Eye-Tracking Based Control of a Robotic Arm and Wheelchair for People with Severe Speech and Motor Impairment (SSMI)

Maryam Asad Samani, Kiana Hooshanfar, Helia Shams Jey and Seyed Majid Esmailzadeh

February 20, 2024

Proceedings of the 11th RSI International Conference on Robotics and Mechatronics (ICRoM 2023), Dec. 19-21, 2023, Tehran, Iran

Eye-Tracking Based Control of a Robotic Arm and Wheelchair for People with Severe Speech and Motor Impairment (SSMI)

1st Maryam Asad Samani
dept. Electrical Engineering
Iran University of Science and Technology
Tehran, Iran
asad_m@elec.iust.ac.ir

2nd Kiana Hooshanfar
dept. Electrical and Computer Engineering
University of Tehran
Tehran, Iran
k.hooshanfar@ut.ac.ir

3rd Helia Shams Jey
dept. Electrical Engineering
Iran University of Science and Technology
Tehran, Iran
helia_shams@elec.iust.ac.ir

4th Seyed Majid Esmailzadeh
dept. Electrical Engineering
Iran University of Science and Technology
Tehran, Iran
smailzadeh@elec.iust.ac.ir

***Abstract*—Eye-tracking technology has rapidly gained popularity as a revolutionary solution for individuals with severe physical disabilities, empowering them to engage in their daily activities with newfound independence and efficiency. By utilizing advanced eye-tracking systems, individuals with limited mobility are able to control various devices and interfaces simply by moving their eyes. This paper utilizes deep learning techniques to create a low-cost real-time eye-tracking interface for controlling systems. A smart wheelchair and a robotic arm have been developed to design an eye-tracker, aiming to address the challenges faced by paralyzed people with severe physical limitations. The results demonstrate that eye-tracking is both fast and accurate, making it an effective tool for improving the interactions and accessibility for disabled individuals.**

Keywords*—*Eye-Tracking, Computer Vision, Deep Learning, Control System, Robotic Arm, Smart Wheelchair

I. INTRODUCTION

In today's world, a significant portion of the population faces physical disabilities and limitations, either from birth or as a result of accidents. These individuals experience physical impairments that necessitate assistance even for simple everyday activities. For example, individuals with paralysis or motor disabilities often resort to using long sticks held in their mouths to type. Unfortunately, this reliance on such tools can lead to long-term emotional challenges, including feelings of depression. To address the daily difficulties associated with physical impairments, robotic arms and wheelchairs have emerged as two important solutions [1]. Individuals who are elderly or have disabilities often rely on manual or electric powered wheelchairs (EPW) as a permanent means of transportation [2]. While manual wheelchairs require assistance from others, electric wheelchairs offer various control options such as joysticks, touchscreens, voice commands, or other technologies [3].

Nevertheless, these wheelchairs may not always provide optimal comfort and effectiveness, particularly for people with severe movement disorders resulting from spinal cord injuries, strokes, or neurological deficits. Everyday activities can pose challenges for wheelchair users [4]. Moreover, research conducted on quadriplegic patients has revealed that 10% of users encounter significant difficulties when operating a traditional electric wheelchair with a joystick. For the paralyzed folk, unable to move their bodies or communicate verbally, the eyes may be their only means of performing actions [5]. Therefore, one potential solution is to employ eye-tracking methods for these people. Consequently, this study focuses on the development of a gaze-controlled robotic arm and a smart wheelchair to assist impaired individuals in writing and mobility tasks.

In [6], various image processing methods for eye-tracking are reviewed. Additionally, real-time gaze tracking and blink detection provide essential insights into human behavior and attention, making them valuable for psychophysics studies and neuromarketing applications [7].

Eye trackers are typically designed to monitor eye movements while also capturing head movements. They incorporate software algorithms for data collection and analysis [8]. In comparison to conventional input devices like joysticks and voice recognition, the eye-tracking approach is particularly beneficial for patients with different types of neuromuscular diseases or neurological disorders including spinal cord injuries, cerebral palsy, and other conditions that impair movement [9]. One viable solution for eye-tracking involves glasses equipped with close-up cameras. Although these methods are accurate and reliable, they can be expensive due to the specialized hardware required for processing infrared camera images.

In recent decades, machine and deep learning methods have been widely applied, with adaptive models learning specific tasks by extracting information from previous data [10]. These modern methods are commonly used for eye gaze point detection. For instance, a camera-based eye tracker is proposed in [11] to control the computer cursor, where the user's head position is fixed, and the camera scans the user's eye, processing the data using a neural network. Another tracking algorithm in [12] utilizes Yolo (v3) for detection, while [13] designs a Convolutional Neural Network (CNN) for eye tracking with an infrared LED. [14] presents an algorithm trained on in-the-wild datasets for real-time blink detection in videos using facial landmarks. In [15], gaze point in videos is estimated through a feed-forward Artificial Neural Network (ANN) without the need for facial detection. Similarly, [16] generates, segments, and reconstructs eye-tracking data with CNN models. [17] controls an iPhone by tracking eye movements with ResNet10. The gaze of a robot is also studied in [18].

These advanced methods have demonstrated satisfactory outcomes in tracking eye movements. However, they often require the individual to maintain a still head position or utilize glasses and LED devices, which can be inconvenient for people with disabilities. Moreover, precise calibration is typically necessary for each person, and there are safety concerns related to the use of infrared LED technology. On the other hand, existing intelligent eye tracking systems only focus on tracking eye movements, which may result in lower accuracy. However, incorporating a graphical user interface (GUI) can enhance accuracy and improve comfort. Therefore, this paper aims to develop a system that utilizes a standard laptop webcam for eye tracking, enabling control of a robotic arm and wheelchair through a GUI. This empowers individuals with significant physical disabilities, particularly those with limited control over their hands, with a means to achieve independence. The system serves as a supportive tool for writing and offers a sustainable solution for wheelchair control.

For the robotic arm, the user selects a letter from an on-screen keyboard with a blink of an eye, and the robot writes the desired letter on paper with programmed accuracy. Additionally, a smart wheelchair is developed and controlled using eye blinks on the eye-tracking interface (forward, backward, right, left, stop). The wheelchair is equipped with ultrasonic sensors to detect obstacles and halt movement, ensuring user safety. Our proposed systems are efficient, suitable, and cost-effective for individuals with severe disabilities, thanks to the ease and convenience of eye-tracking. The remainder of this paper discusses eye-tracking algorithms, the robotic arm, and smart wheelchair algorithms, the training dataset, programming environment, and evaluation metrics in the methodology section, concluding with the reported results from the systems.

II. Methodology

A. Eye-Tracking Algorithm

The eye-tracking system developed in this study is created and executed in Jupyter Notebook, using the Python programming language. The employed Python libraries are outlined in Table I.

TABLE I. Python libraries used in the eye-tracking system

Library	**Function**
Open-CV	Eye-tracking webcam
Dlib	Face detection model
Pyautogui	Mouse control
Numpy	Array and parameter definition
Math	Mathematical computation
Time	Tracking time

Overall, the system is constituted of two main sections: A deep learning model for eye detection and mouse cursor control. Dlib is a prominent and open-source C++ model which can be used for computer vision tasks. It offers a pre-trained deep learning which can be used for facial landmark detection. The method involves assigning points to various landmarks on the human face, which are then detected by a neural network with corresponding probabilities for each prediction. In our research, we trained the 68 points Dlib model as illustrated in Fig. 1.

The proposed system tracks the human eye by comparing the predictions from the Dlib model with specific thresholds. To accomplish this, certain image processing steps are required, which will be discussed subsequently.

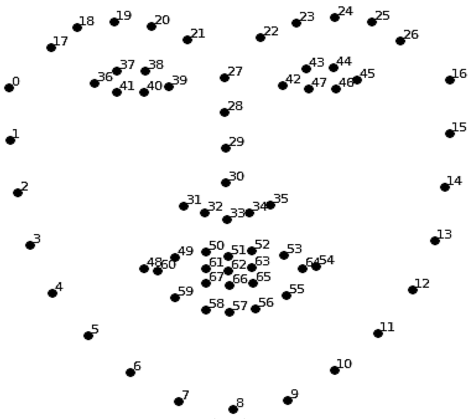

Fig. 1. Dlib model points for face detection with 68 points

Within the Dlib model, numbers 36 to 47 are specifically assigned to represent the human eyes (with 6 points allocated to each eye, denoted as p_1 to p_6). From these 6 points, the minimum and maximum values along the x and y axes are identified from Equations (1) to (4), resulting in the coordinates of the eye region.

$$x_1 = \min\big(x(p_1), x(p_2), x(p_3), x(p_4), x(p_5), x(p_6)\big) \tag{1}$$

$$x_2 = \max\big(x(p_1), x(p_2), x(p_3), x(p_4), x(p_5), x(p_6)\big) \tag{2}$$

$$y_1 = \min\big(y(p_1), y(p_2), y(p_3), y(p_4), y(p_5), y(p_6)\big) \tag{3}$$

$$y_2 = \max\big(y(p_1), y(p_2), y(p_3), y(p_4), y(p_5), y(p_6)\big) \tag{4}$$

In order to differentiate between the iris and the cornea and accurately identify the user's point of view, the eye region is processed. Initially, the specified eye region is converted into a gray-scale image. Then, each pixel in the image is standardized by comparing it with a specific constant value. Based on this comparison, the pixel's value is adjusted to either the minimum possible value (0) or the maximum possible value (255) (Equation (5)). This process results in the division of the eye region into two distinct partitions, representing the iris and the cornea, respectively as demonstrated in Fig. 2.

$$\text{Pixel Standardazation} \begin{cases} v = 0 & v < 70 \\ v = 255 & v \geq 70 \end{cases} \tag{5}$$

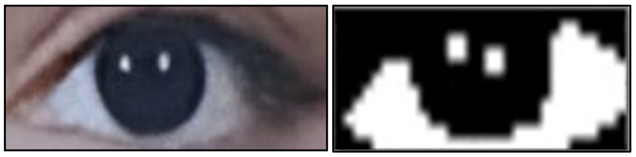

Fig. 2. a) eye region b) standardized eye region

To detect horizontal eye movements, the eye region is divided into two equal parts, x_0x_2 and x_1x_0, using a vertical line segment (depicted in Fig. 3 (a)). In each section, the number of non-zero pixels is calculated separately for the left side (L) and the right side (R). Based on the proportion of L and R, the user's eye point of view (left, right, or center) is predicted using Algorithm 1.

Similarly, for vertical eye movements, the screen is divided into two equal parts, y_0y_2 and y_1y_0, using a horizontal line segment. The number of non-zero pixels in each section is counted for the top (U) and bottom (D) regions. Based on these measurements, the eye movement is classified as up, down, or center. Additionally, our eye-tracking model takes into account the average ratio derived from both the left and right eyes to enhance the accuracy of tracking.

Algorithm1: Eye Movements

Horizontal Movements

Step1 Divide the cascade into equal regions $\boldsymbol{x_0x_2}$ (right) and $\boldsymbol{x_1x_0}$ (left)
Step2 Estimate the values of $\boldsymbol{L}$ and $\boldsymbol{R}$
Step3 Find the ratio of $\boldsymbol{L}$ to $\boldsymbol{R}$
Step4 Calculate the average ratio of both eyes as Equation (6)

$$\boldsymbol{LR} = \frac{\frac{\boldsymbol{L}}{\boldsymbol{R}}_{Right\ Eye} + \frac{\boldsymbol{L}}{\boldsymbol{R}}_{Left\ Eye}}{2} \tag{6}$$

Step5 Compare $\boldsymbol{LR}$ with defined thresholds as: $\begin{cases} \boldsymbol{Right} & \boldsymbol{LR} \leq \boldsymbol{0.8} \\ \boldsymbol{Center} & \boldsymbol{0.8} < \boldsymbol{LR} \leq \boldsymbol{1} \\ \boldsymbol{Left} & \boldsymbol{1} < \boldsymbol{LR} \end{cases}$

Vertical Movements

Step1 Divide the cascade into equal regions $\boldsymbol{y_0y_2}$ (up) and $\boldsymbol{y_1y_0}$ (down)
Step2 Estimate the values of $\boldsymbol{U}$ and $\boldsymbol{D}$
Step3 Find the ratio of $\boldsymbol{U}$ to $\boldsymbol{D}$
Step4 Calculate the average ratio of both eyes as Equation (7)

$$\boldsymbol{UD} = \frac{\frac{\boldsymbol{U}}{\boldsymbol{D}}_{Right\ Eye} + \frac{\boldsymbol{U}}{\boldsymbol{D}}_{Left\ Eye}}{2} \tag{7}$$

Step5 Compare $\boldsymbol{UD}$ with defined thresholds as: $\begin{cases} \boldsymbol{Up} & \boldsymbol{UD} \leq \boldsymbol{4.3} \\ \boldsymbol{Center} & \boldsymbol{4.3} < \boldsymbol{UD} \leq \boldsymbol{4.7} \\ \boldsymbol{Down} & \boldsymbol{4.7} < \boldsymbol{UD} \end{cases}$

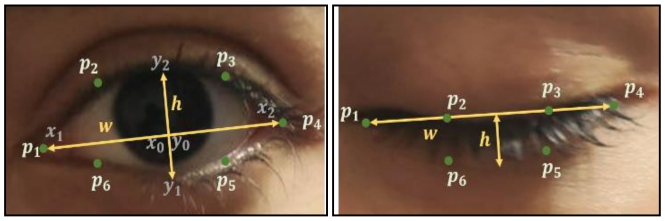

Fig. 3. a) eye points (normal) b) eye points (blinked)

The next step is to detect the blinking mode of the user. To do so, we consider a hypothetical horizontal line that is equal to the longitudinal distance of the eye, represented by points p_1 to p_4 (depicted in Fig. 3 (b)). Subsequently, the coordinates of the middle points between p_2 and p_3, as well as between p_5 and p_6, are determined using Equation (8), where i and j represent the point indexes. These two obtained points are then connected with a vertical line that has the same width as the eye's width. It is important to note that the horizontal line remains constant in all cases, whereas the vertical line varies with blinking. Finally, the ratio of the vertical line to the horizontal line is calculated. By comparing this ratio with specific thresholds, it is determined whether the user is in a blinking, scrolling or normal mode.

$$p_{i,j_{min}} = \left(\frac{x(p_i) + x(p_j)}{2}, \frac{y(p_i) + y(p_j)}{2}\right) \tag{8}$$

Algorithm2: Blink Detection

Step1 Draw $\boldsymbol{p_1p_4}$
Step2 Find $\boldsymbol{p_2p_{3_{min}}}$ and $\boldsymbol{p_5p_{6_{min}}}$ as Equation (8)
Step3 Connect $\boldsymbol{p_2p_{3_{min}}}$ to $\boldsymbol{p_5p_{6_{min}}}$
Step4 Compute the ratio of $\boldsymbol{p_2p_{3_{min}}p_5p_{6_{min}}}$ to $\boldsymbol{p_1p_4}$
Step5 Calculate the average ratio of both eyes as Equation (9)

$$\boldsymbol{B} = \frac{\frac{\boldsymbol{p_2p_{3_{min}}p_5p_{6_{min}}}}{\boldsymbol{p_1p_4}}_{Right\ Eye} + \frac{\boldsymbol{p_2p_{3_{min}}p_5p_{6_{min}}}}{\boldsymbol{p_1p_4}}_{Left\ Eye}}{2} \tag{9}$$

Step6 Compare $\boldsymbol{B}$ with defined thresholds as: $\begin{cases} \boldsymbol{Blink} & \boldsymbol{B} < \boldsymbol{0.2} \\ \boldsymbol{Scroll} & \boldsymbol{0.2} \leq \boldsymbol{B} \leq \boldsymbol{0.3} \\ \boldsymbol{Normal} & \boldsymbol{0.3} < \boldsymbol{B} \end{cases}$

TABLE II. Cursor Status based on the predicted eye direction

Eye Direction	Cursor Status
Right	20mm in the positive direction of x-axis
Left	20mm in the negative direction of x-axis
Up	7mm in the positive direction of y-axis
Down	7mm in the negative direction of y-axis
Center	No Change

TABLE III. Cursor Status based on the predicted blink direction

Blink Detection	Cursor Status
Right	Right Click
Left	Left Click
Scroll + Looking Up	40mm scroll in the positive direction of y-axis
Scroll + Looking Down	40mm scroll in the negative direction of y-axis

Table II and III provide an overview of the cursor's status based on eye movement and blinking, respectively. Additionally, the variations of these parameters over a three-minute tracking period are depicted in Fig. 4, 5 and 6. The results show that each parameter associated with eye movements (up, down, right, left and center) fluctuates as the user shifts their gaze. By comparing these parameter values to specific thresholds, the cursor's position is determined at any given moment.

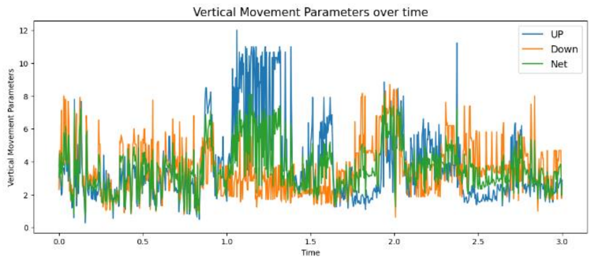

Fig. 4. Vertical movement parameters over time (minute)

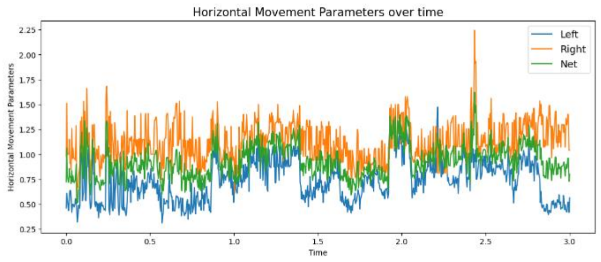

Fig. 5. Horizontal movement parameters over time (minute)

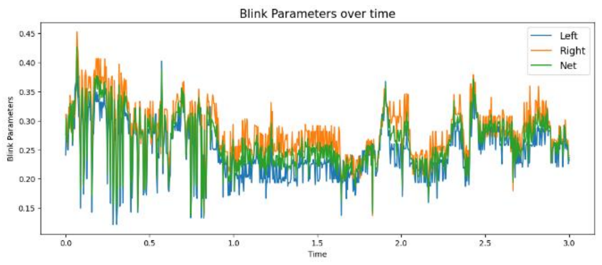

Fig. 6. Blink parameters over time (minute)

Lastly, the eye-tracking environment is illustrated in Fig. 7. The detected face and eyes are highlighted with boxes. Additionally, the two vertical and horizontal lines that have been calculated are visible. In addition, the vertical eye position (up, down, center) and horizontal eye position (right, left, center) is stated in left and right side of the screen respectively. Furthermore, blink and scroll status is shown in the center.

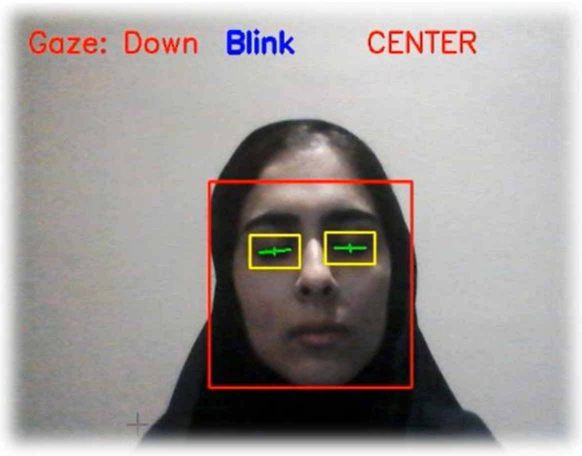

Fig. 7. Eye-tracking environment

B. Robatic Arm

The parallel robot arm used in this study (Fig. 10) has three degrees of freedom and is equipped by three SG90 servo motors plus an Arduino UNO board. It consists of 26 different pieces that are assembled using screws. The electrical components used to design the robot arm are listed below and can be seen in Fig. 8.

- Arduino UNO
- Bread Board
- Three SG90 servo motors
- +5 Volt voltage source
- Jumper Wire

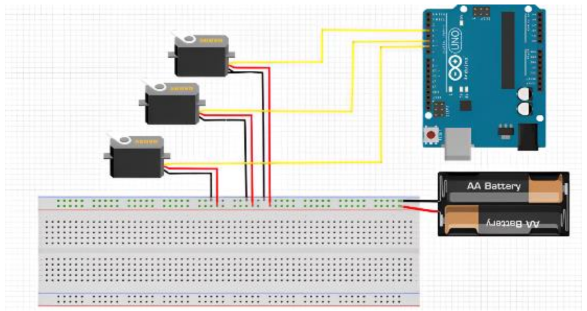

Fig. 8. Schematic of the electronic elements used in the Robotic Arm

The majority of letters in the Latin alphabet can be broken down into two main components: straight lines and curves. Therefore, to simplify the writing process for the robot, each letter is converted into a combination of vertical and horizontal lines [19]. For example, the letter C is represented as shown in Fig. 9.

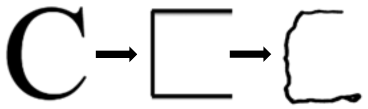

Fig. 9. A simple version of the letter with vertical and horizontal lines

Inverse kinematics is utilized to position the servo motors accurately and enable the robot to write the desired letter. In direct kinematics, the coordinates of the robot arm's end are determined based on the positions (angles) of the actuators (servo motors). Nevertheless, in inverse kinematics, the process is reversed: the actuators are positioned based on the input coordinates of the arm's end. This operation is essential for various robotic tasks such as moving a tool along a specific path, relocating objects, or obtaining a specific viewpoint. Inverse kinematics has been extensively studied, and multiple techniques have been developed to solve it.

In the next step, the dimensions of the arm are measured. The exact angles of the servo motors for each letter are then determined based on the inverse kinematics of the robot. After transforming all the letters into lines and obtaining the coordinates of the robot arm's pen tip (p_x, p_y, p_z), as well as the start and end points of each line, these values are used as input for the inverse kinematics calculation. This process yields the required angles for writing each letter. Finally, the Arduino board is programmed using the calculated angles to actuate the servo motors and enable the robot to write the letters.

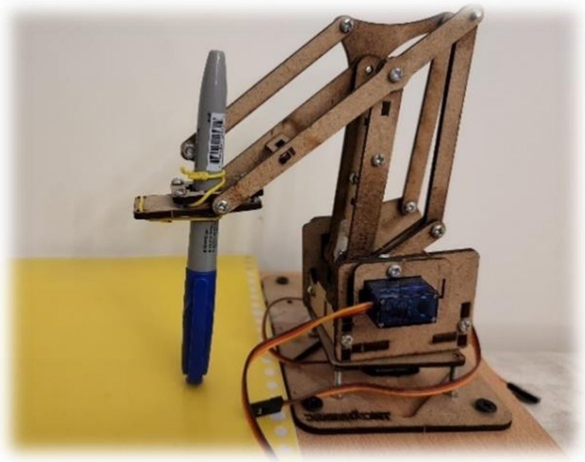

Fig. 10. The final robotic arm

The corresponding parameters for each servo motor are presented in Table IV and demonstrated in Fig. 11, 12, and 13. The angles q_1, q_2 and q_3 represent the angles of the middle, right, and left servo motors, respectively. Additionally, Equations (10) to (22) describe the equations associated with the dynamics of the robot.

TABLE IV. Robot Arm Parameters

Parameter	**Definition**
X	x axis
Y	y axis
Z	z axis
l_0	distance between the center of the coordinate and the center of the robotic arm
l_1	distance between the center of the robotic arm and l_3
l_2	length of the front piece of the arm
$l_{3,0}$	upper distance between l_2 and l_4
$l_{3,1}$	lower distance between l_2 and l_4
l_4	length of the back piece of the arm
q_1	anticlockwise angle between y-axis and the arm (the angle of the middle servomotor)
q_2	anticlockwise angle between x-axis and l_2 (the angle of the right servomotor)
q_3	anticlockwise angle between z-axis and $l_{3,1}$ (the angle of the left servomotor)
$q_{3,0}$	anticlockwise angle between l_2 axis and the arm
α	angle between s and r
β	angle between l_2 and s
γ	angle between l_2 and l_3
φ	angle between $l_{3,1}$ and e
Ψ	angle between l_2 and e
d_5	distance between the center of the robot and the pen
e	diagonal distance between l_2 and l_4
s	distance between the center of the servo motor and center of the pen
r	horizontal distance between the center of the servo motor and center of the pen
z	vertical distance between the center of the pen and the center of the servo motor
h	distance between the center of the servo motor and the ground

$$q_1 = \tan^{-1}\frac{p_x - l_0}{p_x} + \sin^{-1}\frac{d_5}{\sqrt{(p_x - l_0)^2 + p_y^2}}) \tag{10}$$

$$q_2 = \pi - \alpha - \beta \tag{11}$$

$$q_3 = \Psi + \varphi + \frac{\pi}{2} - q_2 \tag{12}$$

$$q_{3,0} = \pi - \gamma \tag{13}$$

$$\alpha = \tan^{-1}\frac{z}{r} \tag{14}$$

$$\beta = \cos^{-1}(\frac{l_2^2 - l_3^2 + s^2}{2sl_2}) \tag{15}$$

$$\gamma = \cos^{-1}(\frac{l_2^2 + l_3^2 - s^2}{2l_2l_3}) \tag{16}$$

$$s = \sqrt{r^2 + z^2} \tag{17}$$

$$r = \sqrt{{p_x}^2 + p_y^2} - l_1 \tag{18}$$

$$e = \sqrt{l_{3,0}^2 + l_2^2 - 2l_{3,0}l_2\cos(q_{3,0})} \tag{19}$$

$$\Psi = \sin^{-1}(\frac{l_{3,0}\sin(q_{3,0})}{e}) \tag{20}$$

$$\varphi = \cos^{-1}(\frac{e^2 + l_{3,1}^2 - {l_4}^2}{2el_{3,1}}) \tag{21}$$

$$z = p_z - h \tag{22}$$

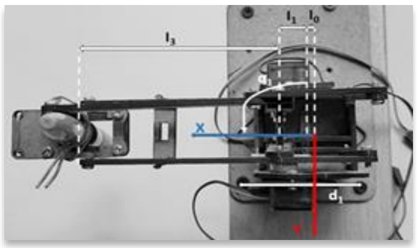

Fig. 11. Robotic arm parameters (1)

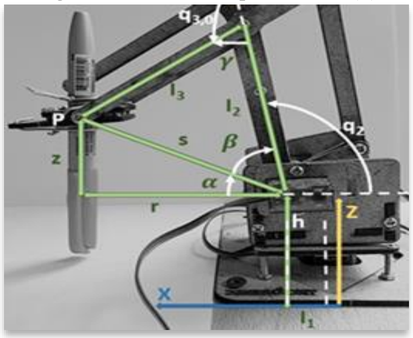

Fig. 12. Robotic arm parameters (2)

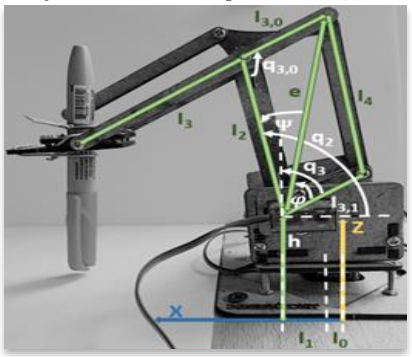

Fig. 13. Robotic arm parameters (3)

To establish a connection between the robot and the eye-tracking system, a Graphical User Interface is created using the Processing IDE, as shown in Fig. 14. The program communicates with the Arduino through serial communication. The user is able to control the cursor using eye movements and select a letter by blinking. When a key is pressed, a corresponding sound is played, enhancing the user's understanding and interaction with the system. Finally, some words are printed out in order to evaluate the handwriting of the robot. The word ‘CONTROL’ in Fig. 15 is an example.

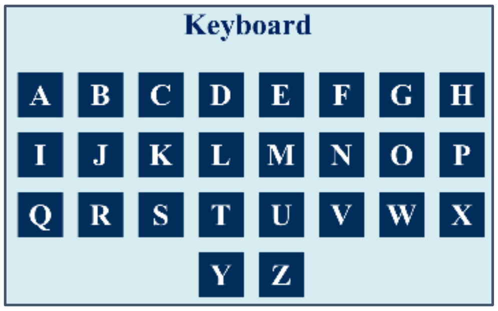

Fig. 14. GUI for the robotic arm

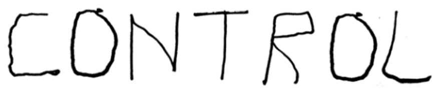

Fig 15. Writing the word 'CONTROL' with the robotic arm

C. Smart Wheelchair

This section highlights the development of the eye-tracking-based smart wheelchair intended for individuals with mobility impairments. The smart wheelchair is equipped with various components, including Arduino UNO, Driver L298N, Bluetooth 06HC, Two DC Motors, and an Ultrasonic Sensor, as depicted in Fig. 16.

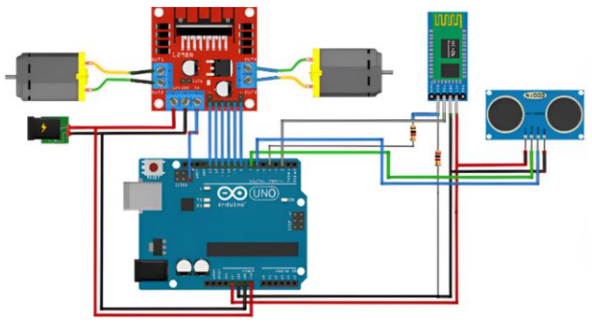

Fig. 16. Electronic elements used in smart wheelchair

The image processing section of the system involves a built-in webcam on a laptop and custom image processing software developed in Python. Open-CV is utilized to determine the direction of eye movements based on captured images. The user interacts with the system through a user interface displayed on the computer, which sends commands to the Arduino via Bluetooth. The Bluetooth module receives these command signals at the receiving end and finally Arduino controls the motors accordingly.

In the wheelchair's warning module, ultrasonic sensors are employed to detect objects in front of the wheelchair by emitting and receiving sound waves. The Arduino processes the data to calculate the distances to obstacles. If an obstacle is detected, the Arduino sends a stop command, and the motors halt.

The L298 driver, in conjunction with the Arduino, allows for the control of the speed and direction of the DC motor using pulse width modulation and an H bridge. Pulse width modulation adjusts the average input voltage by generating a sequence of on and off pulses. The duty cycle, which determines the width of the pulses, affects the average voltage applied to the DC motor. A higher duty cycle results in a higher average voltage and, consequently, a higher speed, while a lower duty cycle leads to a lower average voltage and a lower speed. The rotation direction of the DC motor can be controlled by changing the polarity of its input voltage, typically achieved using an H bridge. The H bridge circuit consists of four switches arranged in the shape of the letter "H," with the motor in the center. By closing specific switches simultaneously, the polarity of the voltage applied to the motor is reversed, causing it to rotate in the opposite direction.

The designed graphical interface is illustrated in Fig. 17. Serial communication is employed between the program and the Arduino, with data being transferred via Bluetooth and the speed of communication between the computer and the Bluetooth module must be set accurately to ensure proper serial communication. The user controls the smart wheelchair (Fig. 18) by sending one movement command as in Table V from the GUI to the Arduino.

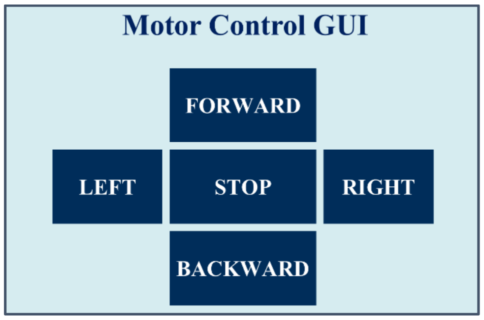

Fig. 17. GUI for wheelchair

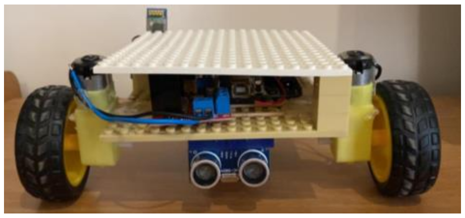

Fig. 18. Smart Wheelchair overview

TABLE V. Direct Current Motor Status

Direction	IN1	IN2	IN3	IN4
Stop	0	0	0	0
Forward	1	0	1	0
Backward	0	1	0	1
Right	1	0	0	1
Left	0	1	1	0

III. Evaluation

In order to assess the effectiveness of the eye tracking GUI, a step function is defined and tracked, as depicted in Fig. 19. The axes represent the dimensions of the laptop screen (1920*1080). It can be observed that the vertical movements exhibit slightly higher accuracy, while the horizontal movements show some level of noise.

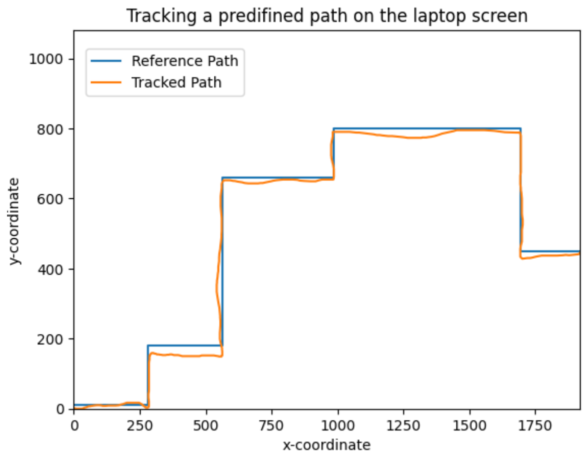

Fig. 19. Eye Tracking Evaluation

Moreover, the Mean Square Error (MSE) and Root Mean Square Error (RMSE) is utilized for quantitative evaluation of the system. Equation (23) and (24) represents the formula for MSE and RMSE, where q and q_d indicate the tracked and reference path values for the i^{th} data point, respectively, and N denotes the number of samples. Table VI demonstrates the MSE values for both vertical and horizontal directions, indicating that the system performs better in terms of vertical movements.

$$\text{MSE} = \frac{1}{N} \sum_{i \in \text{data}} (q(i) - q_d(i))^2 \tag{23}$$

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i \in \text{data}} (q(i) - q_d(i))^2} \tag{24}$$

TABLE VI. MSE and RMSE for vertical and horizontal tracking

Movement Direction	MSE	RMSE
Vertical	0.168	0.410
Horizontal	0.254	0.504
Total	**0.333**	**0.577**

IV. DISCUSSION

This study has successfully developed an eye-tracking-based automatic wheelchair control system, providing individuals with physical disabilities the ability to control the wheelchair through eye movements only by looking at a screen. To ensure safety, ultrasonic sensors are integrated into the wheelchair, enabling immediate stops when obstacles are detected. This gaze-controlled wheelchair not only offers easy accessibility for the disabled people but also enhances safety by providing automatic obstacle protection. Additionally, the study has also achieved the capability of writing using a robot arm. This advancement is expected to be well-received, as it allows the folk with severe physical and mobility limitations to create content without requiring human assistant. However, our system's tracking accuracy is impacted by the lighting conditions in the environment. To improve the performance in future works, the following suggestions are proposed.

- Employing a higher-resolution camera to improve tracking accuracy.
- Utilizing servo motors with higher torque for accurate positioning of the robot arm at higher locations.
- Developing a robot arm with stronger and more stable hardware that is lightweight and easy to reposition.
- Incorporating servo motors with a high weight tolerance.
- Implementing a system where the paper continually moves on a rail, enabling the robot to write all the letters in one location without the need for arm extension.

V. CONCLUSION

This paper focuses on implementing an eye-tracking based method using a robotic arm and a wheelchair to assist individuals with severe disabilities in their daily activities. Both the robotic arm and smart wheelchair are connected to the eye-tracking system through a graphic user interface on a local computer. The results indicate satisfactory performance in terms of both speed and accuracy. In future work, there is potential for improving the hardware components. Additionally, since eye trackers are not widely utilized by the general public, further research in this field can serve as a foundation for conducting applied research in related areas.

ACKNOWLEDGMENT

This research received no external funding.

AUTHOR CONTRIBUTION

The authors contributed equally on this paper and the last author supervised this study.

REFERENCES

[1] N. S. Zamani, M. N. Mohammed, M. I. Abdullah, and S. Al-Zubaidi, "A new developed technique for handwriting *robot*," in *2019 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, 2019.

[2] S. Mahmud et al., "A multi-modal human-machine interface for controlling a smart wheelchair," in *2019 IEEE 7th Conference on Systems, Process, and Control (ICSPC)*, IEEE, 2019, pp. 10–13.

[3] C. Aruna, A. D. Parameswari, M. Malini, and G. Gopu, "Voice recognition and touch screen control based wheel chair for paraplegic persons," in *2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*, 2014.

[4] B. Gold, N. Morgan, and D. Ellis, "Speech and audio signal processing: processing and perception of speech and music," *John Wiley & Sons*, 2011.

[5] S. M. T. Saleem, S. Fareed, F. Bibi, A. Khan, S. Gohar, and H. H. Ashraf, "IMouse: eyes gesture control system," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 9, 2018.

[6] J. B. Mulligan, "Image processing for improved eye-tracking accuracy," *Behav. Res. Methods Instrum. Comput.*, vol. 29, no. 1, pp. 54–65, 1997.

[7] B. Wąsikowska, "The application of eye tracking in business," 2014.

[8] A. Taghizade and Z. Aghakasiri, "Eye-Tracking Method'Usage for Understanding the Cognitive Processes in Multimedia Learning," *Journal of Educational Studies*, vol. 11, pp. 41–52, 2018.

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

[10] A. Namdari, M. A. Samani and T. S. Durrani, "Lithium-ion battery prognostics through reinforcement learning based on entropy measures," *Algorithms 15, no.* 11, p. 393, 2022.

[11] E. Demjén, V. Aboši, and Z. Tomori, "Eye tracking using artificial neural networks for human computer interaction," *Physiol. Res.*, vol. 60, no. 5, pp. 841–844, 2011.

[12] I. Rakhmatulin and A. T. Duchowski, "Deep neural networks for low-cost eye tracking," *Procedia Comput. Sci.*, vol. 176, pp. 685–694, 2020. [13] J. Griffin and A. Ramirez, "Convolutional neural networks for eye tracking algorithm," arXiv Prepr. 2018.

[14] T. Soukupova and J. Cech, "Eye blink detection using facial landmarks," in *21st computer vision winter workshop*, Rimske Toplice, Slovenia, 2016.

[15] N. Zdarsky, S. Treue, and M. Esghaei, "A deep learning-based approach to video-based eye tracking for human psychophysics," *Front. Hum. Neurosci.*, vol. 15, p. 685830, 2021.

[16] W. Fuhl, Y. Rong, and E. Kasneci, "Fully convolutional neural networks for raw eye tracking data segmentation, generation, and reconstruction," in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021.

[17] H. D. Kannan, "Eye tracking for the iPhone using deep learning," (Doctoral dissertation, Massachusetts Institute of Technology), 2017.

[18] F. Manzi et al., "The understanding of congruent and incongruent referential gaze in 17-month-old infants: an eye-tracking study comparing human and robot," *Sci. Rep.*, vol. 10, no. 1, p. 11918, 2020.

[19] S. Yussof, A. Anuar, and K. Fernandez, "Algorithm for robot writing using character segmentation," in *Third International Conference on Information Technology and Applications (ICITA'05)*, 2005.