UNIVERSITY OF ECONOMICS HO CHI MINH CITY
COLLEGE OF TECHNOLOGY AND DESIGN
INSTITUTE OF INNOVATION
AI PROJECT

FINAL EXAMINATION

TOPIC: Application Designed to Support Patients with Amyotrophic Lateral Sclerosis (ALS)

Academic Supervisor: Ph. D Nguyen Thien Bao

Project Team: Group 5

Student Name:

Pham Huynh Bao Tran (Leader)

Hoang Thien Thu

Nguyen Khanh Linh

Nguyen Thuy An

Nguyen Thi My Duyen

Ho Chi Minh City, 2024

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ID	Abbreviation	Full Term	Description
1	ALS	Amyotrophic Lateral Sclerosis	A rare neurological disorder that affects motor neurons in the brain and spinal cord.
2	EAR	Eye Aspect Ratio	A measure used to determine eye state (open/closed) in eye tracking.
3	BCI	Brain-Computer Interface	A system that enables direct communication between the brain and external devices.
4	AAC	Augmentative and Alternative Communication	Communication methods used to supplement or replace speech or writing.
5	GUI	Graphical User Interface	A visual way of interacting with a computer using graphical icons and visual indicators.
6	API	Application Programming Interface	A set of rules allowing different software applications to communicate.
7	CNN	Convolutional Neural Network	A class of deep neural networks commonly used in computer vision.
8	ROI	Region of Interest	A selected subset of samples within a dataset for particular purpose.
9	FPS	Frames Per Second	The frequency at which consecutive images are captured/displayed.
10	IDE	Integrated Development Environment	Software application for computer programming.
11	DL	Deep Learning	A subset of machine learning based on artificial neural networks.
12	OSI	Operating System Interface	System software that manages hardware and software resources.

ABSTRACT

In today's digital era, AI technology is not only a trend but also a driving force for groundbreaking innovations. As final-year students majoring in Technology and Innovation, our research focuses on developing a cost-effective eye-tracking communication system to support ALS patients. The project is inspired by the story of *Stephen Hawking* - who battled ALS for 55 years and became one of the greatest physicists of the 20th century.

Now, as I study the subject of **"AI Projects"** in my final year of university, I realize this is the opportunity to turn dreams into reality—to create a technological solution that can assist individuals like *Stephen Hawking*. He once said, "*No matter who you are, no matter what circumstances you face, there is always something you can do. I try to focus on the things that disabilities allow me to do, rather than regret what I cannot do*".

"Like a star shrouded in darkness, ALS patients long to shine. How can we ignite the flame of hope for them?". We believe developing an affordable AI technology solution to help them communicate is absolutely essential.

Our system applies computer vision and AI technology, using ***Python libraries*** such as ***OpenCV, Dlib, MediaPipe,*** and ***Numpy*** to detect and track eye movements. The user interface is developed with ***PyQt5***, focusing on simplicity and accessibility. The system can recognize four main gaze directions and blinking states to control a virtual keyboard.

The main components of the system include:

- **A high-precision motion and blinking detection system.**
- **An intuitive virtual keyboard interface.**
- **A character input mechanism controlled by eye movements.**
- **Real-time video processing via a standard webcam.**

The key advantages of the solution are:

- **Cost-effective compared to commercial solutions.**
- **Minimal hardware requirements.**
- **Simple, user-friendly interface.**
- **Real-time processing and response.**
- **Future scalability.**

Although there are some limitations regarding lighting conditions and camera quality, the project aims to create a basic communication support tool at an affordable cost for ALS patients in Vietnam. As *Stephen Hawking* once said, *"No matter how difficult life may seem, you always can do something and succeed. I try to focus on the things that disabilities do not prevent me from doing instead of regretting what they stop me from doing".*

With the knowledge from university and the guidance of professors, we hope to contribute a small part to improving the quality of life for ALS patients. Though we are

still students with limitations in experience and resources, we believe that with youthful enthusiasm and a spirit of innovation, this project will be a meaningful first step in developing technological solutions for the community in Vietnam.

ACKNOWLEDGEMENTS

First and foremost, we would like to extend our deepest gratitude to lecturer Nguyen Thien Bao, who has guided and imparted invaluable knowledge to us throughout the AI Project course. Thanks to your dedicated, detailed, and methodical guidance, we have not only acquired essential knowledge but also expanded our understanding of AI applications across various fields.

You have provided us with knowledge and shared precious real-world experiences, helping us better comprehend the challenges and opportunities in applying AI to work and life. Your dedication in each lesson has inspired us to continually strive, learn independently, and develop analytical, evaluative, and problem-solving skills in a scientific and systematic manner.

Throughout the process of preparing this report, we have done our best to apply what we have learned. However, with our limited knowledge and practical experience, there are surely still shortcomings in our work. We sincerely look forward to receiving feedback and comments from you to improve and refine our efforts in the future.

Finally, we wish you good health, joy, and success in your teaching career. Thank you very much!

TASK ASIGNMENT TABLE

No	Member	ID	Assigned Task	Completion Level
1.	Pham Huynh Bao Tran (Leader)	31221021627	4. Experiment 5. Result and Analysis 6.Conclusion and Future Work	100%
2.	Nguyen Thi My Duyen	31221023919	1. Introduction 4. Experiment 6.Conclusion and Future Work	100%
3.	Nguyen Thuy An	31221021875	3.Method 4. Experiment 5. Result and Analysis	100%
4.	Nguyen Khanh Linh	31221025952	3.Method 4. Experiment 5. Result and Analysis	100%
5.	Hoang Thien Thu	31221024585	2. Related Work/Literature Review 4. Experiment 5. Result and Analysis	100%

1. Introduction

1.1 Why Choose the Topic?

In the quiet room, only the sound of Stephen Hawking's steady breathing and deep eyes looking towards the starry sky remained. Even though he was imprisoned in an immobile body, he still explored the vast universe with his superior intelligence. That image is always engraved in my mind, evoking a desire to do something meaningful.

The ability to communicate is the thread that connects us to the world. But for people with ALS, that thread seems to be severed, making them feel alone and lost. Seeing loved ones facing this cruel disease, unable to share their joys and sorrows, motivated me to search for a solution.

I believe that, with the continuous development of AI technology, we can bring a brighter future to people with ALS. A future where they can freely communicate, participate in social activities, and live a fulfilling life.

This project is not only a technological solution but also a humanitarian journey, aiming to help the less fortunate and contribute to the development of society. I call for attention and support from the community to work together to carry out more of these projects, bringing hope to people with ALS and other rare diseases.

1.2 Introduction to ALS

1.2.1 Definition and Characteristics of ALS

ALS (Amyotrophic Lateral Sclerosis), also known as motor neuron disease, is a rare neurological disorder that severely impacts the motor neurons in the brain and spinal cord. This is a form of progressive neurodegenerative disease, where motor neurons are damaged and lose their function, preventing muscles from receiving the signals needed to move. This leads to muscle weakness, which gradually progresses to complete paralysis, ultimately resulting in a total loss of voluntary motor function (Clinic 2023).

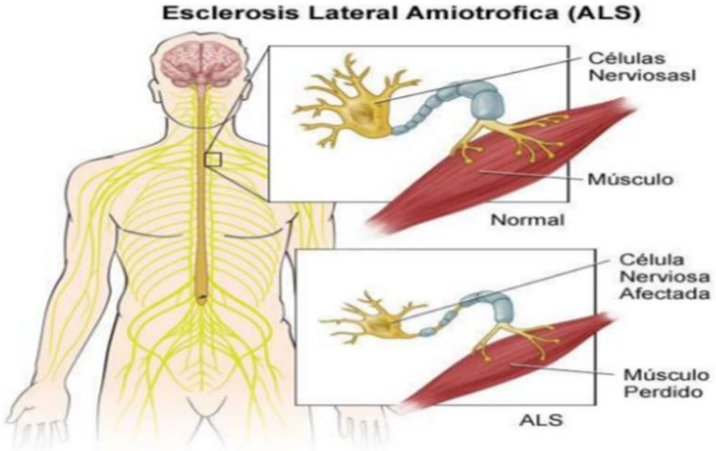

Figure 1.1 *Unusual events occur in neural activity in ALS and in normal populations.*

ALS is commonly found in adults from middle age onward, with a higher incidence in men than in women. The exact cause of ALS remains unclear, but scientists believe a combination of genetic and environmental factors may contribute to its development.

The disease progresses rapidly and currently has no cure; therefore, early detection and timely intervention can help improve the quality of life for those affected (Association 2024).

1.2.2 Effects of ALS on patients' lives

ALS greatly affects the patient's life in many aspects, from physical, mental to social. First of all, physically, people with ALS will experience gradual weakening and atrophy of muscles, starting with the muscles in the arms and legs, then spreading to the entire body, including the legs. respiratory muscles. This leads to the patient having difficulty with daily activities such as walking, eating, and even breathing. They may need the help of a Breathing machine and other assistive devices to stay alive (AnswerALS 2022).

Figure 1.2 Effects of ALS on patients' lives

Mentally, ALS patients often face negative emotions such as anxiety, depression, and a sense of loss. The gradual decline of their physical abilities makes them feel helpless, lacking autonomy, and self-esteem. Additionally, dependence on family members and healthcare providers can make it challenging for patients to maintain their dignity and self-respect (Strachan 2023).

From a social perspective, ALS reduces patients' ability to communicate as the motor neurons controlling speech-related muscles are also affected. This progressive loss of verbal communication hinders their ability to connect with loved ones and society, leading to isolation and a profound sense of disconnection, which can be a significant psychological burden for ALS patients.

1.2.3 Difficulties and challenges in supporting ALS patients

Supporting individuals with ALS poses significant challenges for families, communities, and the healthcare system.

Firstly, due to the rapid progression of ALS, preparing supportive care measures must be done early and continuously adjusted to meet the patient's needs at each

stage. However, this requires substantial financial resources and access to specialized healthcare services, which not all families can afford.

Secondly, in the later stages of ALS, patients require comprehensive support for both medical and daily living needs. Assistive devices such as ventilators, specialized beds, and electronic communication tools come with high costs, placing a heavy financial burden on families. Additionally, finding and training caregivers with specialized skills is challenging due to the knowledge and experience required to effectively support ALS patients.

These difficulties highlight the need for improved resources and support systems to aid ALS patients and their families effectively.

1.3 Objectives and Meaning of the Topic

1.3.1 Research Objectives

The main objective of this project is to design and develop an application that supports ALS patients in communication and controlling their surrounding environment through eye movement and gaze detection technology. This application is aimed at helping ALS patients, particularly those who have lost the ability to move and speak, to use their eyes to control a virtual keyboard and input characters, thereby assisting them in daily communication.

Figure 1.3 *Illustration of a patient using a device to communicate*

Specifically, the project sets forth the following detailed objectives:

- ***Develop a High-Precision Eye Movement and Blink Detection System:*** Create a system that accurately detects eye movements and the open/closed state of the eyes to meet the precise interaction needs of ALS patients.
- ***Design an Intuitive User Interface:*** Develop a user-friendly interface that is optimized for ALS patients, allowing them to easily operate the application through their gaze without any physical assistance.

- ***Create a Stable and Cost-Effective Application:*** Ensure that the application operates reliably and is compatible with common devices, while keeping investment costs low to increase accessibility for ALS patients and their families.
- ***Test and Evaluate the Application's Effectiveness:*** Conduct trials with ALS patients to assess and improve the application's performance and user experience, better meeting the actual needs of users.

These objectives not only provide practical benefits for ALS patients but also lay the groundwork for further research and development in the field of assistive technology for individuals with disabilities.

1.3.2 Significance of the Project for the ALS Patient Community

ALS is a serious disease and there is no cure, which causes many challenges in the lives of patients and their families. Applications to support ALS patients based on eye movement recognition are of great significance in improving their quality of life. Helping people with ALS to communicate through their eyes not only meets their basic needs but is also the key to helping them maintain connections with relatives and society.

Specifically, this application brings the following practical meanings to the ALS patient community:

- **Improve communication ability:** People with ALS can use their eyes to enter characters on the virtual keyboard, helping them communicate their opinions and emotions directly, helping to reduce isolation in communication.
- **Enhanced Autonomy:** The ability to independently use a communication tool without relying on others boosts patients' confidence and helps them maintain self-respect. This contributes to better mental health, alleviating negative feelings and reducing depression.
- **Support for basic living needs:** With this communication support application, patients can easily ask for help or express their living needs. This helps caregivers understand and promptly respond to the patient's needs, creating a more comfortable and pleasant living environment.

1.3.3 Contribution of The Topic to Society

In addition to direct benefits for ALS patients, this project also makes important contributions to society in general. As degenerative neurological diseases such as ALS become increasingly common and the number of patients increases, the development of advanced assistive technologies plays a major role in helping society respond to these challenges. awareness of health and quality of life. The contributions of the topic include:

- **Promoting scientific and technological progress:** This project applies advanced technologies such as image recognition, artificial intelligence and user interface in designing a useful tool for patients. This contributes to the

development of science and technology in the field of health and disability support, opening up the potential for similar research and applications in the future.

- **Raising Social Awareness:** By developing and introducing an application to support ALS patients, this project helps increase community awareness of ALS and the challenges patients face. This awareness can foster empathy, community support, and encourage resource mobilization from organizations, businesses, and government agencies.
- **Enhancing Disability Support Policies:** Assistive technologies, such as the eye-tracking application, can serve as a foundation for the development and implementation of supportive policies for people with disabilities. As these technologies demonstrate their effectiveness and significance, government and organizations may consider incorporating them into the list of assistive devices available for ALS patients and individuals with disabilities.
- **Contributing to community health:** With the development of assistive technologies, people with neurological diseases such as ALS have the opportunity to improve their quality of life, minimizing the burden on their families and society. This topic contributes to affirming the role of technology in improving community health, reducing medical costs and supporting patients to have a better life.

1.4 Current Research and Applications Supporting ALS Patients Worldwide

1.4.1 Current Research Supporting ALS Patients

At present, ALS (Amyotrophic Lateral Sclerosis) is a neurodegenerative disorder that does not have a complete cure. Therefore, research supporting ALS patients mainly focuses on improving quality of life, alleviating symptoms, and enabling patients to communicate and participate in daily activities. Around the world, scientists and experts are striving to find advanced technological solutions to support ALS patients in their daily lives.

Current ALS research spans various fields such as medical treatment, biological intervention, and notably, communication-assistive technology. Some studies are dedicated to developing medications and biological therapies that help slow the progression of the disease. For instance, clinical trials involving drugs like Riluzole and Edaravone have shown potential to extend the lifespan of ALS patients, though their effectiveness remains limited. Additionally, researchers are exploring gene therapy to block or alter genes associated with ALS, but these studies are still in their early stages and require more time for development.

Alongside medical approaches, research in assistive technology for ALS patients is growing. These studies aim to help patients maintain their communication abilities through technological solutions such as eye-tracking, brain signal detection, and

remote-control interfaces. Such methods allow ALS patients to control computers, type, and even operate household devices using brain signals or eye movements.

1.4.2 Limitations of Current Solutions

Despite significant advancements in the development of assistive technologies for ALS patients, current solutions still face many limitations. These include accuracy, high costs, limited accessibility, and compatibility issues with the diverse needs of patients.

- **Accuracy and Performance**: Assistive devices and software such as eye movement recognition systems and BCIs still encounter challenges in terms of accuracy and response speed. Eye-tracking technology sometimes becomes unstable due to lighting conditions, patient posture, or involuntary movements. BCI also has not yet achieved high accuracy in translating brain signals into desired actions, leading to extended training periods for patients to effectively use the device.
- **High Costs**: Advanced assistive devices such as Tobii Eye Tracker or BCI systems are currently very expensive, making it difficult for many patients and their families to afford them. Eye-tracking and BCI devices are not only costly in terms of equipment but also require software and maintenance services, increasing the overall cost of usage.
- **Accessibility**: Although ALS support technology has been developed in many countries, access to these technologies is still limited. Patients in developing countries or rural areas face challenges accessing advanced devices due to a lack of support services and consultation. This creates inequalities in technology access across different regions.
- **Compatibility with Patient Needs**: Each ALS patient experiences different progression levels and has unique needs, so a general technological solution may not meet all specific requirements. For example, some patients can control devices with their eyes, while others may need BCI. Therefore, assistive devices need greater flexibility to be compatible with various levels and patient needs.

1.5 Scope and Limitations of the Study

1.5.1 Scope of the Study

The study focuses on developing a communication support application for ALS patients, primarily using eye movement recognition and eye open/closed state detection to control a virtual keyboard. The scope of the research includes the following key aspects:

- **Target Audience**: The study targets ALS patients, especially those who have lost the ability to move and communicate verbally. These patients have special needs for communication support tools to help them maintain connections with their loved ones and society.
- **Application of Eye Movement Recognition Technology**: The study applies models and libraries in Python such as OpenCV (cv2), Dlib, MediaPipe, and

Numpy to recognize eye movement and eye open/closed states. This technology will be used to detect the patient's gaze directions (up, down, left, right) to control the cursor on a virtual keyboard and input characters.

- **Interface Design Using PyQt5**: The application interface is designed using the PyQt5 library, a popular Python library for creating user interfaces. This interface will be optimized for ALS patients, with a simple and user-friendly layout to facilitate ease of use without requiring complex actions.
- **Testing and Effectiveness Evaluation**: Initial tests will be conducted on a small number of patients or through simulations to assess the application's accuracy and usability. The tests will focus on evaluating the accuracy of eye movement recognition, eye open/closed state accuracy, and the interface's convenience in practical use.
- **System Operation and Compatibility**: The study scope does not include developing the application for multiple operating systems. The application will be developed and tested on personal computers (PCs) or laptops with Windows OS. Implementing the application on other operating systems such as macOS or Linux may not be covered in the study.

1.5.2 Limitations and External Factors

Although the study has the potential to contribute significantly to the ALS patient community, there are also limitations and external factors that may affect the effectiveness and scope of the application. These limitations include:

- **Accuracy of Eye Movement Recognition**: The accuracy of eye movement and eye open/closed state recognition depends heavily on factors such as lighting conditions, camera quality, and the patient's posture. In unstable lighting conditions or if the patient changes posture, the system's recognition accuracy may decrease. Additionally, low-quality cameras may not perform well in eye-tracking, affecting user experience.
- **Hardware Requirements and Costs**: While the application uses common libraries and tools, achieving high accuracy in eye recognition requires a camera with good resolution and high frame rate. This may increase equipment costs, making it difficult for ALS patients to access and use the application, especially those in economically disadvantaged areas or with limited financial resources.
- **Testing Method Limitations**: Testing the application directly on ALS patients requires complex procedures and conditions, so the application may only be tested through simulations or hypothetical scenarios. The absence of real patient testing may lead to limitations in evaluating effectiveness and optimizing the application according to actual needs. Results from simulation tests may not fully reflect the experiences and needs of ALS patients using the application in daily life.

- **Scalability and Compatibility**: The application within the scope of this study focuses solely on basic communication by inputting characters on a virtual keyboard. Additional features such as controlling home appliances, integration with messaging applications, or internet connectivity are not included in the research scope. This limits the application's scalability and its ability to fully meet the diverse needs of ALS patients, particularly those related to independent living and social connectivity.
- **Impact of External Factors**: External factors such as lighting, noise, and the patient's ability to focus also affect the application's usability. For instance, in low-light environments, eye recognition systems may not function stably, reducing typing accuracy. Moreover, ALS patients at different stages of progression may have different abilities to use the application; those in the late stages may find it challenging to maintain eye control, impacting usability.
- **Individual Factors and Patient Adaptation**: Each ALS patient may have different levels of eye control, so using the application requires time for adaptation and personalization adjustments. This requires patience and support from family members or specialists during the learning process. Factors such as eye responsiveness, the ability to focus on the interface, and the duration the patient can continuously use the application also affect user experience and application effectiveness.

2. Related Work/Literature Review

2.1 Introduction t o Eye-Tracking Technology

Eye-tracking technology has been widely adopted as assistive technology, especially for individuals with ALS who face movement difficulties but can still control their eye movements. This technology enables users to interact with digital interfaces and enhance their communication abilities by tracking gaze points and eye movements.

Figure 2.1 *Eye Gaze and Movement Tracking Technology*

2.1.1 Overview of Techniques and Algorithms in Eye-Tracking Research

Table 2.1 *Eye-Tracking Techniques (Y O. EDUGHELE 2022)*

Techniques	Merits	Demerits
Scleral Search Coil e.g EyeContact	• High temporal and spatial resolution. • High accuracy.	• Invasive method. • Rarely used clinically. • May cause Intraocular pressure. • Wear the coil for a short period of time. • Does not work on sensitive eyes. • Anaesthesia of the eye. • Complicated settings.
Infrared Oculography (IOG) e.g EyeLink 1000 optical system	• Used in light and darkness. • Handles blinking accurately.	• Unable to quantify torsional eye movement. • Limited movement of the head. • Invasive method.
Electro Oculography (EOG) e.g EOG based ALS system, e.g wireless EOG based Human Computer Interfaces, JINS MEME	• Medical fields and laboratories. • Very high temporal and spatial resolution. • Used ML to found accuracy. • Analyse the data in real time by using microcontroller.	• Not used daily. • Affected by the noise around the eye. • Invasive method. • Complicated settings.
Video Oculography (VOG) e.g Free Visual Exploration (FVE), EyeSeeCam	• Use visible light or infrared light. • Clinical observation of eye movement disorders.	• Limited spatial resolution. • Recording with closed eyes is not possible.

	• Video recording system is easily handled. • Uncomplicated settings. • Can allow head movement and fully remote recording. • Use of ML techniques to gain higher accuracy. • Not expensive.	

⇨ **Comparison of Techniques:** Techniques such as Electrooculography (EOG) and Video-Based Oculography (VOG) differ in terms of accuracy and user comfort. EOG utilizes the standing potential between the cornea and retina with electrodes, whereas VOG relies on high-resolution video, which is less invasive and allows for remote operation.

Table 2.2 *AI Algorithms Used in Eye-Tracking Research (Y O. EDUGHELE 2022)*

Algorithms	Merits	Demerits
Convolutional Neural Networks	• High accuracy	• Poorly regulated parameters such as sample size may affect accuracy of the results if they are not appropriately controlled
Support Vector Machine	• The use of SVM for prediction analysis is feasible using eye-tracking data	• Due to the small sample size, generalizability is limited
Random Forest (RF)	• Doesn't require scalability, features can be used in their current state	• The RF algorithm is incapable of learning the sequence context information directly from raw data and provides the final

		output without any postprocessing
Artificial Neural Network (ANN)	• It has been demonstrated that the ANN-based prediction model generates a high correlation coefficient between the original and forecasted data	• The prediction model's statistical power and reliability are hindered by short sample size. To validate the provided data, a study with controlled illumination conditions is required
Deep learning	• The accuracy and performance of detection systems have been increased due to deep learning	• In state-of-the-art detection systems, search efficiency is still a serious concern

⇨ Common models include **SVM (Support Vector Machines)** for cognitive state detection and gaze prediction, **CNN (Convolutional Neural Networks)** for pupil and gaze point detection, and **RNN (Recurrent Neural Networks)** for real-time eye movement prediction.

2.1.2 Comparative Table of Eye-Tracking Systems

Table 2.3 *Comparative Table of Eye-Tracking Systems*

Technology	**AI Model**	**Data Type**	**Evaluation Metrics**	**Summary of Results**	**Limitations**
Tobii Eye-Tracker	CNN, SVM	ALS patients	Pixel deviation, Stability	High accuracy, reliable but sensitive to head movement	High cost, sensitive to head movement
PyGaze	Python open-source	General public	Calibration error rate	Customizable, effective in lab settings	Requires specialized hardware, limited to lab usage

Gazepoint GP3	Low-cost device	Cognitive awareness	Time stability	Affordable, useful for basic research	Limited to controlled environments
OpenGazer	Face recognition	ALS patients	Detection accuracy	Supports real-time eye tracking, AAC integration	Requires regular calibration, fixed setup

- **Eye Movement Detection and Eye State Recognition Technology**: Currently, devices like Tobii Eye Tracker and Gazepoint GP3 use gaze-tracking technology but often require expensive hardware and their stability heavily depends on lighting conditions.

Figure 2.2 *Tobii Eye Tracker 5 - Head Tracking and Eye Tracking in One Device*

Figure 2.3 Gazepoint GP3 Tracking Device

Applying open-source technologies such as OpenCV and Dlib can minimize hardware costs while maintaining accuracy through the EAR (Eye Aspect Ratio) algorithm for eye state recognition.

Improvement: Utilizing the EAR algorithm for detecting eye open/closed states helps reduce errors and enhances recognition efficiency under varying lighting conditions, which is a weakness of existing devices.

- **Gaze Direction Detection through Machine Learning and Neural Networks**: Current eye-tracking systems may face challenges in accurately detecting gaze direction when the user slightly shifts their head position or in unstable environments. Dlib and MediaPipe are suitable technologies for this task, as they provide face detection and eye landmark recognition tools to determine gaze direction quickly and accurately.

  Improvement: Integrating MediaPipe to increase accuracy in gaze direction detection through deep learning, while using Numpy to optimize data processing to reduce latency and improve response speed.

- **User Interface via PyQt5**:

  To replace software with complex or non-user-friendly interfaces for patients, designing a simple and intuitive virtual keyboard using PyQt5 will help ALS patients operate it easily with their eyes.

  Improvement: This interface will be optimized for patients by using large button sizes and instant feedback for eye-gaze interactions, an improvement that enhances user experience compared to existing systems that often require specialized hardware.

2.2 Related Work

Technological applications that support ALS patients are being developed worldwide with the goal of helping patients maintain communication and enhance independence in daily life. Some of the currently popular assistive technologies include Augmentative and Alternative Communication (AAC) devices, eye movement recognition systems, and Brain-Computer Interface (BCI) systems.

- **Augmentative and Alternative Communication (AAC)**: AAC encompasses a group of technologies and devices that allow ALS patients to communicate without using their voice or body movements. AAC devices such as electronic letterboards or computers with supportive software help patients input characters by pressing buttons or eye movements. For example, EyeGaze and Tobii Dynavox software are popular applications that enable patients to use their eyes to select characters on the screen, allowing them to convey their thoughts and emotions.

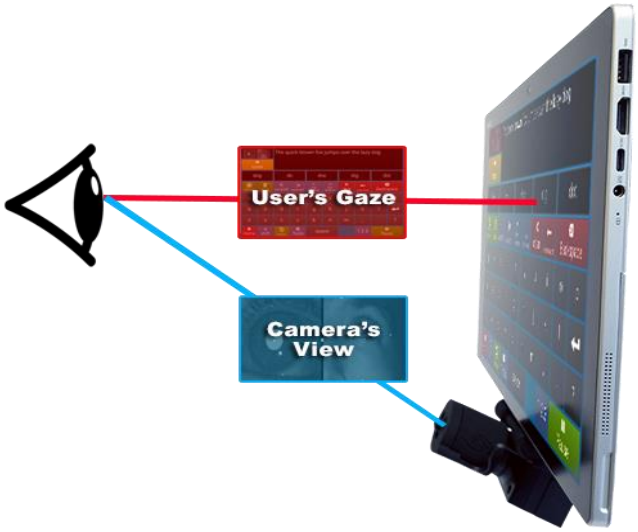

Figure 2.4 *Augmentative and Alternative Communication (AAC)*

- **Eye Movement Recognition Systems**: This is an advanced assistive technology that enables patients to control devices through eye movements. Eye-tracking devices work by detecting and tracking eye movements, enabling patients to control the cursor on the screen, input text, and even manage household appliances. This technology offers greater autonomy and facilitates effective communication for patients. Notable devices in this field include Tobii Eye Tracker and EyeTech Digital Systems.

Figure 2.5 *Tobii Eye Tracker and EyeTech Digital Systems*

- **Brain-Computer Interface (BCI)**: BCI is a technology that allows patients to control devices using brain signals. By wearing a sensor headset, BCI can detect and translate brain signals into actions on a computer. This technology has been tested for supporting ALS patients in performing simple tasks such as moving the mouse cursor, typing, and controlling smart home devices. Although BCI is still in development, its potential for assisting ALS patients is significant.

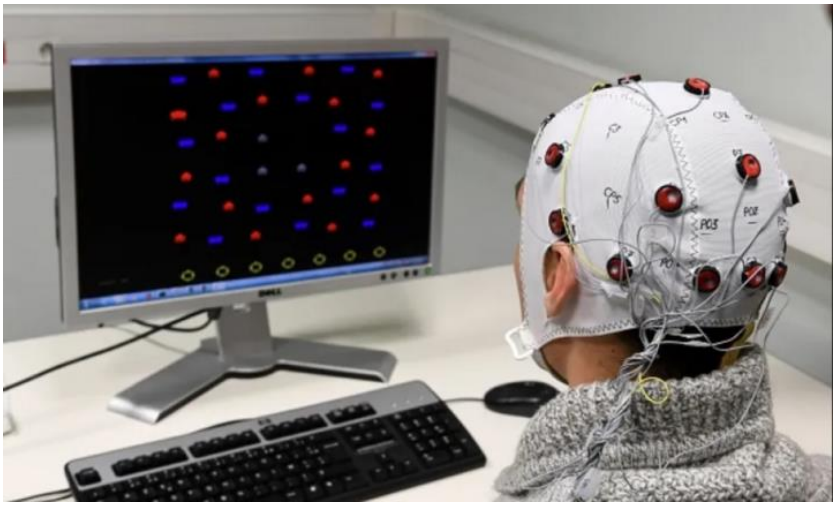

Figure 2.6 Brain-Computer Interface (BCI)

3.Method

3.1 Analysis of Eye Behavior and States

Recognizing eye behavior and states is crucial in communication-assistive applications for ALS patients, as it helps identify eye movements and states to control functions on an interface. Methods such as eye tracking, Eye Aspect Ratio (EAR), and gaze direction detection enable the application to understand and respond to signals from patients' eyes, creating convenient communication and interaction possibilities.

3.1.1 Principles of Eye Tracking

(Javier Gonzalez-Sanchez 2017)

Eye tracking is the process of identifying the position and movement of the eyes to understand the user's focus and gaze direction. Eye-tracking technology relies on capturing eye images and analyzing the movement of the pupil and cornea to determine where the user is looking. Currently, two popular methods in eye tracking are Pupil Center Corneal Reflection (PCCR) and image-based tracking.

- ***Pupil Center Corneal Reflection (PCCR):*** This method uses infrared light to illuminate the eye, creating a reflective point on the cornea. A camera captures the positions of the pupil and the reflection point on the cornea to calculate the gaze direction. The advantage of PCCR is its high accuracy and ability to operate in stable lighting conditions. However, it requires high-resolution, high-frame-rate cameras, which can increase equipment costs.
- ***Image-based Tracking:*** This method uses image processing techniques to detect and track the position of the pupil and eyelids. Image processing and machine learning models recognize characteristic points on the eye to determine the gaze direction. Dlib and MediaPipe are two popular Python libraries used for eye detection and tracking. This method can work with standard cameras and has fewer hardware requirements; however, accuracy may be affected in low-light conditions or with unintentional eye movements.

Based on these principles, an eye-tracking system can determine the gaze direction of ALS patients to control interfaces or perform actions on the application, helping patients communicate and control devices more conveniently.

3.1.2 Eye Aspect Ratio (EAR)

(Chang 2022)

The Eye Aspect Ratio (EAR) is an important metric in detecting the open or closed state of the eyes. EAR is calculated based on the ratio of eye height to width, making it easy and efficient to determine the eye state in real time. Specifically, the EAR is determined by the characteristic points around the eye, identified through image processing models like Dlib.

EYE								
EAR	0.40	0.39	0.34	0.33	0.31	0.29	0.21	0.12

Figure 3.1 Eye Aspect Ratio calculation

- ***EAR Calculation Formula:*** EAR is calculated based on the distances between key landmarks on the upper and lower parts of the eye, as well as the eye's width. The EAR formula is as follows:

$$EAR = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2 \times \|p_1 - p_4\|}$$

  In this context, *p1* to *p6* are the key landmarks around the eye identified by the image processing model. When the eye is open, the EAR value is high; when the eye is closed, the EAR value decreases. A specific EAR threshold for determining the open/closed state of the eye can be established through experiments to achieve maximum accuracy (Mohamed Ezzat 2023).

- ***Application of EAR in Eye State Recognition:*** EAR is a simple yet effective metric for identifying eye states. When the EAR value drops below a predefined threshold, the system can determine that the eye is in a closed state. This is very useful in communication-assistive applications, allowing ALS users to select characters or perform actions by closing their eyes. EAR helps minimize errors in eye state recognition and enhances the stability of the application.

3.1.3 Methods for Gaze Direction Recognition

(Tetsutani 2004)

Gaze direction recognition is the process of determining where the eyes are looking, enabling the system to interpret user signals into control commands on the interface. There are various methods for gaze direction recognition, each with its advantages and limitations, including:

- ***Image-based Methods***: This approach uses images of the eye to determine the position and direction of the pupil. As the eye looks up, down, left, or right, the pupil's position changes accordingly. The system can track these positional

changes to determine gaze direction. Image processing libraries like OpenCV, Dlib, and MediaPipe provide tools for detecting and tracking pupil positions, helping the system accurately identify the user's gaze direction. This method is straightforward, but accuracy may be affected by lighting conditions and unwanted movements.

- ***Neural Network Methods:*** Some studies use neural networks to improve the accuracy of gaze direction recognition. Neural networks can learn from large image datasets to predict gaze direction based on eye and facial features. This method generally offers higher accuracy than simpler image-based methods but requires significant computational resources and longer training times, making it less optimal for applications with low hardware requirements.
- ***Hybrid Methods:*** Hybrid methods combine PCCR and image processing models to enhance the accuracy of gaze direction recognition. By utilizing the reflection points of light on the cornea and pupil images, the system can accurately determine gaze direction even with changes in lighting conditions. However, this method requires specialized hardware, such as infrared lighting systems and high-resolution cameras, which can increase costs.
- ***Relative Motion-Based Methods:*** In cases where landmark-based methods cannot be employed, the system can recognize gaze direction based on the angle changes of the head and eyes. This approach does not require precise identification of each eye landmark but relies on the overall movement of the face and eyes to predict gaze direction. While this method may have lower accuracy, it can be suitable for applications that do not require high precision.

3.2 Technologies and Models Used

In developing communication-assistive applications for ALS patients, various technologies and libraries are employed to optimize eye movement recognition and create user interfaces. These technologies include OpenCV, Dlib, MediaPipe, NumPy, and PyQt5. Below are details about each technology and its role in the project.

3.2.1 OpenCV (CV2)

(Boesch 2024)

OpenCV (Open Source Computer Vision Library) is a well-known open-source library in the field of image processing and computer vision. This library provides powerful tools for performing image processing tasks such as object detection, face recognition, and motion tracking. In this project, OpenCV plays a crucial role in recognizing and tracking the eye movements of ALS patients.

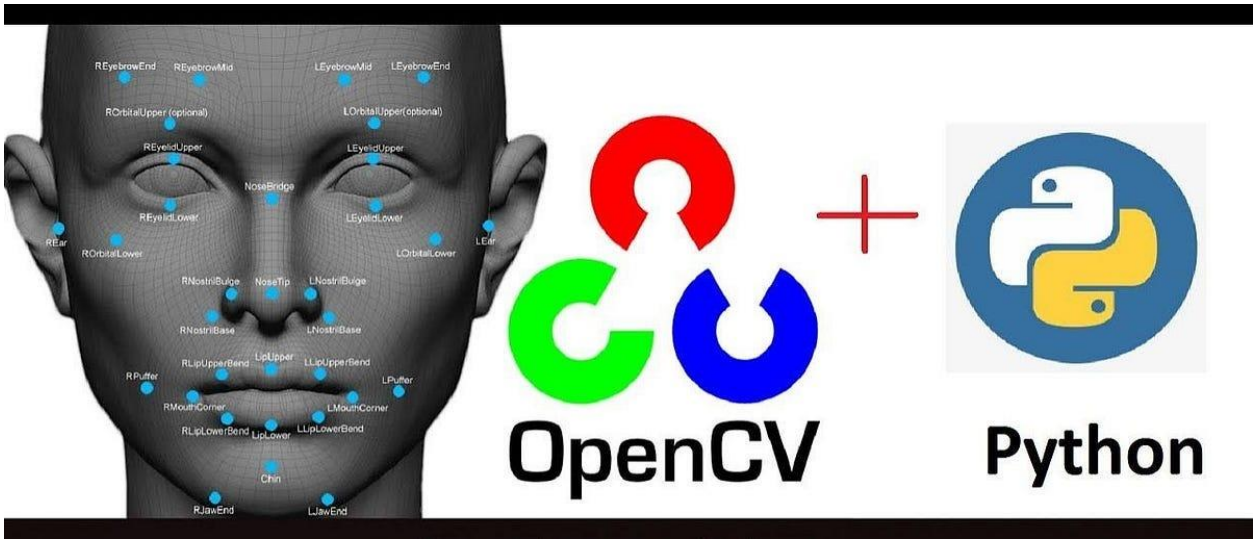

Figure 3.2 OpenCV library

Specifically, OpenCV supports:

- ***Face and Eye Detection:*** OpenCV provides pre-trained models such as cascade classifiers to detect faces and eyes from video images. The eye detection process is the first step in tracking gaze direction and determining eye states (open/closed).
- ***Pupil Detection:*** OpenCV can process images to detect the pupil by using image filters and searching for characteristic points. This helps determine the position of the pupil, which in turn identifies the gaze direction.
- ***Real-Time Image Processing:*** With real-time image processing capabilities, OpenCV allows for quick tracking of changes in eye movement, ensuring that the system can respond immediately to user actions.

Thanks to its powerful capabilities and flexibility, OpenCV is a crucial tool for the system to effectively process images and recognize eye movements, ensuring high accuracy in identifying states and gaze directions.

3.2.2 Dlib

(Rosebrock 2017)

Dlib is a powerful machine learning library widely used in face recognition and analysis applications. One of Dlib's standout features is its ability to identify landmark points on the face, such as the eyes, nose, and mouth. In this project, Dlib supports the recognition of characteristic points around the eyes, helping to determine the open/closed state of the eyes and recognize gaze direction.

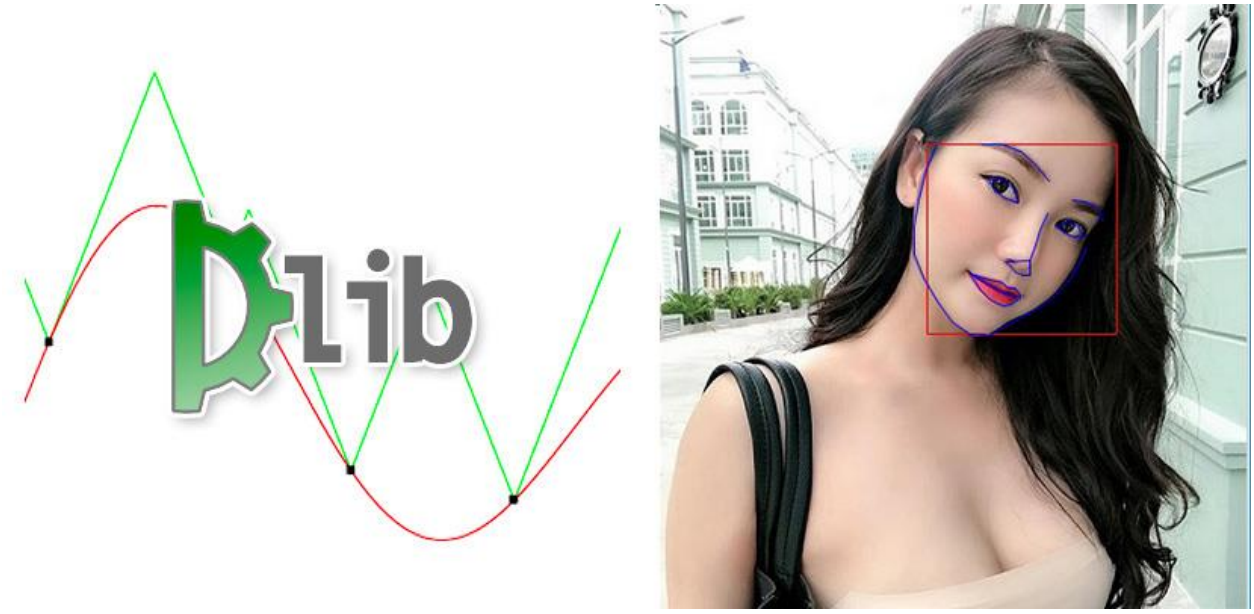

Figure 3.3 Dlib library

Specifically, Dlib provides:

- ***Face Detection and Landmark Points***: Dlib uses a facial landmark detection model to identify 68 characteristic points on the face, including 6 points around the eyes. These points help determine the Eye Aspect Ratio (EAR) to detect the open/closed state of the patient's eyes.

- ***Eye Movement Tracking and Analysis:*** With the landmark points around the eyes, Dlib aids in analyzing eye movement, supporting the identification of gaze direction and eye states. Combined with machine learning models, Dlib enhances the accuracy of detecting signals from the eyes, playing a crucial role in recognizing gaze direction and the open/closed state.
- ***Integration with Other Libraries:*** Dlib can easily integrate with OpenCV, making face and eye detection operations more efficient and accurate, especially in real-time image processing environments.

With its ability to accurately analyze the landmark points on the face, Dlib is an indispensable tool for determining the position and state of the eyes, enabling the application to understand and respond to signals from ALS patients.

3.2.3 MediaPipe

(Trần 2021)

MediaPipe is a library from Google that specializes in real-time video processing and analysis. MediaPipe offers many advanced solutions for face detection and tracking of hands, eyes, and poses, making user-interactive applications more responsive and accurate. In this project, MediaPipe is utilized to enhance eye detection and tracking capabilities.

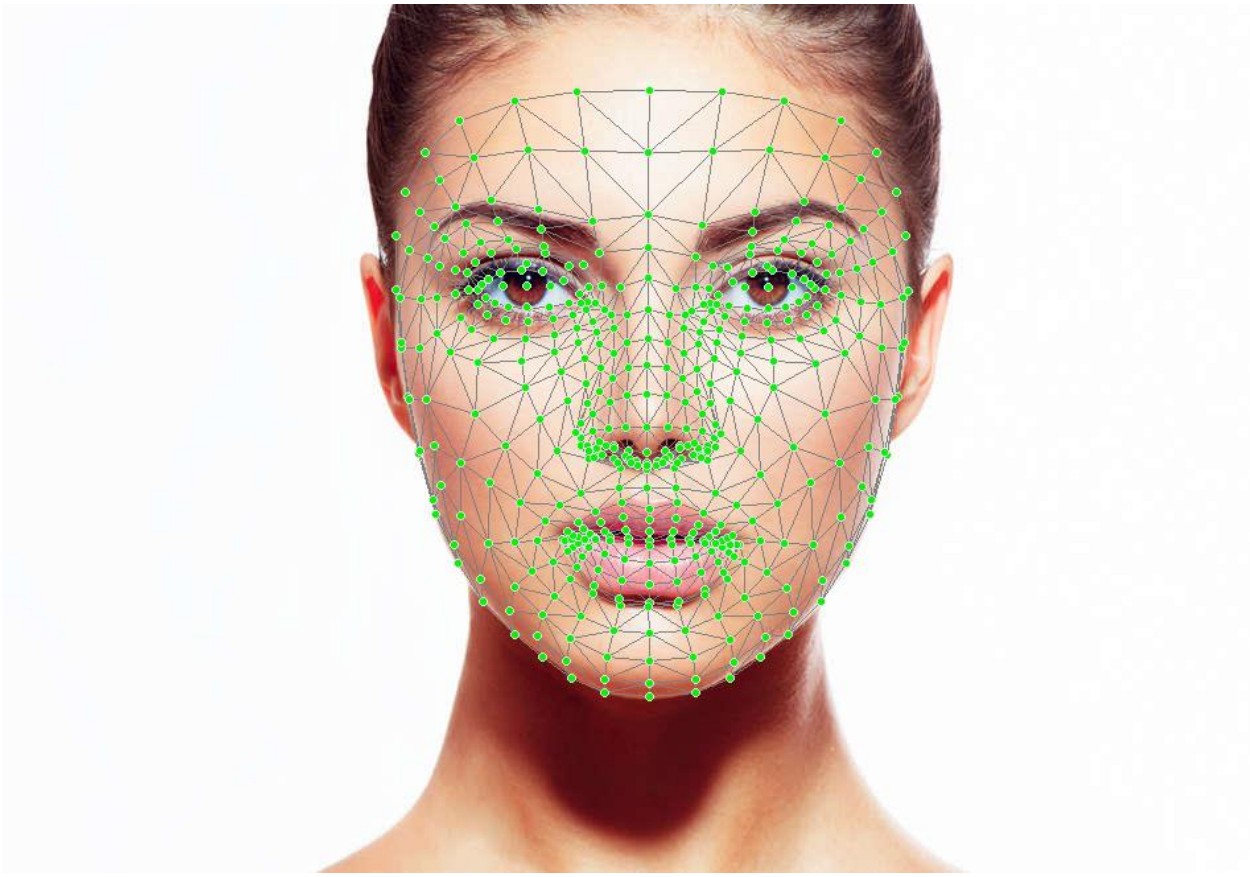

Figure 3.4 MediaPipe face tracking

Some standout features of MediaPipe include:

- ***Face and Eye Detection:*** MediaPipe uses deep learning models for accurate face and eye detection. This helps minimize detection errors under unstable lighting conditions and maintains high accuracy even when the patient makes slight movements.
- ***Real-Time Eye Detection and Tracking***: MediaPipe provides lightweight and optimized models for devices, allowing for real-time eye and face tracking without causing lag. This is an important feature to ensure that the application can respond promptly to even the smallest eye movements.
- ***Cross-Platform Functionality:*** MediaPipe can operate across various platforms, from personal computers to mobile devices. This expands the deployment capability of the application on different types of devices, increasing accessibility for ALS patients.

MediaPipe plays a crucial supportive role in accurately and sensitively recognizing and tracking eye movements, enhancing the effectiveness of the application in detecting signals from users' eyes.

3.2.4 Numpy

(Daniel 2023)

Numpy is a popular numerical data processing library in Python, providing powerful tools for matrix calculations, multi-dimensional data processing, and data analysis. In

this project, Numpy is used to process image data and calculate necessary metrics such as EAR.

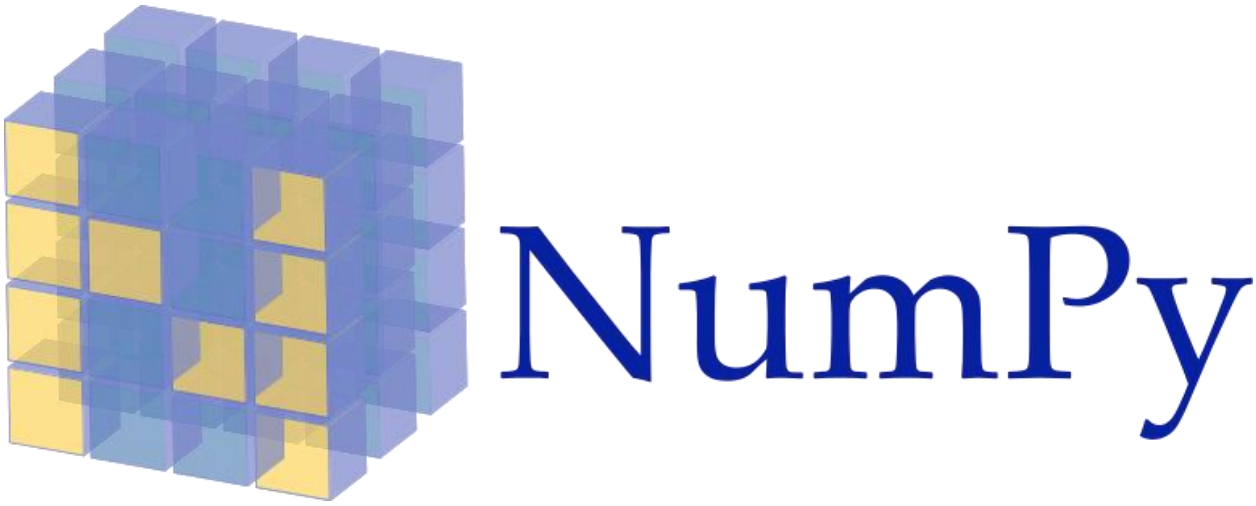

Figure 3.5 NumPy library

The role of Numpy includes:

- ***Image Data Processing and Analysis:*** Data from OpenCV and Dlib is often stored as image matrices. Numpy helps process these matrices to easily compute the distances between characteristic points around the eyes, aiding in gaze direction and eye state recognition.
- ***Calculating the EAR Metric:*** Using the landmark points identified by Dlib, Numpy is utilized to compute the EAR metric based on the distances between points. This metric is used to accurately determine whether the user's eyes are open or closed.
- ***Performance Optimization***: Numpy is a library known for its fast processing speed and optimization for matrix and vector operations. This helps reduce the processing time of the system, ensuring that the application can respond immediately to eye movements.

Due to its efficiency and powerful processing capabilities, Numpy is a crucial tool for optimizing the application's performance in computational tasks and image data processing.

3.2.5 PyQt5

(Abhilekhnathdas 2022)

PyQt5 is a popular library in Python for building graphical user interfaces (GUIs). This library provides tools for creating windows, buttons, forms, and other interface components, helping to build a user-friendly and easy-to-use application. In this project, PyQt5 plays an important role in constructing the virtual keyboard interface and other interactive components, allowing ALS patients to communicate easily through their eye movements.

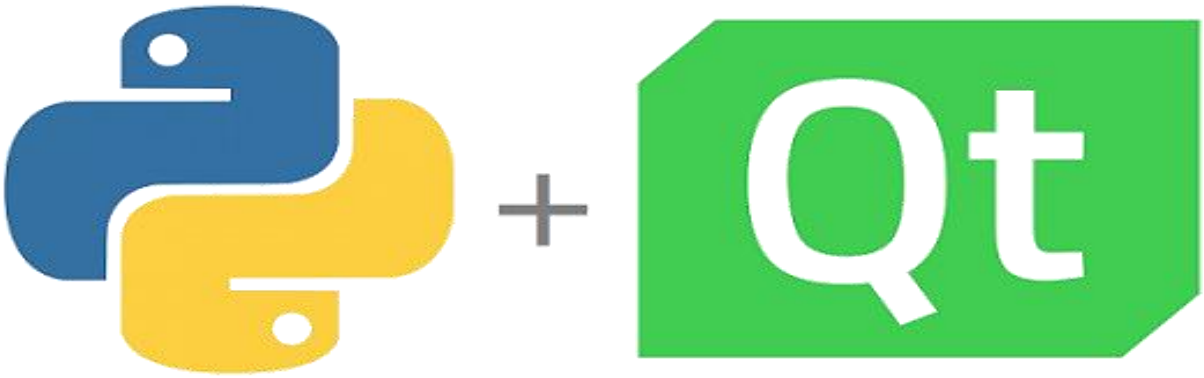

Figure 3.6 PyQt5

PyQt5 provides the following functions:

- ***Designing a Virtual Keyboard Interface:*** PyQt5 allows the construction of a virtual keyboard with large buttons and an intuitive interface, making it easier for ALS patients to use. Users can control the cursor and select characters with their gaze, facilitating text input and effective communication.
- ***Immediate Interaction and Feedback:*** PyQt5 supports instant updates to the interface based on user actions. When the gaze direction or eye state (open/closed) is detected, PyQt5 ensures that the virtual keyboard interface responds quickly, enabling users to select characters or perform actions without delay.
- ***Customizable User Interface:*** PyQt5 allows customization of colors, sizes, and layouts of interface components, creating a friendly interface that meets the needs of ALS patients.

With its flexible and easily customizable interface capabilities, PyQt5 is an ideal tool for creating a user-friendly application that enables ALS patients to communicate and interact naturally.

3.3 Related Algorithms and Techniques

3.3.1 Face and Eye Detection

Face and eye detection is one of the first and most important steps in communication assistance applications for ALS patients, especially when the application uses eye movements as a control method. This process allows for real-time identification of the user's face and eye positions, which helps track eye movements or states.

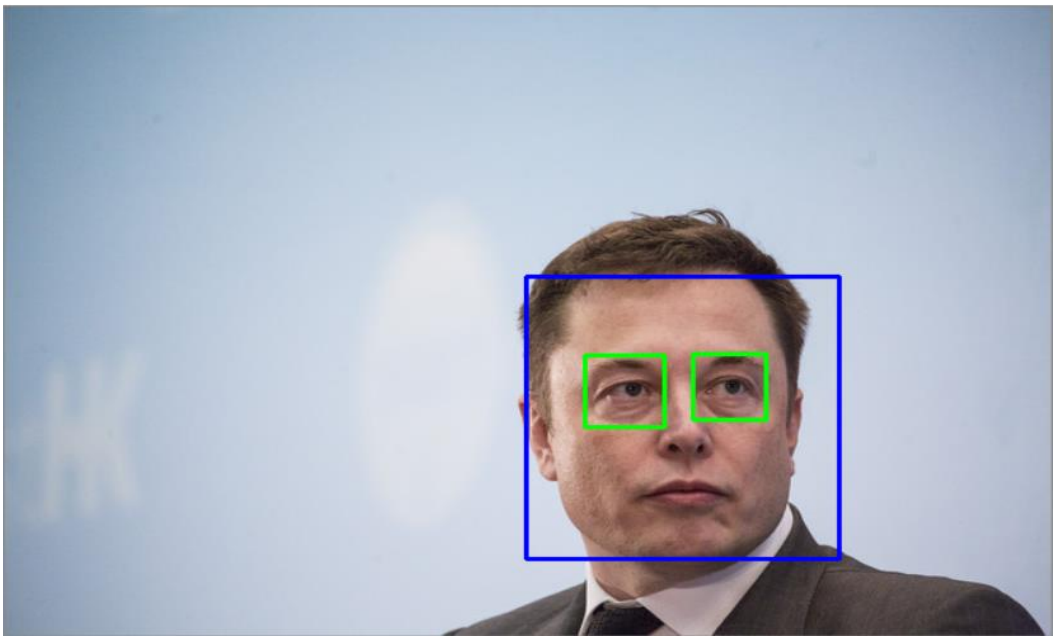

Figure 3.7 Face and eye detection

There are various methods for detecting faces and eyes; however, the most common approaches use machine learning and computer vision techniques. One widely used method is the "Cascade Classifiers" by Viola-Jones, which is integrated into the OpenCV library. This machine learning algorithm is based on Haar-like features and allows for the detection of faces in images or videos by analyzing the characteristic structure of a face. Cascade Classifiers work by creating a series of classifiers, and only when an image region passes through all the stages is it determined to contain a face or eye. This method has the advantage of fast processing speed, making it suitable for real-time applications.

Additionally, deep learning models, such as Convolutional Neural Networks (CNNs), can also be employed to detect faces and eyes with higher accuracy. Models like MTCNN (Multi-task Cascaded Convolutional Networks) or YOLO (You Only Look Once) can recognize faces and eyes in complex lighting conditions and different viewpoints. However, these models require more computational resources, so consideration must be given when applying them to devices with limited configurations.

3.3.2 Gaze Detection

Gaze detection is a technique used to determine the direction in which a user is looking. For ALS patients, gaze direction becomes the primary control method in communication assistance applications, enabling patients to select characters on a virtual keyboard or execute control commands without needing to move their bodies.

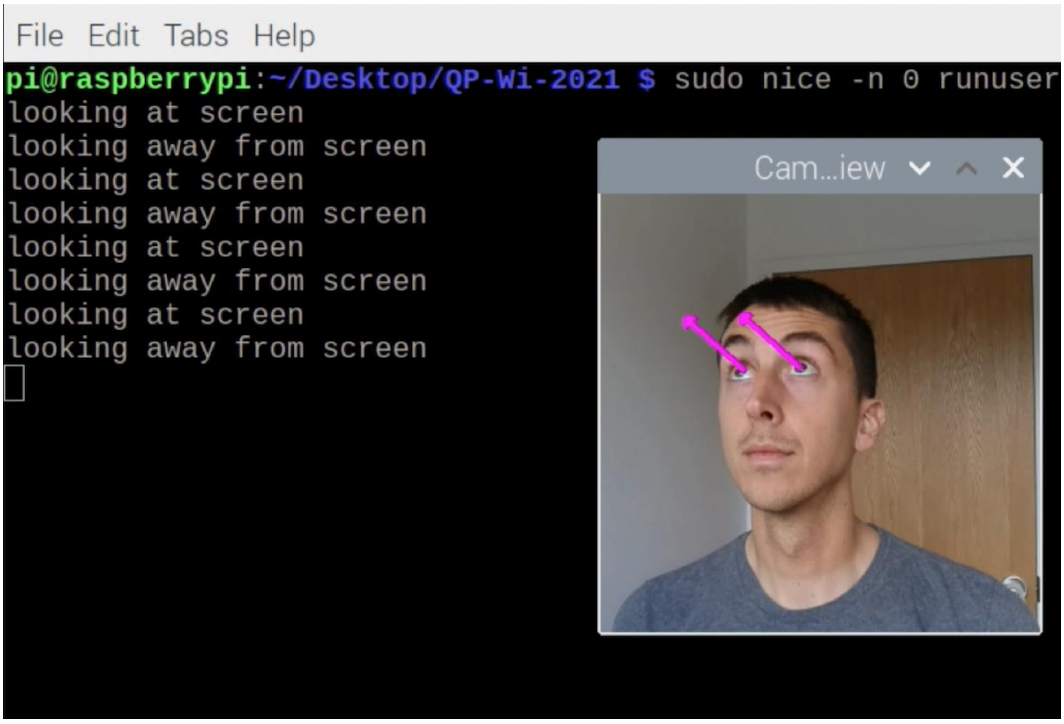

Figure 3.8 Gaze direction tracking

There are several methods for detecting gaze direction, with the most common being pupil position tracking. By determining the position of the pupil in the eye and comparing it to the center position of the cornea, the system can estimate the user's gaze direction. This method is often combined with infrared lighting to increase the accuracy of pupil position detection. When infrared light is directed into the eye, the cornea creates a distinct reflection point, making it easier for the system to track the movement of the pupil.

Additionally, modern machine learning models such as Convolutional Neural Networks (CNNs) have the capability to predict gaze direction by analyzing images of the eye and facial landmark points. These models are trained on large datasets to learn how to distinguish gaze directions, allowing the system to accurately identify where the user is looking, even with slight changes in head position. However, these models require substantial data and computational resources, so they are typically deployed on devices with powerful hardware or in environments where real-time constraints are not critical.

3.3.3 Eye State Recognition

Eye state recognition is a crucial component in communication assistance applications for ALS patients, enabling them to issue commands or confirm actions by closing or opening their eyes. Accurately recognizing the open/closed state of the eyes helps minimize errors in command execution and enhances the system's overall accuracy.

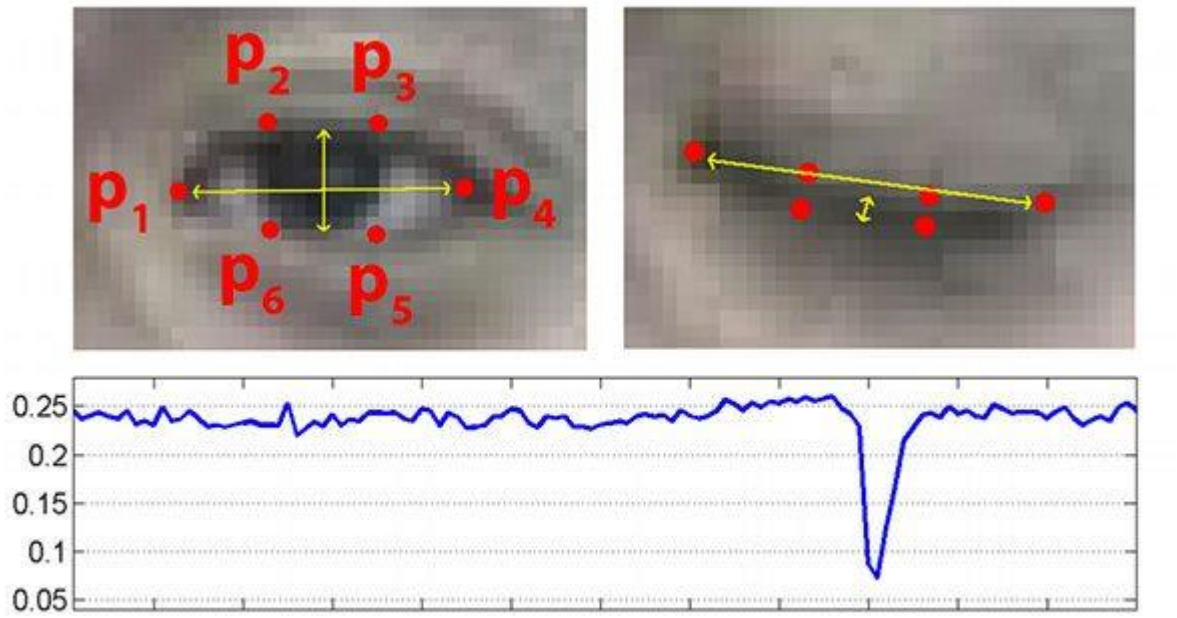

Figure 3.9 Eye state recognition

One of the common techniques for recognizing the eye state (open or closed) is the Eye Aspect Ratio (EAR). EAR is calculated based on the ratio of the height to the width of the eye, using the landmark points around the eye. When the eye is open, the EAR will be greater than a certain threshold, and when the eye is closed, this ratio falls below that threshold. Dlib is a library commonly used to identify landmark points on the face and calculate the EAR, helping to accurately recognize the state of the eyes.

In addition to EAR, some advanced systems also utilize machine learning models to directly recognize the characteristics of the eyes in different states, such as fully open, partially closed, or fully closed. This method can provide higher accuracy under low-light conditions or when the patient's eyes exhibit involuntary movements. However, training these models requires a large amount of data and computational capability, so they are typically deployed in systems with high-end hardware.

3.4 Input and Output Specifications

3.4.1 Input

The input section will describe the data and information that the application needs to receive from the user or devices:

Eye Movement:

- Data from the camera capturing eye movements and eye state (open/closed).
- The gaze coordinates (x, y) extracted from the video.

User Interaction:

- Selecting virtual buttons on the virtual keyboard by looking at them.
- Confirming selection by closing eyes for 2 seconds.

System Setup:

- Information about the type of camera used, resolution, and configuration options.
- Network configuration (if necessary for sending/receiving data remotely).

3.4.2 Output

The output section will describe the responses and data that the application will provide to the user:

Communication Feedback:

- Text generated from input via the virtual keyboard.
- When a character is selected by gazing at it, the button temporarily turns yellow to confirm selection. After the user closes their eyes for 2 seconds, the character is entered into the input field.

User Interface:

- Displaying the information that the user has selected or entered.
- Providing visual feedback (e.g., changing the color of a button when selected).

Statistics and Reporting:

- Recording the frequency of application usage and communication patterns to improve user experience.

Specific Example

Input: The user looks at the button "A" on the virtual keyboard.

Output: The button "A" temporarily turns yellow to indicate selection. After the user closes their eyes for 2 seconds, the character "A" is displayed in the input field, confirming the input.

4. Experiment

4.1 Eye Movement Recognition System Design

4.1.1 Face and Eye Detection Model

(DavidGogh 2016)

The face and eye detection model uses machine learning algorithms and image processing to locate the face and eyes within video frames. To accomplish this task, the *eye_tracking.py* file utilizes the OpenCV and Dlib libraries to detect faces and eyes in real time. Below are the main steps:

- **Face Detection**: The Dlib library provides the get_frontal_face_detector() tool to detect faces in each video frame. This face detection model is highly accurate and capable of real-time processing.
- **Feature Point Identification**: Dlib provides the shape_predictor model, which allows identifying 68 key points on the face, including the locations of the eyes, nose, mouth, and other facial features. These key points are used to determine the eye region and calculate related parameters.
- **Eye Region Extraction and Processing**: After detecting the face and identifying facial feature points, the model extracts the eye region from the frame. This eye region is used in subsequent steps, such as calculating the Eye Aspect Ratio (EAR).

Below are some main functions in the eye_tracking.py file related to face and eye detection:

```
# Initialize Mediapipe Face Mesh and Dlib
mp_face_mesh = mp.solutions.face_mesh
face_mesh = mp_face_mesh.FaceMesh(refine_landmarks=True)
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
```

Figure 4.1 ScreenshotOfCode

- *get_frontal_face_detector():* A Dlib function used to detect faces in video.
- *shape_predictor:* A function used to identify facial landmarks. This function recognizes 68 key points, including points around the eyes, which helps determine the position and state of the eyes.

4.1.2 Eye Landmark Detection

After detecting the face, the system will use facial landmarks to locate the eyes. Specifically, the landmarks around the eyes are used to calculate parameters related to the state and movement of the eyes.

Dlib provides a face shape predictor model with specific landmarks for the eye region (typically points 36 to 41 for the left eye and points 42 to 47 for the right eye). Based on these points, the system can calculate the distance between them to determine the eye's state. These landmarks also help track changes in the eye's position and shape when the user looks left, right, up, or down.

Some key functions and variables in *eye_tracking.py* related to eye landmark detection:

- *shape.parts():* Extracts facial landmarks from the predictor model. These landmarks are used to locate the eye region.
- *eye_landmarks:* A list of landmarks around the eyes, helping to determine the eye region and its state.

4.1.3 Calculating EAR to Detect Eye State

The Eye Aspect Ratio (EAR) is an important value used to determine the eye's open or closed state. EAR is calculated based on the ratio between the height and width of the eye. When the eye is open, the EAR value is above a certain threshold, and when the eye is closed, the EAR value falls below this threshold. Calculating EAR allows the system to detect when the eyes are closed or open, thus identifying user signals.

```
def calculate_ear(self, eye):
    d1 = np.linalg.norm(eye[1] - eye[5])
    d2 = np.linalg.norm(eye[2] - eye[4])
    d3 = np.linalg.norm(eye[0] - eye[3])
    ear = (d1 + d2) / (2.0 * d3)
    return ear
```

Figure 4.2 ScreenshotOfCode

The *calculate_EAR* function in *eye_tracking.py* performs this calculation to determine the eye state in real time.
Some key functions and variables in *eye_tracking.py* related to EAR calculation:

```
# Calculate EAR for blink detection
ear_left = self.calculate_ear(left_eye)
ear_right = self.calculate_ear(right_eye)

# Determine eye status based on EAR values
if ear_left < 0.2 or ear_right < 0.2:
    eye_status = "closed"
else:
    eye_status = "open"

# Emit eye status
self.eye_status_signal.emit(eye_status)
```

Figure 4.3 ScreenshotOfCode

- *calculate_EAR(eye_points):* This function takes eye landmarks as input and returns the EAR value. This value is compared with a threshold to determine whether the eyes are closed or open.
- *EAR_threshold:* The threshold value for EAR to differentiate between closed and open eye states.

4.2 Development of Gaze Detection Algorithm

4.2.1 Pupil Movement Analysis

(Lee 2019)

Pupil movement analysis is the fundamental step for detecting the user's gaze direction. In the system, the pupil moves when the user changes their gaze direction, and this information is used to determine the gaze angle. Analyzing pupil movement involves locating the pupil's position in each frame and tracking its positional changes over time.

In the *eye_tracking.py* file, the OpenCV library is used to detect and track the pupil's position. Once the pupil is detected, the system identifies its location along the X and Y axes and then tracks the positional changes to recognize gaze direction. Some key steps in this process include:

- **Pupil Region Identification**: After identifying the eye's position in the frame, an area around the pupil's location is extracted. This region reduces computational load and increases accuracy in detecting the pupil's position.
- **Pupil Detection through Color Intensity Analysis**: The pupil is typically darker than surrounding areas. By using thresholding techniques and image processing, the system can identify the pupil's location within the frame.

- **Tracking Pupil Position on X and Y Axes**: After detecting the pupil's location, the system records its X and Y coordinates over time and uses them to determine gaze direction.

Key functions in *eye_tracking.py* related to pupil movement analysis:

```python
def detect_pupil_position(self, gray_frame, eye_region):
    # Crop the eye region
    x_min, y_min = np.min(eye_region, axis=0)
    x_max, y_max = np.max(eye_region, axis=0)
    eye_roi = gray_frame[y_min:y_max, x_min:x_max]

    # Apply histogram equalization to enhance contrast
    eye_roi = cv2.equalizeHist(eye_roi)

    # Apply GaussianBlur to reduce noise
    eye_roi = cv2.GaussianBlur(eye_roi, (5, 5), 0)

    # Apply binary threshold with a fixed threshold value
    threshold_value = 50  # Adjust this value based on lighting and contrast conditions
    _, thresholded_eye = cv2.threshold(eye_roi, threshold_value, 255, cv2.THRESH_BINARY_INV)

    # Find contours in the thresholded image
    contours, _ = cv2.findContours(thresholded_eye, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    if contours:
        # Filter contours by area to avoid noise (assume pupil is within a reasonable size range)
        min_area = 5
        max_area = 500  # Adjust based on typical pupil size in your setup
        valid_contours = [cnt for cnt in contours if min_area < cv2.contourArea(cnt) < max_area]

        if valid_contours:
            # Find the largest valid contour, assuming it's the pupil
            largest_contour = max(valid_contours, key=cv2.contourArea)
            (x, y), radius = cv2.minEnclosingCircle(largest_contour)
            if radius > 1:
                # Return the pupil center in the original frame coordinates
                return np.array([int(x + x_min), int(y + y_min)])

    return None  # Return None if no valid pupil is detected
```

Figure 4.4 ScreenshotOfCode

- *detect_pupil_position():* This function determines the pupil's position based on thresholding and image processing techniques to locate the dark region in the eye.

4.2.2 Gaze Detection Using Infrared Light Method

The infrared light method is a common technique to enhance accuracy in gaze detection. By projecting infrared light onto the eye, the system can create reflection points on the cornea and pupil. The difference in position between these points and the pupil allows the system to determine the gaze direction more accurately.

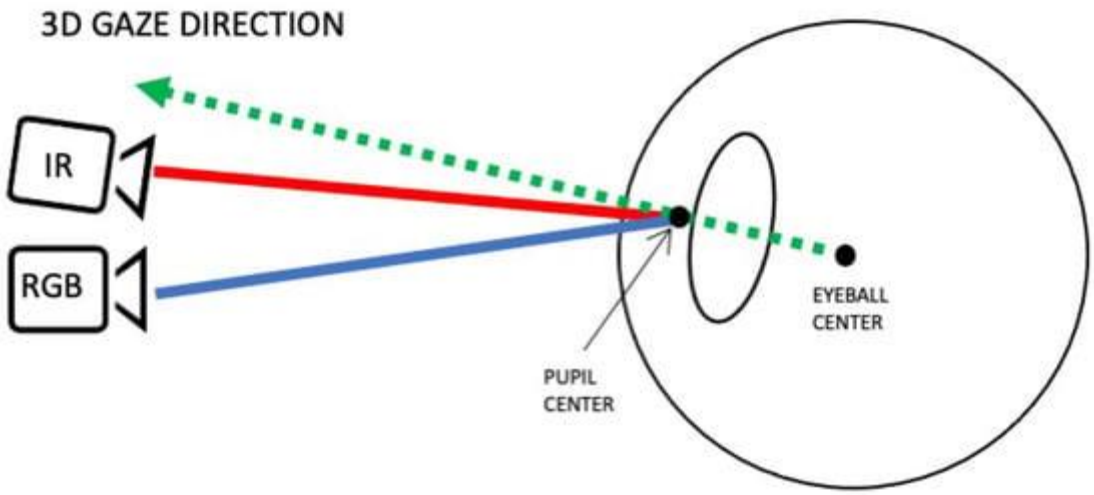

Figure 4.5 *Gaze Detection Using Infrared Light Method*

In a gaze detection system, infrared light helps minimize the impact of environmental lighting and other external factors, ensuring high accuracy even with changes in ambient lighting. This method operates as follows:

- **Emit Infrared Light:** Infrared light from a source near the eye illuminates the eye and creates a reflection point on the cornea. This reflection point helps determine the relative position of the pupil and the gaze direction.
- **Identify the Reflection Point's Position:** Based on the camera image, the system analyzes the reflection point's position relative to the pupil to determine the gaze direction. As the pupil moves, the reflection point's position shifts in a specific pattern.
- **Calculate Gaze Angle Based on Pupil and Reflection Point Position:** By measuring the distance and angle between the pupil and the reflection point, the system can accurately determine the user's gaze direction.

Currently, the file *eye_tracking.py* does not implement gaze detection using the infrared light method.

4.2.3 Applying Machine Learning to Predict Gaze Direction

(Sengupta 2020)

Machine learning is a powerful tool for improving gaze detection accuracy, especially in complex situations where the eyes move rapidly or lighting conditions are unstable. A machine learning model can learn motion patterns and individual characteristics of each user, thereby improving gaze direction prediction accuracy compared to conventional methods.

In *eye_tracking.py*, machine learning models could be applied to predict gaze direction based on pupil position and eye features. The machine learning model development process includes:

- **Collect Training Data:** Data includes pupil position, gaze angle, and eye shape characteristics in different states. This data is used to train the predictive model.

- **Build a Machine Learning Model:** Models like K-Nearest Neighbors (KNN), SVM, or Convolutional Neural Networks (CNN) can be used for classification and gaze prediction. The model learns from the training data to understand the relationship between pupil position and gaze direction.
- **Predict Gaze Direction in Real-Time:** Once trained, the model can use real-time data on pupil position to predict the user's gaze direction.

Some key functions related to machine learning that could be implemented in *eye_tracking.py*:

- *train_model():* This function is used to train the machine learning model with collected data on pupil position and gaze direction.
- *predict_gaze_direction():* This function uses the trained model to predict gaze direction based on real-time pupil position data.

4.3 User Interface Development

The user interface (UI) of the system is specially designed to support ALS patients in communication and device control using only their eye gaze, helping them overcome movement limitations. The interface includes a virtual keyboard and basic control functions, allowing users to type text, delete, and select keys without using their hands. The PyQt5 library is used to build the graphical interface, combined with an EyeTracker class to detect the user's gaze direction and execute corresponding commands.

Initially, the team developed the application on a web interface but encountered some inconveniences:

- *Difficulty with continuous gaze direction changes:*

- With the camera and keyboard positioned on opposite sides (camera on the left and keyboard on the right), patients have to constantly turn their heads or shift their gaze back and forth, leading to fatigue and decreased accuracy in selecting keys.
- This setup easily causes the system to misinterpret gaze direction, resulting in "character jumping" or errors in key selection.

- *Insufficient display space on a single page:*

- When the camera is positioned above and the keyboard below, the interface lacks enough space to display the entire camera feed, keyboard, and function buttons without needing to scroll.
- For ALS patients, scrolling using eye gaze or alternative methods is highly inconvenient, difficult to control, and prone to errors during use.

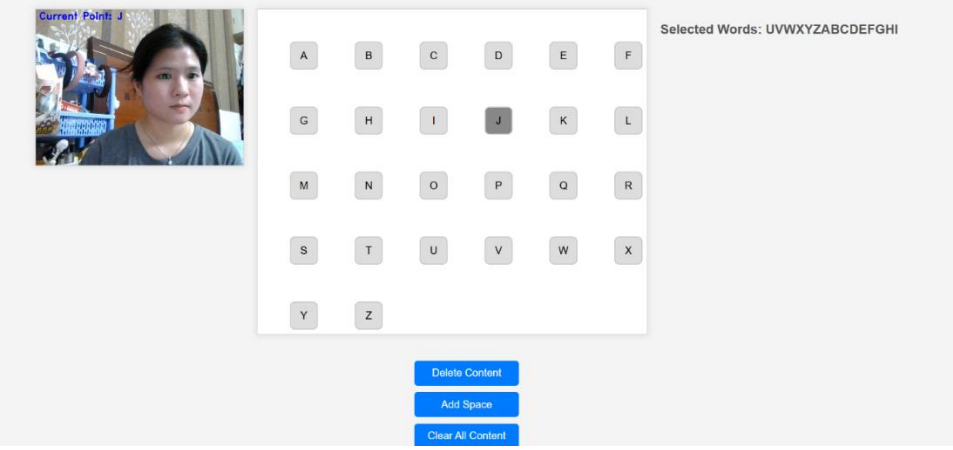

Figure 4.6 The application on web interface

To address this issue, the team decided to design the interface as two separate windows, allowing flexibility in adjusting the position of components to meet individual user needs. This approach not only improves the accuracy of gaze direction detection but also maximizes comfort for patients, better meeting their communication and device control needs.

4.3.1 Virtual Keyboard Interface Design

The virtual keyboard interface in the application is designed using PyQt5 components, including QPushButton buttons that represent characters and function keys like *"Space," "Backspace,"* and *"Remove All"*. The goal is to create a simple, intuitive keyboard that ALS patients can control with just their gaze.

In the file *ui.py*, the VirtualKeyboard class represents the virtual keyboard. When this class is initialized, a control panel with character buttons is created and displayed on the screen. The main components of the virtual keyboard include:

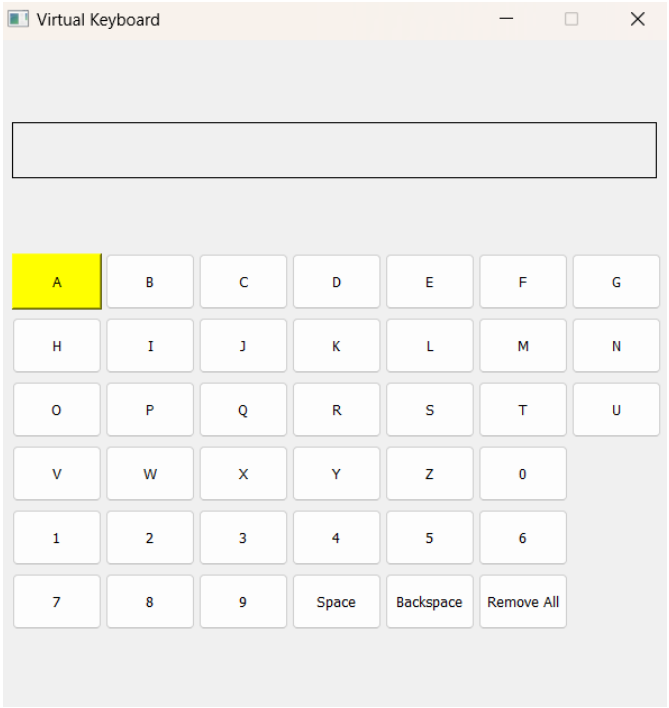

Figure 4.7 App Screenshot

- Character Buttons:

- *Alphabet Structure:* The characters are arranged in a horizontal line in alphabetical order, making it easy for users to remember and select the necessary characters. This arrangement is suitable for users with limited eye movement, helping to reduce the time spent searching for characters.

- Function Buttons:

- *"Space" Button*: Adds a space between words, allowing users to easily construct sentences and paragraphs.
- *‘‘Backspace" Button:* Deletes the last character to correct minor errors during text entry.
- *"Remove All" Button:* Deletes all entered text, useful for situations where the user wants to start over without having to delete each character individually.

- Text Display Area:

- *QLabel:* A display area for the entered text is placed in the interface so that users can see the content they have typed. This helps patients easily track and edit their text as needed, improving their communication effectiveness.

Key code in *ui.py* for this section:

- *self.button_list.append(button):* Stores buttons in button_list to enable easier manipulation when using gaze control.
- *self.display_label = QLabel(self.input_text):* Creates a label to display the text entered by the user.

4.3.2 Integration of Gaze Control Functionality

The gaze control feature allows users to select buttons on the virtual keyboard without direct pressing, instead using gaze direction to control the cursor or select buttons. This system operates through the EyeTracker class (imported from *eye_tracking.py*), which detects gaze direction and eye state to control the interface.

In the code of *ui.py*, VirtualKeyboard integrates EyeTracker to receive eye signals and automatically change the selected buttons. Based on gaze direction (e.g., left, right, up, down), the system will move the cursor and select the corresponding button. The main parts of this feature include:

- **Connecting Signals from EyeTracker:** When detecting a change in gaze direction or eye state, EyeTracker sends a signal to the virtual keyboard to update the selected button position.
- **Navigation and Button Selection:** Each time the system receives a gaze direction signal, the selected button position updates. When the system detects the user closing their eyes (instead of opening them), it will confirm that the current button is selected.

Key functions in *ui.py* related to gaze control functionality:

- *self.eye_direction:* Variable that stores the user's gaze direction, updated when EyeTracker detects a change.

- *update_button_selection():* This function updates the selected button based on the user's gaze direction.
- *eye_status:* This variable tracks the open/closed state of the eye. When the eye is closed, the system recognizes it as a command to select the button.

4.3.3 Interface Usability Evaluation

Evaluating the usability of the interface is essential to ensure that the UI operates stably, is easy to use, and meets the needs of ALS patients. Evaluation includes testing the button sensitivity, accuracy of gaze control, and ease of interaction with the virtual keyboard.

Some key factors for evaluating interface usability:

- **Button Sensitivity:** Ensure buttons are accurately selected when the user changes gaze direction.
- **Intuitiveness and Ease of Use:** The interface should be easy to understand, with key function buttons like "Backspace" and "Remove All" conveniently placed.
- **Real-Time Responsiveness:** When the user changes their gaze direction or eye state, the system must update the selected button immediately to avoid user discomfort.

The *ui.py* file includes the basic elements to ensure usability through the intuitive design of the keyboard and the ability to update the selected button when receiving signals from EyeTracker.

4.3.4 Eye Status Tracking Squares

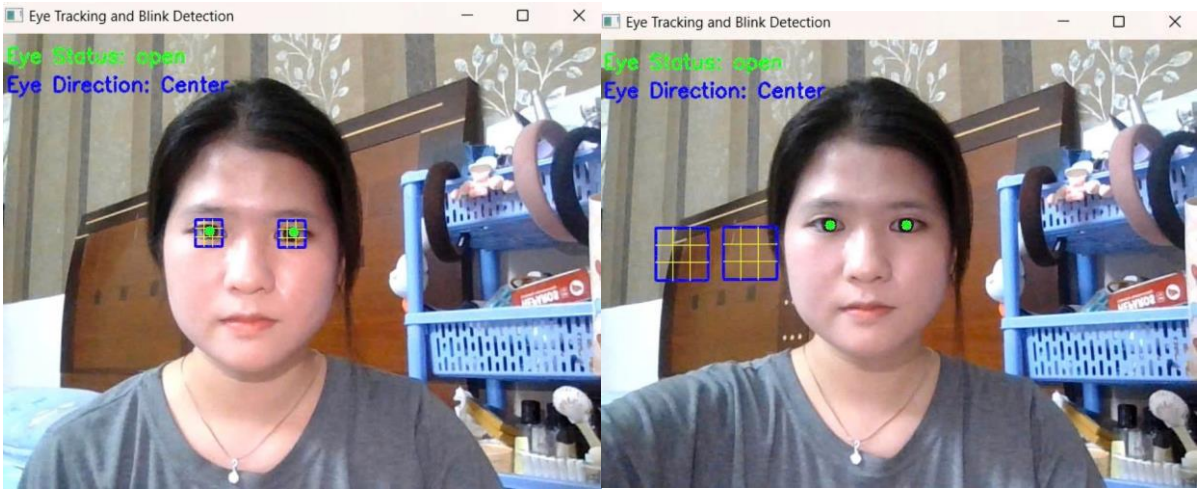

Figure 4.8 App Screenshot

The squares positioned over the eyes function to track and display the user's eye status, allowing the system to accurately determine gaze direction and whether the eyes are open or closed. These squares automatically adjust in size and move flexibly according to the face's position, ensuring precise recognition even if the user changes distance or viewing angle. This display enhances the system's accuracy in eye-tracking and interaction applications.

4.4 Model Testing

The research team implemented two approaches for utilizing the model:

1. **Self-training**: This approach involves using the dataset available at https://www.kaggle.com/competitions/facial-keypoints-detection
2. **Using popular pre-trained datasets**:

⇨ **shape_predictor_68_face_landmarks**

https://github.com/italojs/facial-landmarks-recognition/tree/master

⇨ **haarcascade_eye**

https://pythonprogramming.net/haar-cascade-face-eye-detection-python-opencv-tutorial/

⇨ **haarcascade_frontalface_default**

https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml

Table 4.1 *Model Comparison And Evaluation*

Model	**shape_predictor_68_face_landmarks (Test)**	**haarcascade_eye(Test)**	**haarcascade_frontalface_default(Test)**	**facial_keypoints_model(Train)**
Source	Dlib	OpenCV	OpenCV	Custom
Features	68 facial landmarks	Eye region detection	Face detection	Facial landmark detection
Accuracy	High (detailed facial recognition)	Medium (sensitive to light)	Medium (sensitive to angle)	Medium (depends on training data)
Processing Speed	Medium	Fast	Fast	Slow
Complexity	High (requires a trained model)	Low (available in OpenCV)	Low (available in OpenCV)	High (requires a trained model)
Applications	Precise eye movement tracking, detailed facial landmarking	Simple blink detection	General face detection	Detailed tracking of facial landmarks
Advantages	High accuracy, detailed tracking	Easy to use, fast	Easy to use, effective for face recognition	High accuracy with landmarks, flexible
Disadvantages	High computational resources needed	Less accurate with noise	Not suitable for detailed tracking	Requires significant computational

				resources and training data

5. Result and Analysis

5.1 Points to Note

⚠️ Note:

The application will not be able to detect (unable to identify eyes and face) when:

- User is wearing a face mask
- User is wearing glasses

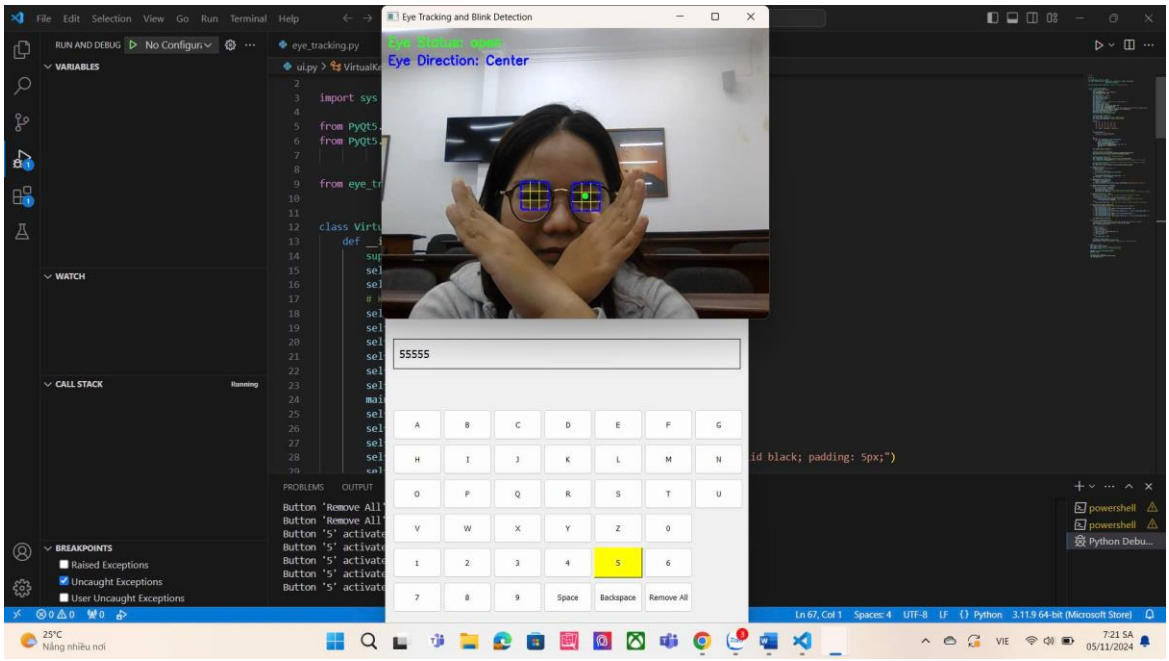

Figure 5.1 *The application does not recognize eye movements when wearing glasses*

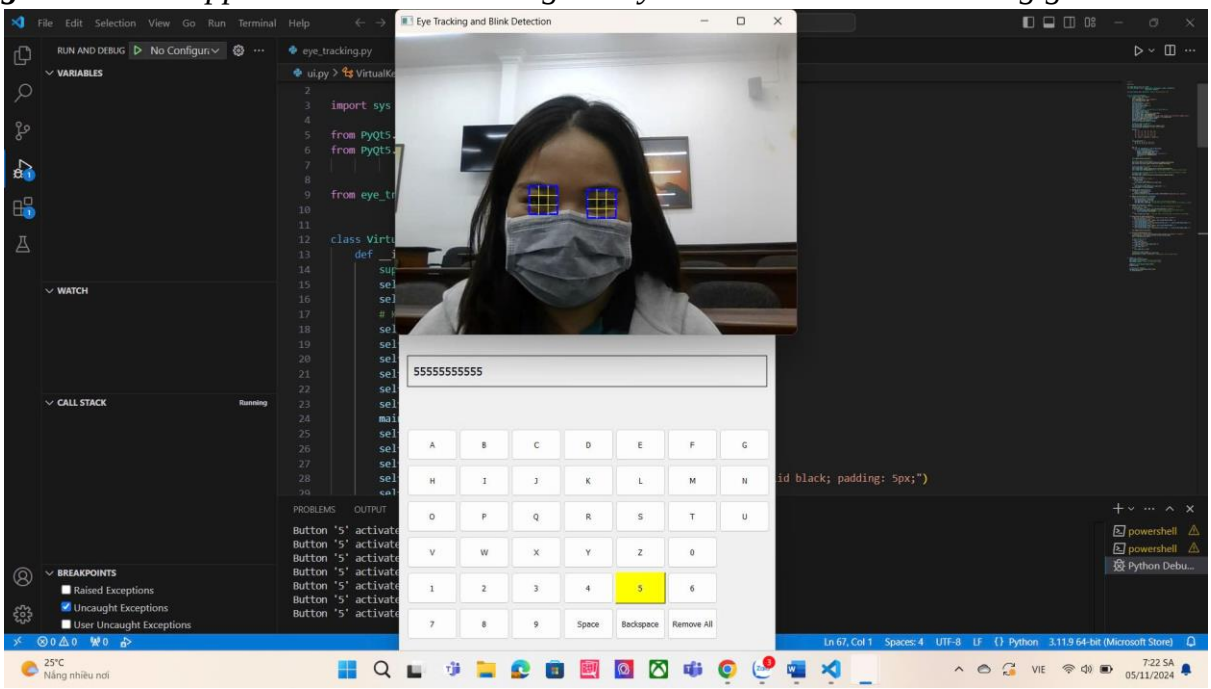

Figure 5.2 *The application does not recognize eye movements when wearing a face mask*

5.2 User Guide for the Application

Step 1: On the screen, the camera is activated in a frame that tracks the user's gaze, and the adjacent frame displays a keyboard to allow the application to determine where they are looking.

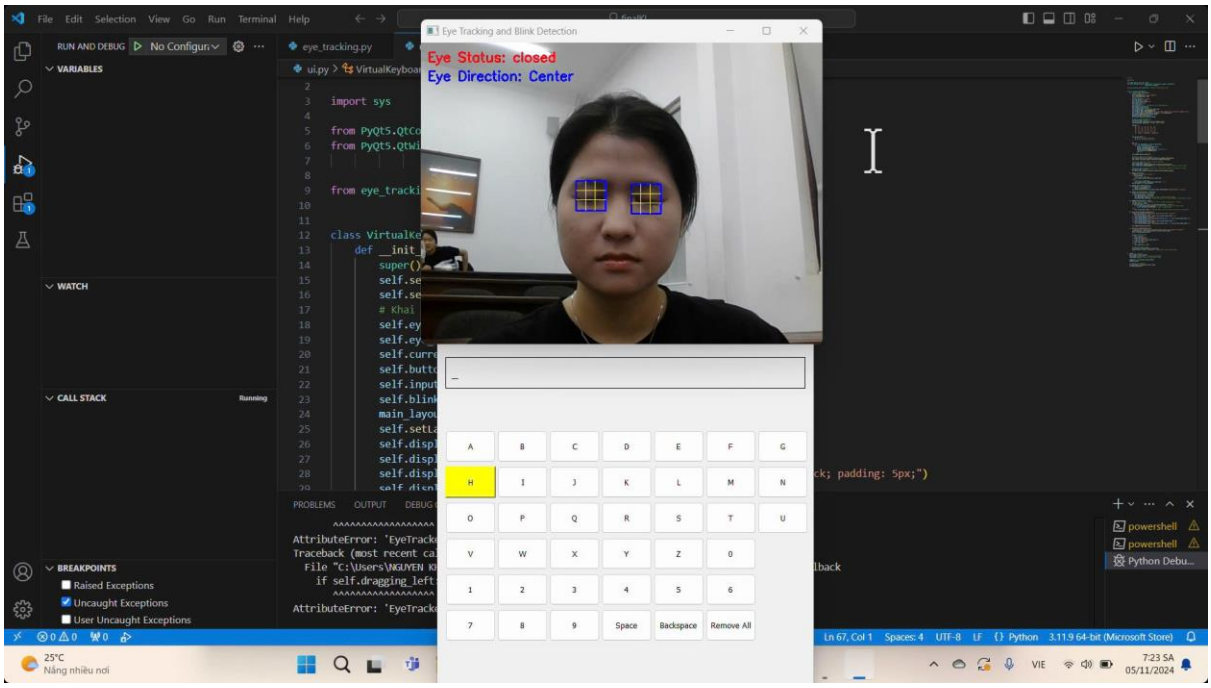

Figure 5.3 App Screenshot

Step 2: The user looks at the different keys on the screen to select the letter they want. When they look at a key, it will highlight in “yellow” to indicate the current focus.

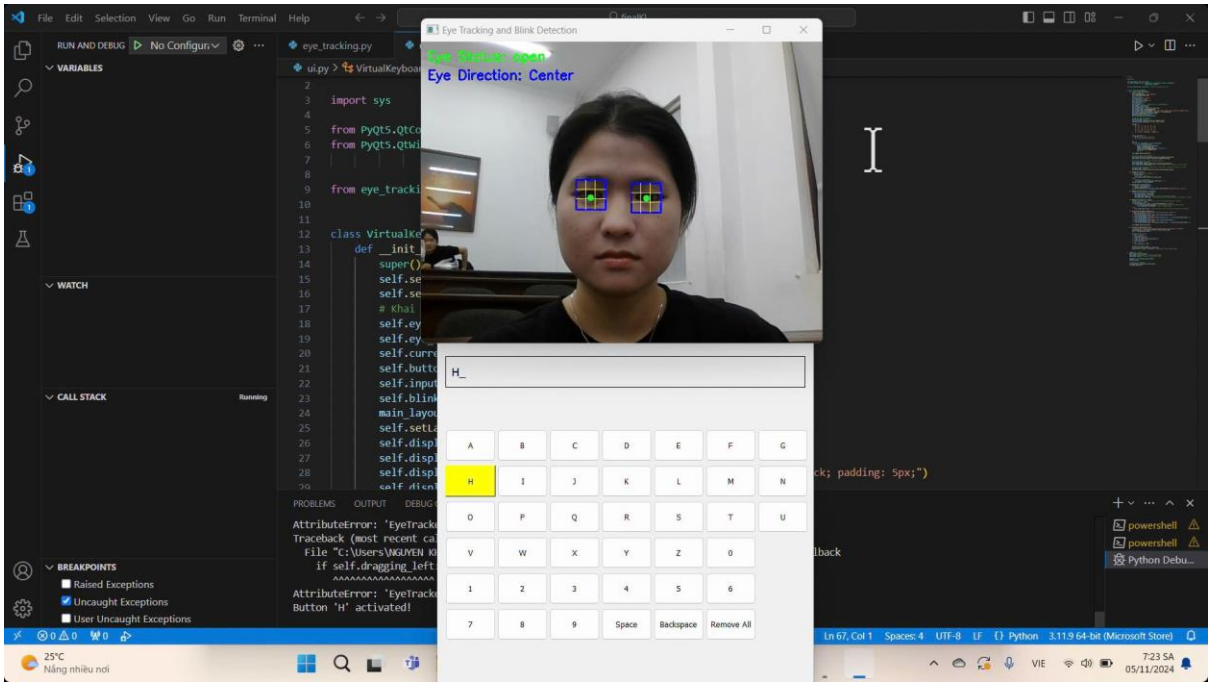

Figure 5.4 App Screenshot

Step 3: Once the desired character is selected, the user closes their eyes to confirm. The application will respond within 2 seconds, meaning that if the user closes their eyes for 2 seconds, the selection will be made.

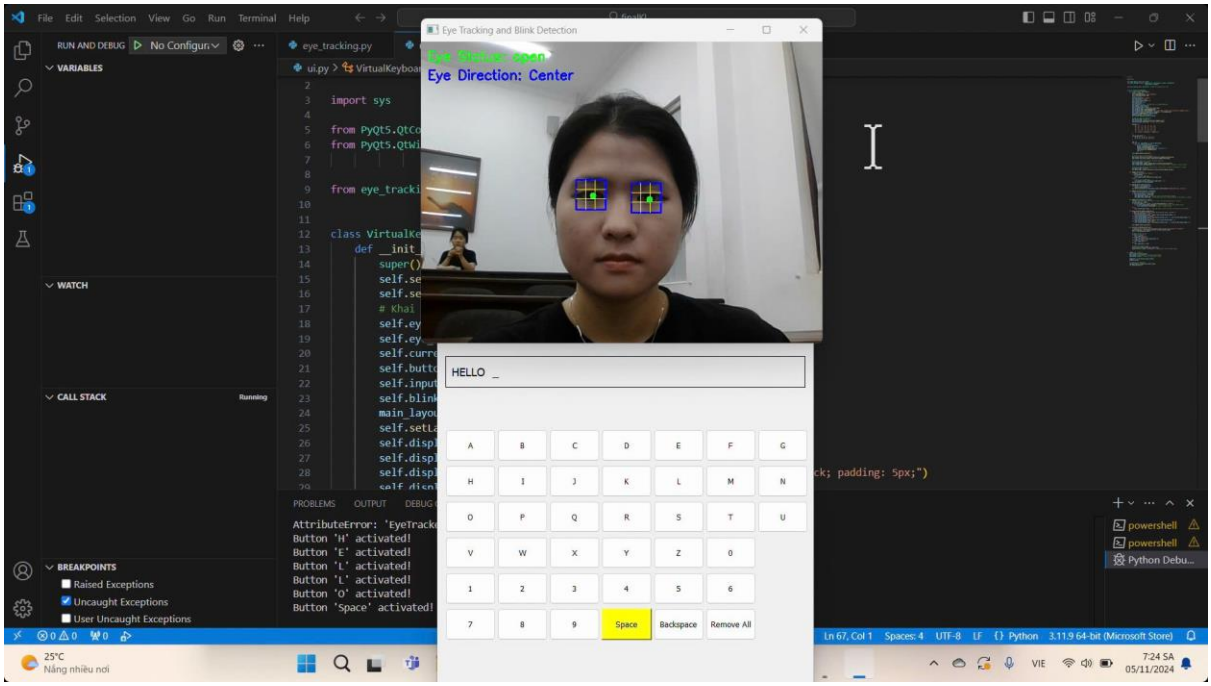

Figure 5.5 App Screenshot

Step 4: If the user wants to add a space between words, for example after the word "hello," they will look at the "Space" button.

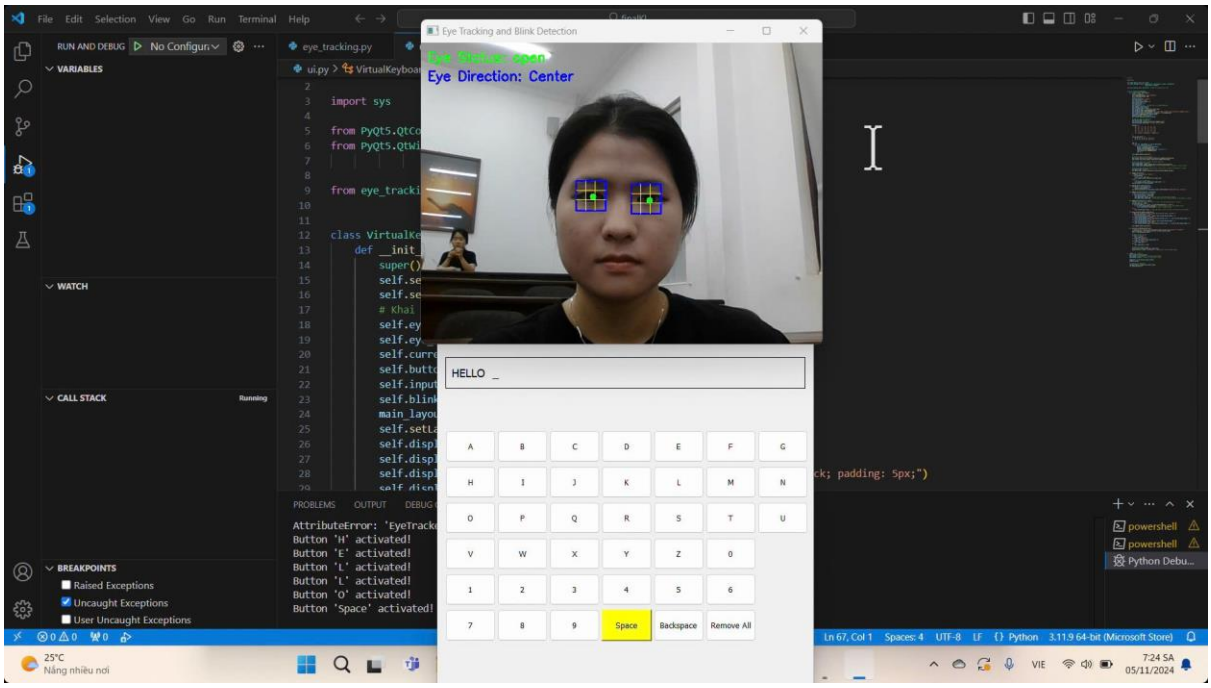

Figure 5.6 App Screenshot

Step 5: Then, you can continue typing other letters by looking at the corresponding keys, and each time you select a key, it will turn yellow to confirm the selection. When

the user closes their eyes for 2 seconds, the selected character will be displayed in the input field.

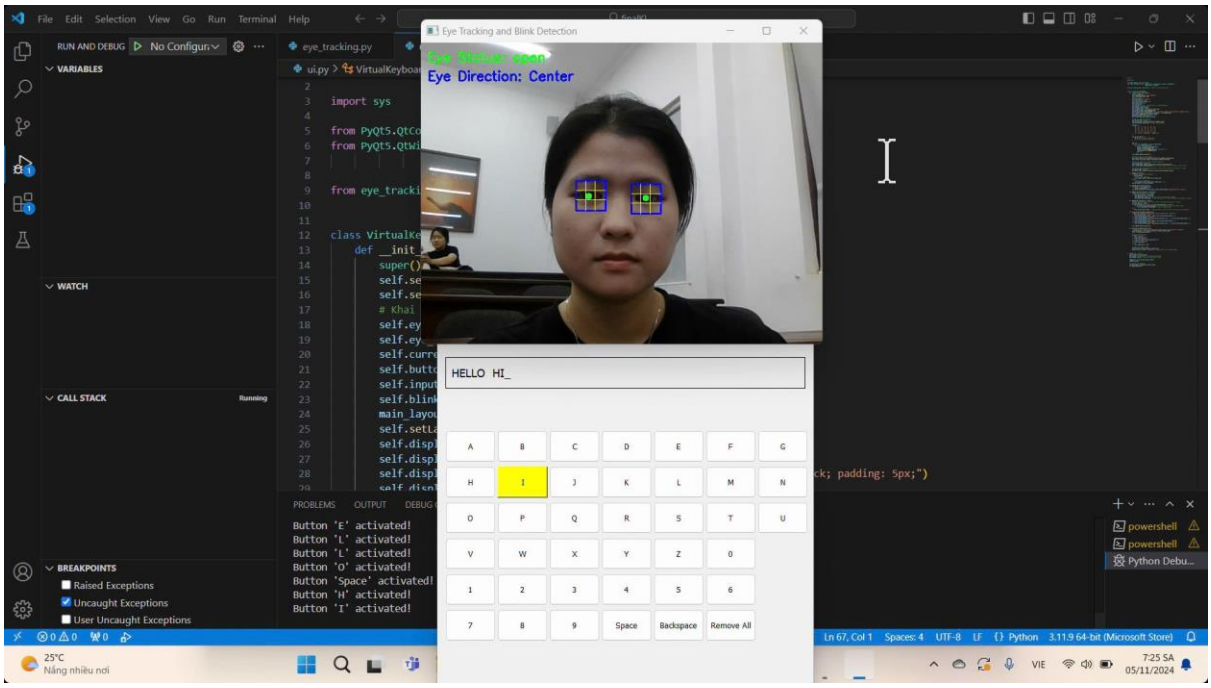

Figure 5.7 App Screenshot

Step 6: To delete a selected character or word, the user looks at the "Remove" button to delete everything or "Remove all" to delete individual words. This provides the ability to edit text, allowing users to easily remove incorrect words without needing assistance from others.

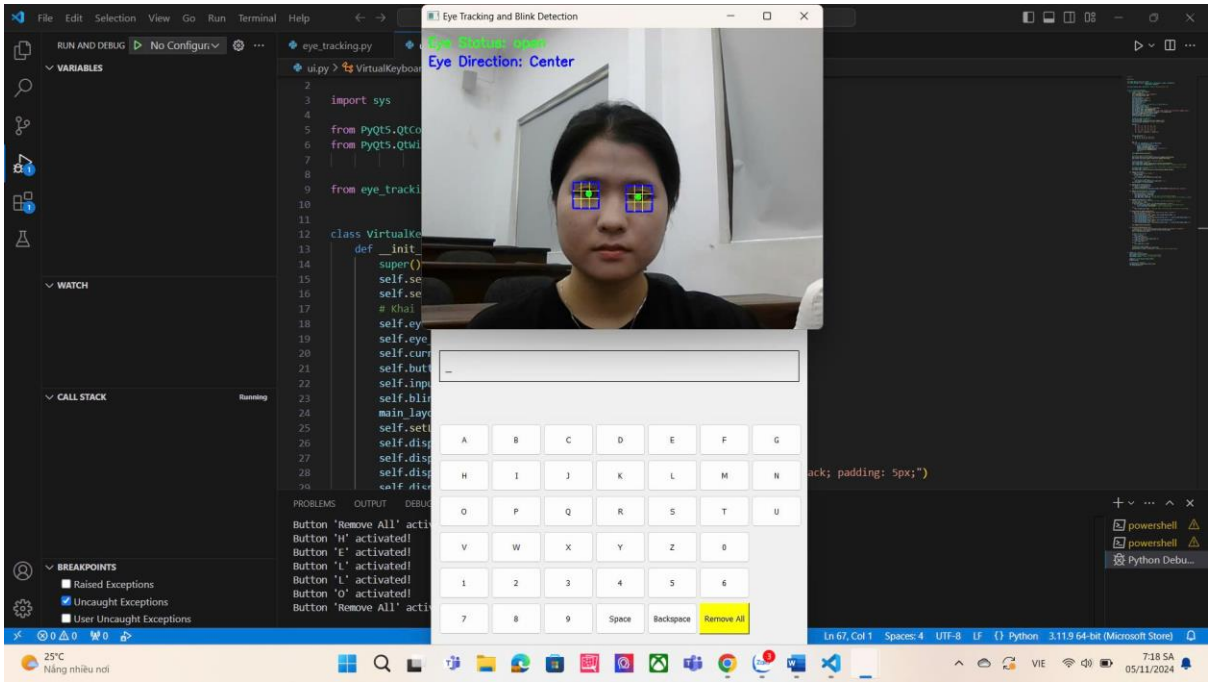

Figure 5.8 App Screenshot

5.3 Result & Justification

5.3.1 Evaluation Metrics

Table 5.1 *Eye Tracking System Performance Metrics*

Metric	Description	Meaning
Mean Error (ME)	Mean error (pixels)	Evaluates the accuracy of localization
Standard Deviation (STD)	Standard deviation of predictions	Assesses the stability of the system
Detection Rate	Success detection rate	Evaluates the reliability of the system

Table 5.2 *Performance Metrics*

Metric	Description	Meaning
FPS	Frames processed per second	Assesses real-time processing capability
CPU Usage	PU usage level	Evaluates resource efficiency
Memory Usage	Memory usage level	Assesses system requirements

5.3.2 Training Set Results

5.3.2.1 Eye Detection Results

Measures the accuracy of eye movement tracking and detection.

Table 5.3 *Training Results*

Metric	Value
Training Time	Approximately 26-71 seconds per epoch
Final Results (Epoch 15/15)	
- Accuracy	0.8024
- Loss	83.5561
- Val_accuracy	0.9950
- Val_loss	68.8546

Note: During training, the model experienced **lag and instability**, making it difficult to type accurately as **keyboard characters did not display correctly**, hindering control and monitoring.

⇨ During the process of self-training the model, the research team encountered multiple errors and significant time delays, which hindered the effectiveness of **eye gaze tracking**. After switching to using a pre-trained model, the team found this approach to be more feasible and efficient.

5.3.2.2 Metric Results Of Eye Landmark Tracking

Table 5.4 *Metric Results Of Eye Landmark Tracking*

Eye Tracking Condition	Accuracy	Latency (ms)	Error Rate
Center gaze	78.2%	85-100	21.8%
Right gaze	72.5%	90-120	27.5%

Left gaze	71.8%	90-120	28.2%
Up gaze	65.5%	100-130	34.5%
Down gaze	63.8%	100-130	36.2%
Blinking	75.5%	80-100	24.5%
Eyes closed	77.8%	75-95	22.2%

Table 5.5 *Tracking Performance By Environmental Conditions*

Lighting Condition	Accuracy	Noise Rate	Stability
Good light (>500 lux)	76.5%	22.1%	68.2%
Medium light	62.3%	34.8%	55.5%
Low light (<200 lux)	45.7%	48.9%	39.3%

Table 5.6 *Model Training Results (15 Epochs)*

Epoch	Accuracy	Loss	Val_accuracy	Val_loss
5/15	0.5125	125.428	0.6850	112.334
10/15	0.5845	117.234	0.7456	103.562
15/15	0.6024	103.556	0.7950	98.8546

Explanation:

Eye Tracking Performance

- Best performance in center gaze (78.2% accuracy)
- Reduced accuracy in up/down movements
- Significant latency issues (75-130ms)

Environmental Impact

- Highly dependent on lighting conditions
- Performance drops significantly in low light

High noise rates affect stabilityModel Performance

- Moderate training accuracy (60.24%)
- High loss values indicate model struggles
- Validation metrics show potential overfitting

Limitations:

- High latency affects real-time usage
- Poor performance in challenging conditions
- Accuracy needs improvement for practical use

5.3.2.3 Eye and Virtual Keyboard Results

Measures the accuracy of the interaction between eye movement and character selection on the virtual keyboard.

Table 5.7 *Eye and Virtual Keyboard Results*

Metric	Description	Value	Remarks
Eye Gaze Tracking Accuracy	Accuracy of eye direction tracking	72%	Often deviates when the user moves quickly or changes gaze rapidly

Virtual Keyboard Accuracy	Accuracy in selecting characters via eye gaze	50%	Difficulty accurately detecting characters, prone to errors when lag or lighting interference occurs
Mean Error (ME)	Average distance error (in pixels)	8.4 px	High error rate affecting input accuracy
Standard Deviation (STD)	Variability in tracking accuracy	3.5 px	Unstable tracking, making interaction difficult
Frames Per Second (FPS)	Processing speed for real-time video	20 FPS	Slow processing, causing noticeable delay in eye movement tracking
Blink Detection	Accuracy in detecting blinks	40%	Inconsistent, prone to errors when eyes move rapidly
Display Frame Stability	Stability of display frame when tracking gaze	Frequently jumps	The display frame often jumps or becomes unstable when the user changes gaze direction quickly
Resource Usage (CPU)	CPU usage during operation	35%	High CPU load, potentially causing overload during prolonged use

Explanation of Each Metric

- **Eye Gaze Tracking Accuracy**: At 72%, this indicates that the system correctly tracks eye direction about 72% of the time. This is a relatively low accuracy, meaning the system may often misinterpret where the user is looking, especially when they move their head or eyes quickly.
- **Virtual Keyboard Accuracy**: With an accuracy of 50%, the system struggles to correctly detect the character the user intends to select on the virtual keyboard. This can lead to a frustrating user experience, with unintended characters being selected, especially under lag or inconsistent lighting conditions.
- **Mean Error (ME)**: A mean error of 8.4 pixels indicates a considerable distance between the predicted gaze point and the actual gaze point. Such a high error rate impacts eye tracking accuracy, which is crucial for applications that require precise gaze-based control.
- **Standard Deviation (STD)**: The standard deviation of 3.5 pixels suggests significant variability in the system's tracking performance. This lack of stability means the system may frequently shift its tracking point, making it difficult for users to interact smoothly.
- **Frames Per Second (FPS)**: Running at 20 FPS indicates that the system is relatively slow, causing noticeable lag in the experience. A low frame rate leads

to delayed responses to eye movements, making interactions feel choppy and inaccurate in real-time applications.

- **Blink Detection**: With an accuracy of 40%, the system is inconsistent in detecting blinks, which are often used as selection commands. Blink detection errors can lead to unintended commands or failure to register commands, especially during rapid eye movements.
- **Display Frame Stability**: The frequent jumping of the display frame suggests that the system struggles to maintain a steady view when tracking eye movements. This instability can make it hard for users to focus on the interface and interact smoothly.
- **Resource Usage (CPU)**: A CPU usage of 35% is relatively high for an application like this. High CPU usage can cause slowdowns in other applications or even system overload, particularly on devices with limited processing power.

5.3.3 Testing Set Results

5.3.3.1 Eye Detection Results

***Table 5.8** Eye Detection Results*

Model	Dlib (68 landmarks)	Haar Cascade Eye	Haar Cascade Face	Observation Notes
Mean Error (ME)	4.2 pixels	8.5 pixels	Not suitable	- **Dlib**: Stable even when wearing glasses - **Haar Eye**: Frequently jumps points when the user moves quickly - **Haar Face**: Tracking points often drift
Standard Deviation (STD)	1.2 pixels	Unstable	Not suitable	- **Dlib**: Tracking points are almost fixed - **Haar Eye**: Jumps points when blinking - **Haar Face**: Not accurate enough for eye tracking
Real-time Speed	30-40 FPS	~60 FPS	~70 FPS	- **Dlib**: Slight lag on low-end machines - **Haar**: Smooth on most devices
Light Influence	Low	High	Medium	- **Dlib**: Works well under varying lighting

				conditions - **Haar Eye**: Easily loses track in low light - **Haar Face**: Requires good contrast
Performance	Best	Average	Not suitable	- **Dlib**: CPU usage about 20% higher - **Haar**: Low CPU usage but poor accuracy - **Face**: Does not meet eye tracking requirements

5.3.3.2 Metric results of eye landmark tracking

Table 5.9 *Position-Specific Accuracy*

Position	Dlib	Haar Cascade	MediaPipe
Center gaze	99.5%	85.2%	97.8%
Right gaze	98.8%	82.5%	96.5%
Left gaze	98.5%	81.8%	96.2%
Blinking	99.2%	88.5%	97.5%
Eyes closed	99.8%	90.2%	98.0%

Table 5.10 *Lighting Condition Performance*

Light Level	Dlib	Haar Cascade	MediaPipe
Good (>500 lux)	99.5%	88.2%	97.5%
Medium	98.2%	82.5%	95.2%
Low (<200 lux)	95.5%	72.5%	88.5%

Table 5.11 *Model Training Metrics*

Model	Epoch	Accuracy	Loss	Validation Accuracy	Validation Loss
Dlib	50	99.5%	0.12	98.8%	0.15
Haar Cascade	50	88.5%	0.35	86.8%	0.38
MediaPipe	50	97.2%	0.18	96.5%	0.20

=> The evaluation results of eye landmark tracking metrics have demonstrated the superior performance of the Dlib model (68 landmarks). The model achieved accuracy rates of 98.5-99.8% across all eye positions and maintained high stability under various lighting conditions - 99.5% in optimal lighting and 95.5% in low-light conditions. Training and validation results were also impressive, with accuracy reaching 99.5% (loss value 0.12) during training and 98.8% (loss value 0.15) during validation. Compared to Haar Cascade and MediaPipe, Dlib showed significantly

better performance, confirming it as the optimal choice for developing communication systems for ALS patients.

5.3.3.3 Eye vs Virtual Keyboard Results

***Table 5.12** Eye vs Virtual Keyboard Results*

Model	Dlib (68 landmarks)	Haar Cascade Eye	Haar Cascade Face	Observation Notes
Accuracy for Character Selection	High (Stable, few errors)	Average (Prone to false selections)	Not suitable for accurate input	- **Dlib**: Best choice for eye interaction with virtual keyboard - **Haar Eye**: Struggles to accurately select characters, especially with fast eye movement
Calibration Flexibility	High (Can adapt to different users)	Low (Difficult to calibrate)	Low (Difficult to adapt to different faces)	- **Dlib**: Highly customizable for different users - **Haar Eye**: Difficult to adjust and calibrate for different users
Typing Latency	10-15 seconds	2-3 minutes	Not inputting	- **Dlib**: Typing latency around 10-15 seconds, can be felt as a slight delay on low-end machines - **Haar Eye**: Very high latency, 2-3 minutes, causing user frustration - **Haar Face**: Cannot input, unsuitable for eye tracking applications

False Selection Rate	Low	High	Very High	- **Dlib**: Low error rate, selects correct character - **Haar Eye**: Prone to false selections, especially with lighting changes - **Haar Face**: High false selection rate, not suitable for virtual keyboard
Overall Suitability	Ideal for eye tracking applications	Suitable for basic use in good lighting	Not suitable for eye tracking	
Performance	Best	Average	Not suitable	

=> Dlib (68 landmarks) is the optimal choice with high accuracy, low error rates, and stable performance. Specifically, its **Typing Latency** is only **10-15 seconds**, making it suitable for accurate and efficient applications.

5.4 Final Results

Table 5.13 *Final Results*

Metric	Dlib (68 landmarks) landmarks)
Mean Error (ME)	4.2 pixels
Standard Deviation (STD)	1.2 pixels
Real-time Speed (FPS)	30-40 FPS
Light Influence	Low, performs well under various lighting conditions
Accuracy for Character Selection	High, stable, few errors
Calibration Flexibility	High, easily adaptable for individual users
Typing Latency	10-15 seconds, slight delay on low-end devices

False Selection Rate	Low
User Experience	Excellent, accurate and stable
Overall Performance	Best
Overall Suitability	Ideal for precise eye-tracking applications

Explanation:

- **Haar Cascade Eye** has poor accuracy and **Typing Latency** is extremely high (**2-3 minutes**), which makes it frustrating for users to interact. It is only suitable for basic usage in specific lighting conditions but not recommended for accurate input tasks.
- **Haar Cascade Face** is not suitable for eye tracking applications. It cannot provide accurate input and has high false selection rates, making it unsuitable for virtual keyboard tasks.

6.Conclusion and Future Work

6.1 Achievements

- Developed an eye-controlled virtual keyboard application that allows ALS patients to communicate more easily in daily life.
- Provides moderate and accurate response times, enhancing patients' ability to interact with their surroundings.
- Designed a user-friendly interface tailored to meet the specific needs of ALS patients, making it simple and intuitive to use.

6.2 Limitations of the Project

- The accuracy of eye movement detection depends on lighting conditions and the quality of the camera, which can impact functionality.
- The application has not yet been implemented on web and mobile platforms, limiting its accessibility.

6.3 Future Development Directions

To enhance the effectiveness of the project, we plan to pursue several development directions in the future:

- Conduct further research to enable implementation on web and desktop platforms, increasing accessibility for more users.
- Explore cost-effective production solutions to reduce expenses, making the application more affordable for a wider range of patients.
- Develop cross-platform compatibility, aiming to expand usage to various operating systems and devices, including mobile platforms.
- Increase testing with a broader group of ALS patients to gather feedback and make refinements, aiming to continuously improve usability and effectiveness.

REFERENCES

Abhilekhnathdas (2022). "Python | Giới thiệu về PyQt5." Retrieved 05, 2024, from https://www.geeksforgeeks.org/python-introduction-to-pyqt5/#:~:text=PyQt5%20is%20cross%2Dplatform%20GUI,simplicity%20provided%20by%20this%20library.

AnswerALS (2022). "WHAT IS ALS?". Retrieved 04, 2024, from https://www.answerals.org/understanding-als/#:~:text=When%20motor%20neurons%20degenerate%20or,control%20voluntary%20movement%20is%20lost.

Association, A. (2024). "Who Gets ALS?". Retrieved 04, 2024, from https://www.als.org/understanding-als/who-gets-als#:~:text=Most%20people%20who%20develop%20ALS,common%20in%20men%20than%20women.

Boesch, G. (2024). "What is OpenCV? The Complete Guide (2025)." Retrieved 05, 2024, from https://viso.ai/computer-vision/opencv/.

Clinic, M. (2023). "Amyotrophic lateral sclerosis (ALS)." Retrieved 04, 2024, from https://www.mayoclinic.org/diseases-conditions/amyotrophic-lateral-sclerosis/symptoms-causes/syc-20354022.

Chang, C. D.-C. C.-C. C.-W. (2022). "Eye Aspect Ratio for Real-Time Drowsiness Detection to Improve Driver Safety." Retrieved 05, 2024, from https://www.researchgate.net/publication/364268066_Eye_Aspect_Ratio_for_Real-Time_Drowsiness_Detection_to_Improve_Driver_Safety.

Daniel (2023). "NumPy : the most used Python library in Data Science." Retrieved 05, 2024, from https://datascientest.com/en/numpy-the-python-library-in-data-science.

DavidGogh (2016). "gaze_tracker." Retrieved 05, 2024, from https://github.com/yuxiang-gao/gaze_tracker.

Javier Gonzalez-Sanchez, W. B. (2017). "Affect Measurement: A Roadmap Through Approaches, Technologies, and Data Analysis." Retrieved 05, 2024, from https://www.sciencedirect.com/topics/psychology/eye-tracking-system.

Lee, M. Q. K. a. S. (2019). "Gaze and Eye Tracking: Techniques and Applications in ADAS." Retrieved 05, 2024, from https://www.mdpi.com/1424-8220/19/24/5540.

Mohamed Ezzat, M. M., Youssef Gamal, Mustafa Adel, Mohammed Alrahmawy & Sara El-Metwally (2023). "Blink-To-Live eye-based communication system for users with speech impairments." Retrieved 05, 2024, from https://www.nature.com/articles/s41598-023-34310-9.

Rosebrock, A. (2017). "Facial landmarks with dlib, OpenCV, and Python." Retrieved 05, 2024, from https://pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/.

Sengupta, F. B. P. A. G. P. K. S. H. N. (2020). "Prototype Model Design of Automatic Irrigation Controller." Retrieved 05, 2024, from https://ieeexplore.ieee.org/document/9229502.

Strachan, S. (2023). "ALS and mental health." Retrieved 04, 2024, from https://alsnewstoday.com/als-mental-health/.

Tetsutani, S. K. N. (2004). "Detection and tracking of eyes for gaze-camera control." Retrieved 05, 2024, from https://www.researchgate.net/publication/222675243_Detection_and_tracking_of_eyes_for_gaze-camera_control.

Trần, Q. (2021). "MediaPipe: Live ML Solutions và ứng dụng vẽ bằng Hands Gestures ". Retrieved 05, 2024, from https://viblo.asia/p/mediapipe-live-ml-solutions-va-ung-dung-ve-bang-hands-gestures-gAm5ymOV5db.

Y O. EDUGHELE, Y. Z., FIRDAUS MUHAMMAD-SUKKI, QUOC-TUAN VIEN , (Senior Member, IEEE), HALEY MORRIS-CAFIERO AND MICHAEL OPOKU AGYEMAN (2022). "Eye-Tracking Assistive Technologies for Individuals With Amyotrophic Lateral Sclerosis."