/08/07/eyes-on-me-google-ai-mediapipe-iris-improves-iris-tracking-and-distance-estimation/

[Sze22] Szeliski, R. (2022). Computer vision: algorithms and applications. Springer Nature.

[Ten22] TensorFlow. GPU delegates for Tensorflow Lite, TensorFlow. Available at: https://www.tensorflow.org/lite/performance/gpu (Accessed: November 27, 2022).

[TiB11] Timm, F., & Barth, E. (2011). Accurate eye centre localisation by means of gradients. *Visapp*, *11*, 125-130.

[Tob22] Tobii (2022). Unlock the future of scientific research. https://www.tobii.com

[TPP19] Tresanchez, M., Pallejà, T., & Palacín, J. (2019). Optical mouse sensor for eye blink detection and pupil tracking: Application in a low-cost eye-controlled pointing device. *Journal of Sensors*, 2019.

[VaV20] Valentin Bazarevsky, Andrey Vakunov, 2020, "MediaPipe Iris: Real-time Pupil Tracking and Gaze Estimation", https://arxiv.org/abs/2106.03195

[VJG19] Vasisht, V. S., Joshi, S., & Gururaj, C. (2019, June). Human computer interaction based eye controlled mouse. In 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA). pp. 362-367. IEEE.

[VJS19] Vasisht, V. S., Joshi, S., Shashidhar, Shreedhar, & Gururaj, C. (2019). Human computer interaction based eye controlled mouse. 2019 3rd International Conference on Electronics, Communication and Aerospace Technology (ICECA). IEEE. https://doi.org/10.1109/iceca.2019.8822033

[WBM16] Wood, E., Baltrušaitis, T., Morency, L. P., Robinson, P., & Bulling, A. (2016). A 3d morphable model of the eye region. Optimization, 1, 0.

[WCD13] Wankhede, S., Chhabria, S., & Dharaskar, R. V. (2013). Controlling mouse cursor using eye movement. *International Journal of Application or Innovation in Engineering & Management*, 36, 1-7.

[Wei91] Weiser, M. (1991). The Computer for the 21 st Century. Scientific American, 265(3), 94-105.

[WSe21] Whaley, Sean (21 November 2018). "What is Frame Rate and Why is it Important to PC Gaming?". Retrieved 5 August 2021. https://www.hp.com/us-en/shop/tech-takes/what-is-frame-rate

[YcK14] Yılmaz, Ç. M., & Köse, C. (2014, April). Computer control and interaction using eye gaze direction detection. In 2014 22nd Signal Processing and Communications Applications Conference (SIU) (pp. 1658-1661). IEEE.

[YSF17] You, F., Li, Y., Schroeter, R., Friedrich, J., & Wang, J. (2017, September). Using eye-tracking to help design HUD-based safety indicators for lane changes. In Proceedings of the 9th International Conference on Automotive User Interfaces and Interactive Vehicular Applications Adjunct (pp. 217-221).

[ZBV20] Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C. L., & Grundmann, M. (2020). Mediapipe hands: On-device real-time hand tracking. arXiv preprint arXiv:2006.10214.

[ZD17] Zhu, W., & Deng, H. (2017). Monocular free-head 3d gaze tracking with deep learning and geometry constraints. In Proceedings of the IEEE International Conference on Computer Vision. pp. 3143-3152.

[ZJ05] Zhu, Z., & Ji, Q. (2005). Robust real-time eye detection and tracking under variable lighting conditions and various face orientations. Computer Vision and Image Understanding, 98(1), 124-154.

[ZKM17] Zhang, X., Kulkarni, H., & Morris, M. R. (2017, May). Smartphone-based gaze gesture communication for people with motor disabilities. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (pp. 2878-2889).

[ZSF15] Zhang, X., Sugano, Y., Fritz, M., & Bulling, A. (2015). Appearance-based gaze estimation in the wild. In Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4511-4520.

[ZU20] Zheng, C., & Usagawa, T. (2020, April). Self-training Pupil Detection Based on Mouse Click Calibration for Eye-Tracking Under Low Resolution. In 2020 6th International Conference on Control, Automation and Robotics (ICCAR). pp. 690-694. IEEE. https://doi.org/10.1109/iccar49639.2020.9107973

[Zwa22] Zwave (2022). You've made it home, let Z-Wave make it smart. https://www.z-wave.com/

[ZYC18] Zhang, C., Yao, R., & Cai, J. (2018). Efficient eye typing with 9-direction gaze estimation. Multimedia Tools and Applications, 77(15), 19679-19696.Zhang, X., Liu, X., Yuan, S. M., & Lin, S. F. (2017). Eye tracking based control system for natural human-computer interaction. Computational intelligence and neuroscience, 2017.