Contents lists available at ScienceDirect

e-Prime - Advances in Electrical Engineering, Electronics and Energy

journal homepage: www.elsevier.com/locate/prime

Design, simulation and evaluation of two coding programming languages for an eye-tracking controlling system for a three degrees of freedom robot useful for paralyzed people

Alireza Abbasimoshaei [a,*], Marco Winkel [b], Thorsten A. Kern [c]

[a] Research Assistant at Hamburg University of Technology, Germany
[b] Ms.C Student at Hamburg University of Technology, Germany
[c] Director at Institute for Mechatronics in Mechanics, Hamburg University of Technology, Germany

ARTICLE INFO

Keywords:
Eye Tracking
Three degrees of freedom robot
Robotic arm
Mathematical modeling
Eye tracking device

ABSTRACT

In this study, the use of an eye tracking system is tested and discussed by using the tracked gaze point as a means to control a device. In this project, a two-dimensional simulated robot for paralyzed people was used to move the arm in the plane where the wrist is in a plane parallel to the frontal plane of the eye. Both MATLAB and Python programs are proposed for programming the eye tracking system (ET) and their results are compared. Moreover, the tests show how can this system works with robot and sampling rate which is important in robotic system is better than previous studies. At the end of the study, after mathematical modeling, a simulation of the three joint movements of the robotic arm was performed. This robot is designed in a way to move the arm of the paralyzed people. Several tests were performed and a multi-body simulation was connected to the eye tracker to verify the resulting interaction concept.

1. Introduction

According to a 2013 report, between 250,000 and 500,000 spinal cord injuries occur annually worldwide [1]. The consequences of such an injury can range from limited body movement in the lower legs to complete loss of control of the somatic nervous system below the neck as well as parts of the autonomic nervous system. To help mobility-impaired individuals regain independence, one option is to set up systems that can track the impaired person. This tracking system can provide the input signals to the control system and can be used to perform various tasks, such as a robotic arm, a robotic hand with multiple fingers, or other controllable devices, depending on the different mechanisms [2–6]. Eye tracking is one such system used in human-computer interaction and is already used in practice to control electrical devices. However, it is not yet used as an option to control a device that allows mobility-impaired individuals to regain control of their limbs and perform simple tasks [7,8]. The purpose of eye trackers is to enable physically impaired individuals to perform movements by having the alignment of their eyes analyzed and interpreted by a program. Some previous work on eye tracking includes electrooculography [9,10], camera-based eye tracking [11,12], and eye-based tracking [13]. Wood et al. [14] applied a limbus method to detect iris edges from camera-based eye images, and their results showed that this model has good efficiency and accuracy on wearable devices. Vielence et al. [15] also proposed a new technique that was able to detect the eye graze under different circumstances, such as detecting the gaze under a glass. Lu et al. [16] used linear regression to increase the accuracy of eye detection. Ince et al. [17] conducted a study on a low-cost 2D eye tracking system using a shape and intensity based method to detect the center of the pupil to detect the gaze more accurately. In a study, Zeng et al. [18] developed a hybrid human-robot interface (HRI) system using a brain-computer interface (BCI) and an eye-tracking interface to continuously modulate movement speed via motor intentions. The new system enabled continuous, smooth, and collision-free movement of the end effector approaching the target. Compared to a system without robotic autonomy support, this system significantly reduces the error rate, and the time and effort required by the user is also less. Cio et al. [19] studied the concept of controlling a robotic arm by a combination of stereovision and gaze tracking, in which the person could successfully grasp the target object with excellent efficiency. In addition, gaze trackers differ in whether they are attached to the head or to objects, such as a table. These different techniques all have their advantages and

* Corresponding author.
E-mail address: al.abbasimoshaei@tuhh.de (A. Abbasimoshaei).

https://doi.org/10.1016/j.prime.2022.100065
Received 23 February 2022; Received in revised form 31 May 2022; Accepted 9 September 2022
Available online 16 September 2022

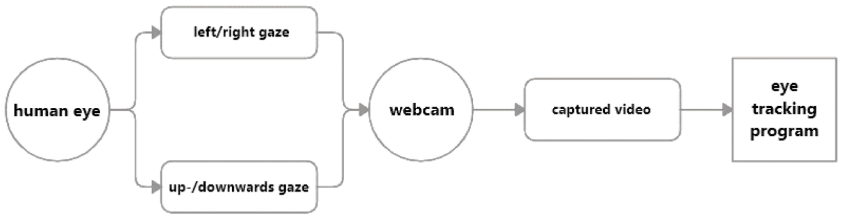


Fig. 1. Exemplary functionality of tracking the gaze.

disadvantages in certain situations and are therefore compared based on the above requirements. Murthy et al. [20] investigated the potential and accuracy of eye-tracking and gaze-guided interfaces in military aviation. They reported that the accuracy of the eye-tracking system is less than five degrees of gaze. However, they pointed out that eye trackers may not work properly under strong external illumination. Tang and Zhang [21] proposed a method using the detection algorithm in combination with gray prediction for eye tracking. The numerical geoscience model was used for their study and prediction of the position of the eye tracker. The predicted position is used as a reference for the eye region to be searched. Raudonis et al. [22] conducted a study to develop an eye tracking system for three online applications (mobile robot, eye tracker, and computer games) to assist people with disabilities. Sharma et al. [23] investigated a self-driving system based on eye-tracking techniques that electronically connects a tablet without requiring a dedicated hardware AR (Augmented Reality) Display. This system helps a person with severe speech impairments control a robotic arm by eye gaze, for patients with severe speech and motor impairments (SSMI). Vidrios-Serrano et al. [24] presented a stiffness controller for the eye tracker based on a nonlinear proportional-derivative structure for robotic manipulators and Hooke's law to model the interaction with the robotic environment, considering elasticity as a generalized bounded spring. The components of force, applied torques and position error in two types of visually constrained stiffness controllers (SP-SD and SPD) and the results are compared and reported. On the other hand, the effectiveness of different approaches followed by analysis of gaze error distribution is analysed by ablation studies [25] and Gibaldi et al. [26] evaluated MATLAB and Tobii Eye tracking controller for research area. A confident web technology-based eye-tracking with required hard- and software is improved for even sophisticated experimental paradigms in all of cognitive psychology [27]. Moreover, an affordable, accessible, and extensible pervasive eye tracking and gaze-based interaction platform is developed by Kassner et al. [28].

This paper investigates a program for object-mounted eye tracking in conjunction with an eye-tracking device. The appropriate eye-tracking device is determined and a suitable eye-tracker is developed based on the conclusions. The eye-tracking program monitors the output of the eye-tracking device, processes the content of the tracked data so that the position of the eye can be detected, and then provides the relative position of the eye as output. This output is used as input to simulate the movement of a human limb based on the data received from the eye tracking system. A mathematical system layout for the arm is determined. The results are then used to determine if it is appropriate to use object-mounted eye tracking to drive a simulation.

2. Method

2.1. Description of object-mounted eye tracking

In this work, a two-dimensional setup was used to move the arm in the plane where the wrist is in a plane parallel to the frontal plane of the eye. A suitable setup requires tracking an additional body function to also allow movements perpendicular to the frontal plane of the eye. An advantage of two-dimensional tracking is that it allows tracking of both eyes, but requires data from only one eye. Once the number of degrees of freedom is determined, a system must compute these degrees of freedom

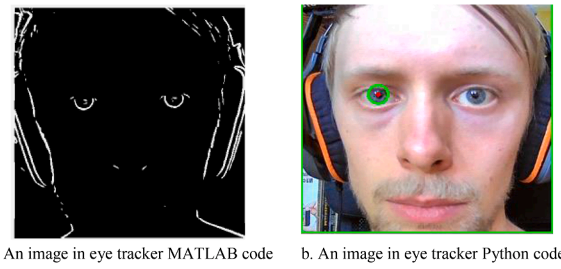

a. An image in eye tracker MATLAB code
b. An image in eye tracker Python code

Fig. 3. Produced images in eye tracker code by MATLAB and Python.

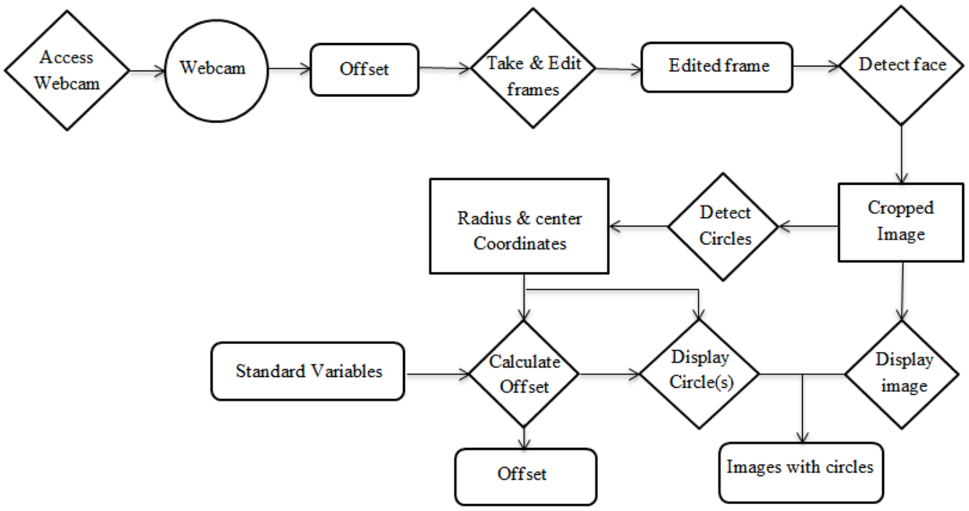


Fig. 2. Exemplary functionality of a coded eye tracker.

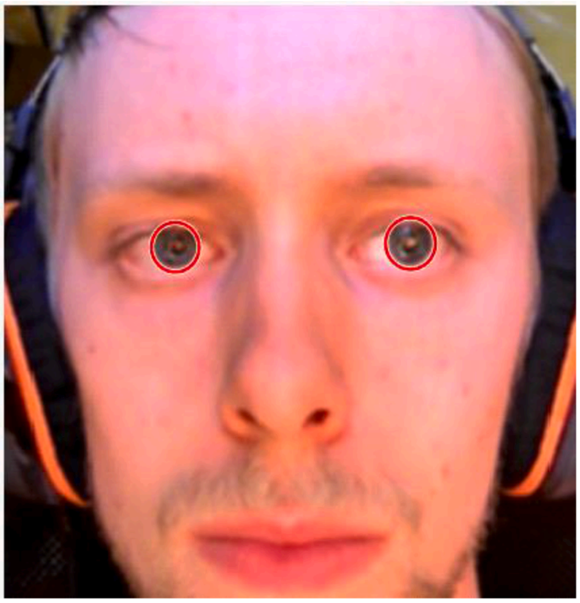


Fig. 4. Eye tracker with a 1080p webcam.

based on the position of the eye. Therefore, a suitable eye tracker must be set up to operate according to the selected number of degrees of freedom and obtain eye position data based on the video-based object-based eye tracking. Fig. 1 shows the schematic diagram of the gaze tracking process. Fig. 2 shows a schematic diagram of the proposed eye detection program based on circle detection.

2.2. Implementation in Matlab and Python

First, a snapshot of the video feed of the desired webcam is taken and then mirrored on the y-axis. A Gaussian filter smooths the mirrored image and the face detection function is then used to detect the face using the Viola Jones algorithm. To track circles on an image, the edges are first detected. The resulting image can be seen in Fig. 3-a. Next, Python code is developed to communicate with a ROS (Robot Operating System) node. Denavit-Hartenberg parameters are specified, as well as default variables and a definition for a function along with the required input variables. Then the actual eye tracking begins, which continues as long as the Python code is accessing a webcam or when it is manually terminated. On the processed image, the selected Haar Cascade function searches for a face. The image must be a grayscale image for the Hough circle transformation can be applied to the image. This transformation determines the centers of potential circles and then determines the most appropriate circle by comparing the expected positions of the facial features. When a circle is found, its values are converted to integers and the circle and its center are displayed in Fig. 3.

3. Discussion

3.1. Comparison of an eye tracker in Matlab and in Python

The two eye trackers in MATLAB and in Python were tested in five different ways. To compare them, the first method involved looking straight at the camera, while the following four methods involved looking 10 cm up, down, left, and right from the camera. They sit at a distance of 40 cm from the camera, resulting in an angle of about 14° between the line of sight and the line between the face and the camera. On the left side of the camera, there was a window through which light could enter. Tests of each eye tracker measured the total number of images, the number of frames with and without a detected eye, the execution time, and angular offset of the detected eye center. These data are then used to calculate the percentage of images with a successfully detected eye and the rate at which images are detected and processed (Fig. 4). The reference point for the eye's center is determined by the average found circle coordinates of the preliminary test runs and the offset in x- and y-direction (in pixel) is then calculated as the distance to this reference point.

In MATLAB, the values for the x- and y- coordinates of the determined position of the center of the eye were taken for the right eye. Accuracy is calculated as the average angular offset (distance) (in degrees of visual angle) between fixation locations and the corresponding locations of the fixation targets [28].

$$accuracy\ [^\circ] = \frac{\sum_{i=1}^{n} \sqrt{offset\_\phi^2 + offset\_\vartheta^2}}{n} \tag{1}$$

In which ϕ is the angle of pupil in horizontal direction and ϑ is in vertical direction. Precision is calculated as the average Root Mean Square (RMS) of the angular distance (in degrees of visual angle) between successive samples: x_i,y_i to x_{i+1},y_{i+1}.

$$precision\ [cm] = \frac{\sum_{i=1}^{n-1} \sqrt{\left(offset\_\phi[i] - offset\_\phi[i+1]\right)^2 + \left(offset\_\vartheta[i] - offset\_\vartheta[i+1]\right)^2}}{n-1} \tag{2}$$

Detection rate is the ratio of images with a detected eye to the total

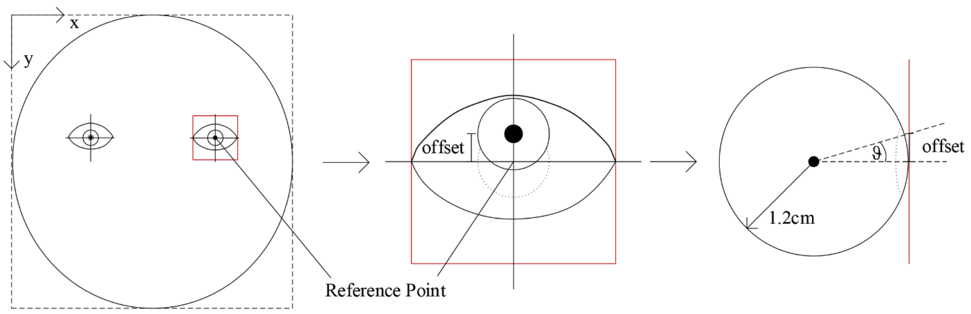


Fig. 5. Offset and reference point definition.

Table 1
Average results of 3 tests

		Accuracy [°]		Precision [°]		Detection Rate [%]		Sampling Rate [1/s]	
		MATLAB	Python	MATLAB	Python	MATLAB	Python	MATLAB	Python
Test 1	⊙	7,93	16,94	6,63	13,69	41	78	2,5	11,62
	◐	25,18	31,33	21,9	9,48	22	76	2,2	11,63
	◑	9,97	18,38	5,81	8,58	86	60	2,09	11,62
	◓	17,25	26,12	10,67	9,05	98	76	2,38	11,61
	◒	13,5	29,18	7,25	13,09	76	44	2,34	11,62
Test 2	⊙	11,83	16,64	11,64	13,47	52	94	2,39	11,64
	◐	11,3	13,77	10,68	11,89	57	83	2,43	11,64
	◑	15,54	12,64	13,24	14,46	89	82	2,41	11,64
	◓	33,11	18,04	32,37	7,35	95	60	2,43	11,64
	◒	22,41	15,8	10,05	17,56	84	71	2,38	11,61
Test 3	⊙	18,2	17,18	8,91	16,28	93	77	2,33	11,65
	◐	16,12	13,31	14,16	16,34	69	78	2,15	11,65
	◑	26,14	11,62	7,72	11,88	96	73	2,36	11,64
	◓	25,98	15,42	10,23	12,3	96	87	2,31	11,62
	◒	13,24	15,74	8,85	20,73	91	72	2,32	11,64

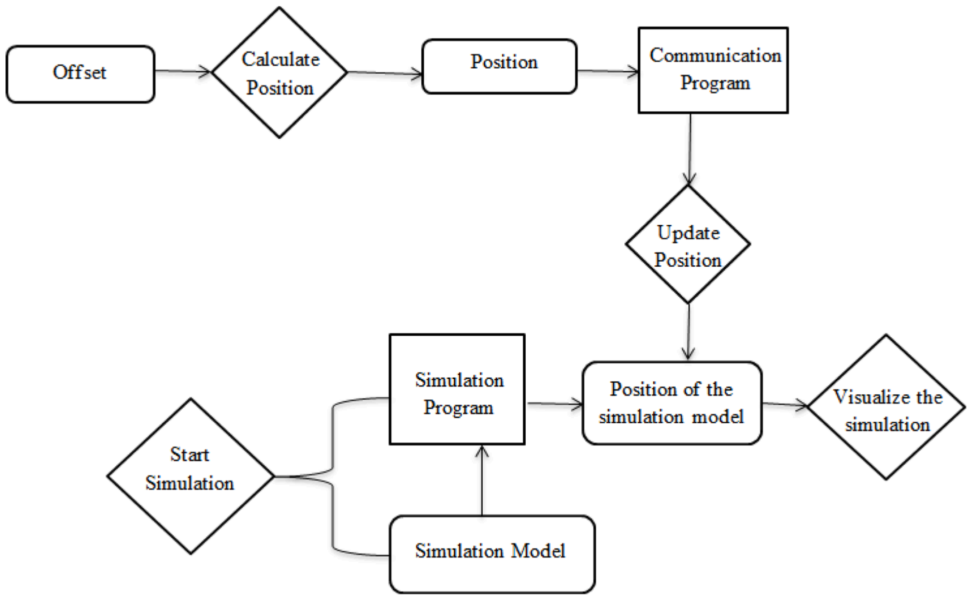


Fig. 6. Concept of visualizing calculated movement.

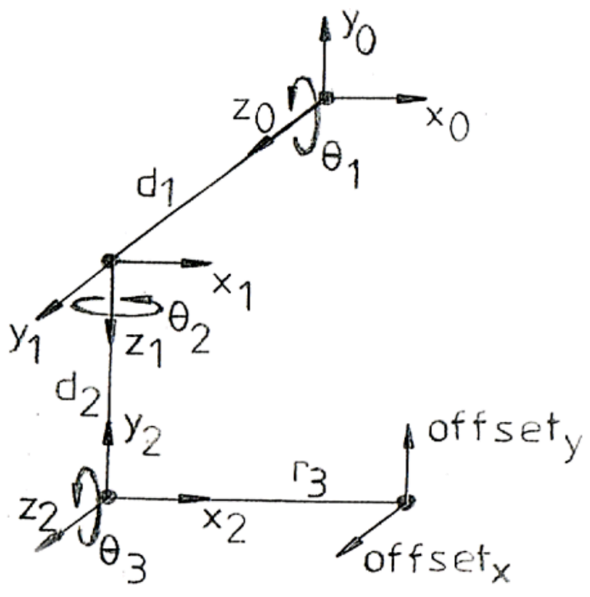


Fig. 7. Convention of the simulated arm.

number of images [28].

$$detection\ rate = \frac{\sum Number\ of\ images\ with\ a\ detected\ eye}{\sum Number\ of\ images} \tag{3}$$

Additionally, sampling rate is defined as number of data samples per second collected for each eye [26].

$$sampling\ rate = \frac{\sum Number\ ofdata\ samples}{\sum Time\ of\ Execution} \tag{4}$$

It is also important to mention that, for the accuracy and precision metrics, from the offset in pixel we calculate them according the visual angles of the eye, ɸ and ϑ (by transforming the offset in pixel to offset in cm and then determining the angle with an assumed eye radius of 1.2 cm. Fig. 5 shows an example of offset in vertical direction and how ϑ results from it. scale_factor is experimentally determined (16/320 for python implementation and 16/380 for Matlab implementation).

$$-\vartheta = atan2\left(\frac{offset[cm]}{1.2}\right),\ offset[cm] = offset[pixels]*scale\_factor \tag{5}$$

The distance between camera and eye was 40 cm, 100 images have been sampled, a chin rest was used, and the gaze points were marked on a paper template around the camera at a distance of 10 cm to the camera lens (corresponds to a visual angle of about 14°). The results in Table 1 are based on the data obtained from the tests and were calculated using formulas (1), (2), (3), and (4).

3.2. Visualization of the calculated movement

After the system is determined, the processed data can be passed to a simulation that visualizes the calculated motion. Optionally, ROS allows communication between different programs, so that processed data in one program can be forwarded to another. For this purpose, an arm is simulated with the given ROS program, so that an eye tracker can specify the position of an arm. The general concept is described in Fig. 6, where the squares represent the software, the diamonds represent the software functions, and the rounded rectangles represent the data.

3.3. Proposed Performable movement and mathematical convention

As mentioned before, the simulation of the arm should move in two dimensions. Therefore, movements of the arm that can perform a movement within a two-dimensional plane must be considered. The joint between the upper and lower arm can simply be described as a hinge joint, although the radius is also sufficient to allow rotation of the lower arm. Considering the design and medical criteria, only two joint movements, flexion and extension of the shoulder joint and elbow joint, can affect the angle of objects held by the hand. Since these two movements are always performed in the same plane of motion, they can counteract each other by flexing the elbow and extending the shoulder and vice versa. Thus, the three joint movements performed in the simulation are flexion and extension in the elbow joint and the shoulder joint, and medial and lateral rotation of the arm in the shoulder joint. Since the two movements in the shoulder joint are rotations, they are also represented by two rotational joints in the simulation, while the elbow joint is represented by a hinge joint.

Table 2
Denavit-Hartenberg parameters

Joint	θ_i	d_i	r_i	α_i
1	θ_1	$d_1 = d_{shoulder}$	0	90°
2	θ_2	$d_2 = d_{upperarm}$	0	-90°
3	θ_3	0	$r_3 = r_{forearm}$	0

For the kinematic calculations, three angles should be determined for the two joints, namely the angle for flexion and extension of the shoulder, the angle for medial and lateral shoulder rotation, and the angle for flexion and extension of the forearm. To calculate the angles of the joints based on the direction of gaze of the tracked eye, the Denavit-Hartenberg convention is appropriate. The direction of the x and y axes is arbitrary, but should be interpreted with respect to the coordinate system of the origin. The point of the body that lies on the sagittal plane and lies on an imaginary line perpendicular to the sagittal plane and leading from this plane to the shoulder is the origin point. This point lies on the plane that divides the body into left and right. On this plane, it is located at the point closest to the shoulder joint. The shoulder socket is represented by a sphere and is connected to a cylinder from the origin to the shoulder socket by a rotational joint that performs flexion and extension of the shoulder. The upper arm is represented by another cylinder and is connected to the simulated shoulder socket via another rotational joint that performs medial and lateral shoulder rotation. Finally, the forearm is represented by a third cylinder and is connected to the upper arm via a hinge joint (Fig. 7).

The Denavit-Hartenberg convention uses four parameters to determine the motion of an object. In the case of the arm simulation, the variables are shown in Table 2.

The Angle θ_i represents rotation around the z-axis, angle α_i represents the rotation around the x-axis, the distance d_i represents translation along the z-axis, and finally, the distance r_i represents translation along the x-axis. For each of these parameters, there is a corresponding translation or rotation matrix. These are then multiplied to determine the position matrix ${}^{i}_{i-1}T$ calculated as follows:

$${}^{i}_{i-1}T = Trans_{z_{i-1}}(d_i)*Rot_{z_{i-1}}(\theta_i)*Trans_{x_i}(r_i)*Rot_{x_i}(\alpha_i) \tag{4}$$

The resulting matrix ${}^{i}_{i-1}T$ looks like this:

$${}^{i}_{i-1}T = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i)*\cos(\alpha_i) & \sin(\theta_i)*\sin(\alpha_i) & r_i*\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)*\cos(\alpha_i) & -\cos(\theta_i)*\sin(\alpha_i) & r_i*\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5}$$

To calculate the position matrix ${}^{i}_{i-1}T$ from the origin to the position where the hand is located, the position matrices from the origin to the shoulder joint, from the shoulder joint to the elbow, and finally, from the elbow to the hand have to be multiplied as follow:

$${}^{3}_{0}T = {}^{1}_{0}T(\theta_1)*{}^{2}_{1}T(\theta_2)*{}^{3}_{2}T(\theta_3) \tag{6}$$

The position matrix ${}^{i}_{i-1}T$ is constructed in this way:

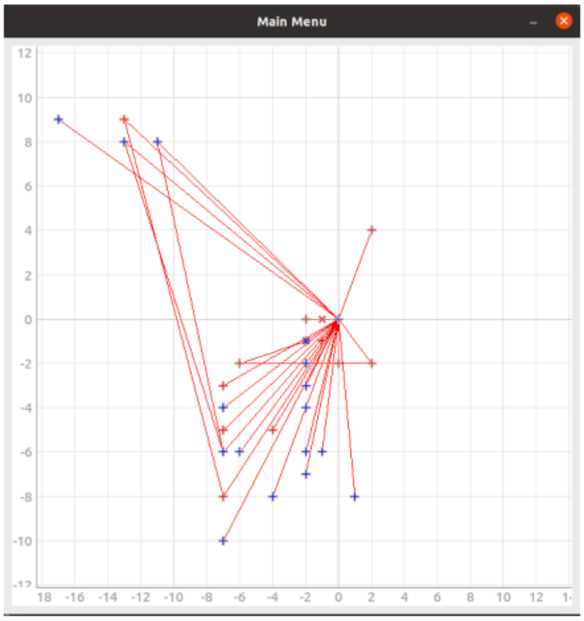


Fig. 8. Test with eye tracker by MATLAB.

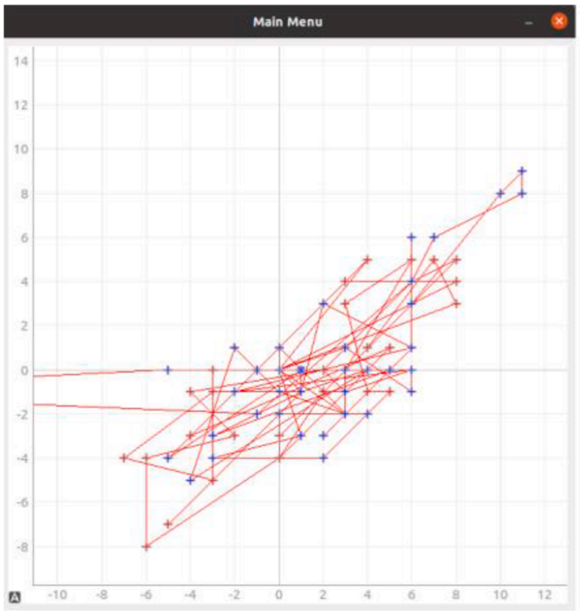


Fig. 9. Test with eye tracker by Python.

$$ {}^{i}_{i-1}T = \begin{pmatrix} R_{xx} & R_{xy} & R_{xz} & T_x \\ R_{yx} & R_{yy} & R_{yz} & T_y \\ R_{zx} & R_{zy} & R_{zz} & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7} $$

Here "R" stands for an element of the rotation matrix within the matrix ${}^{i}_{i-1}T$ and "T" stands for an element of the translation vector within the matrix ${}^{i}_{i-1}T$. After calculating the matrix, the translation vector in (8) on the left side can be equated with the position vector of the hand as determined by the eye tracker on the right side with rearranging the values, such that r_3 is only within the left vector:

$$ \begin{pmatrix} r_{forearm}{}^*[cos(\theta_1){}^*cos(\theta_2){}^*cos(\theta_3) - sin(\theta_1){}^*sin(\theta_3) - 1] \\ r_{forearm}{}^*[sin(\theta_1){}^*cos(\theta_2){}^*cos(\theta_3) + cos(\theta_1){}^*sin(\theta_3)] \\ r_{forearm}{}^*sin(\theta_2){}^*cos(\theta_3) \end{pmatrix} = \begin{pmatrix} -d_{upperarm}{}^*sin(\theta_1) \\ d_{upperarm}{}^*[1 + cos(\theta_1)] + offset_y \\ offset_z \end{pmatrix} \tag{8} $$

The distance of the hand to the shoulder socket depends exclusively on θ_3, since the other two angles only change the distance of the hand to the body, but not to the shoulder socket. Therefore, the angle θ_3 can be determined by transforming the cosine rule. Here, $d_{upperarm}$ serves as the adjacent leg of the triangle, $r_{forearm}$ is the opposite leg, and the hypotenuse is given by $\| \vec{h} - \vec{s} \|^2$, where $\vec{h}$ is the position vector of the hand with respect to the origin and $\vec{s}$ is the position vector of the shoulder with respect to the origin. Using the cosine rule, the equation for determining θ_3 is as follows:

$$ \theta_3 = 90^\circ - \arccos\left(\frac{\| \vec{h} - \vec{s} \|^2 - d^2_{upperarm} - r^2_{forearm}}{-2{}^*d_{upperarm}{}^*r_{forearm}} \right) \tag{9} $$

In the above formula, subtracting the result of the “arccos” from 90° serves to make the forearm point forward when the eyes are looking straight ahead. In this way, both flexion and extension of the elbow can occur in a combined range of 180°. The angle θ_3 and the angle θ_2 is determined by rearranging the z-components of each entry in the position vectors of the hand. This results in the following equation:

$$ \theta_2 = \arcsin\left(\frac{offset_z}{\cos(\theta_3){}^*r_{forearm}} \right) \tag{10} $$

By computing the position vector with $\theta_1$set to a fixed value, one can derive the angle between the actual desired position and the position obtained from the fixed value θ_1. The best setting for the angle $\theta_1$is equal to 0°, because setting it to any other angle would require subtracting that angle from the determined angle between the actual position and the position calculated by setting the angle$\theta_1$to a fixed value. Since the angle $\theta_1$only appears in the x- and y-components of the position vectors, the equation ignores the displacement in the z-direction. The resulting equation looks as follow:

$$ \theta_1 = arctan2(h_x, h_y) - arctan2(h_x[\theta_1 = 0], h_y[\theta_1 = 0]) \tag{11} $$

The function “arctan2” is chosen instead of the regular “arctan” because the use of the latter can lead to incorrectly calculated angles. This can happen because the function “arctan” cannot distinguish between quadrant 1 and 3 as well as between quadrant 2 and 4. The elements “h_x” and “h_y” represent the x- and y-values of the position vector derived by calculating “$offset_y$” and “$offset_z$” , while the values “$h_x[\theta_1 = 0]$” and “$h_y[\theta_1 = 0]$” represent the values of the position vector. It is derived by inserting the previously calculated angles θ_2 and θ_3 into the translation vector and setting the angle θ_1 equal to 0°.

3.4. Implementing the simulation and eye tracking result

The following diagrams show the results of two exemplar tests that measured the displacement of an eye during static gaze at specific points and the robot following the eye inputs. One test was performed with the original eye tracker using MATLAB and one test was performed with an eye tracker using Python.

In each test, the camera was at a distance of about 30 cm from the face. The gaze points are each 4 cm to the right or left and 4 cm above or below the lens. They are marked on a paper template that was placed around the camera.

For each eye tracker, a test measured the offset of 50 detected circles when looking left and the offset of 50 detected circles when looking right. A sound signal indicated the moment to move the gaze to the next point.

The other test run of each eye tracker had the same setup but

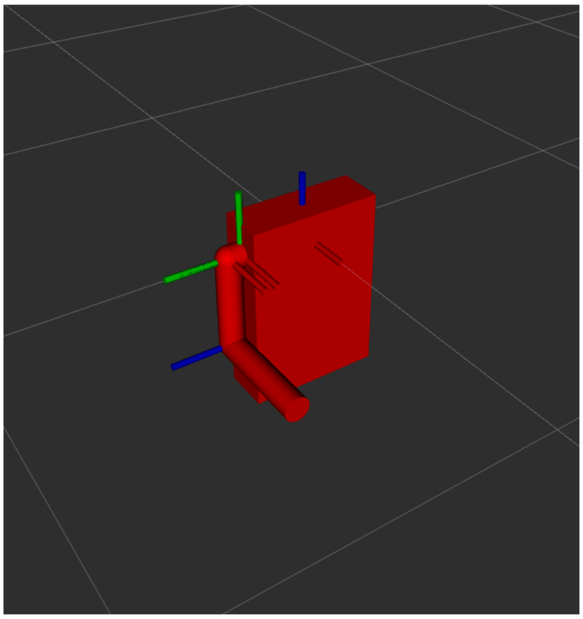


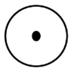
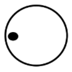
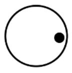
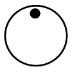
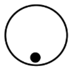
Fig. 10. Robot arm in a start position.

measured the offset of 50 detected circles when gaze was directed to the marked point 4 cm above the lens and then, after the signal, the offset of 50 detected circles when gaze was directed to the marked point 4 cm below the lens. Image resolution was 720×1280 pixels for all tests. Individual data points are marked with a "+" and the mean with an "x".

The graph in Fig. 8 shows the results of the first test with the eye tracker in MATLAB. The red data points show the offset of each detected circle when gaze is directed to the left, and the blue points when gaze is directed to the right. Since many samples are on the same point, the total number of data points appears to be less than 100. Since the blue and red data points do not cluster in a particular area and have almost the same mean value, it would not be possible for an algorithm to distinguish between the two different inputs (left or right gaze). The main reason for this is the fact that the reference point from which the offset is calculated varies randomly from image to image. This variation results from the face detection algorithm that is run on each captured image and then outputs the coordinates of the face in the image.

The graph in Fig. 9 shows the result of the second test with the eye tracker using Python. The red data points show the offset of each detected circle when the gaze is directed upward, the blue points when the gaze is directed downward. The graph shows the same features as the first one, but faster. There are no clusters and the variations appear to be random. Moreover, the test results with robot will be described in the

Table 3
Robot arm angles according to the gaze point movement

	ϕ [°]	ϑ [°]	t1 [°]	t2 [°]	t3 [°]
	0	0	38,39	7,51	17,36
	-14	0	38,39	-7,51	17,36
	14	0	41,65	22,75	14,27
	0	-14	49,45	7,51	17,19
	0	14	28,82	-6,59	14,44

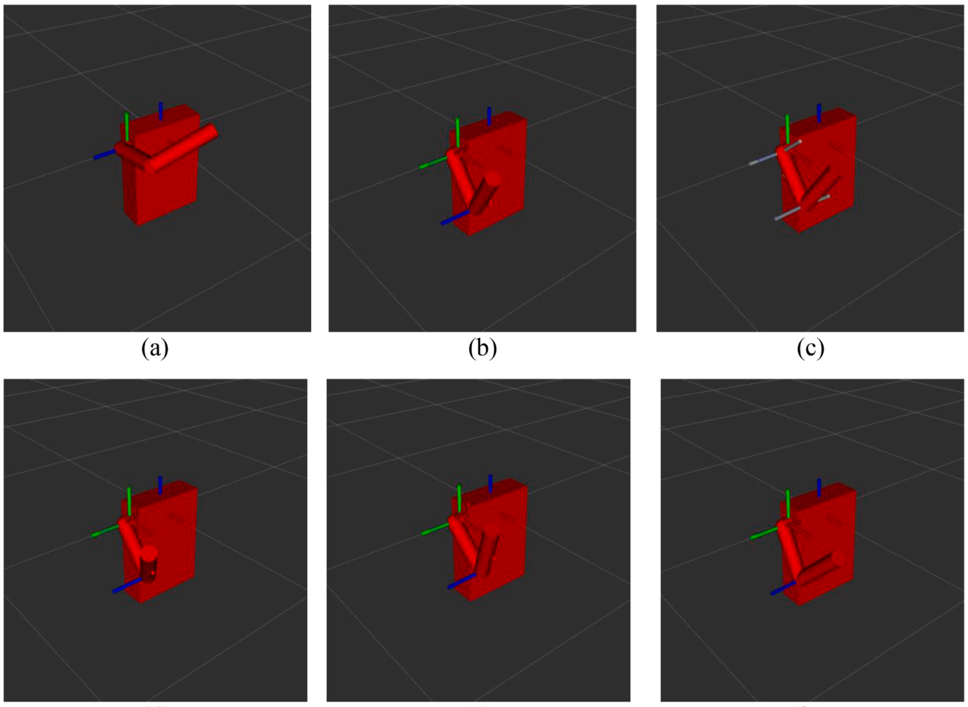


Fig. 11. Robot arm position regarding to (a) a random movement (b) central gaze point (c) left gaze point (d) right gaze point (e) upper gaze point (f) for lower gaze point.

Table 4
Robot arm angles according to the gaze point movement

	Accuracy [°]		Precision [°]		Detection Rate [%]		Sampling Rate [1/s]	
	MATLAB	Python	MATLAB	Python	MATLAB	Python	MATLAB	Python
⊙	12,65	16,92	9,06	14,48	62,00	83,00	2,41	11,64
⊙	17,53	19,47	15,58	12,57	49,33	79,00	2,26	11,64
⊙	17,22	14,21	8,92	11,64	90,33	71,67	2,29	11,63
⊙	25,45	19,86	17,76	9,57	96,33	74,33	2,37	11,62
⊙	16,38	20,24	8,72	17,13	83,67	62,33	2,35	11,62

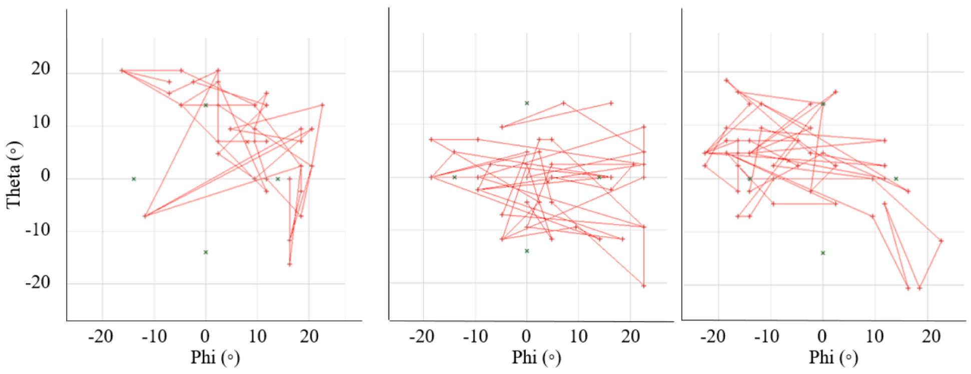


Fig. 12. Test with eye tracker on healthy users.

Table 5
Test results on healthy users

	Test 1	Test 2	Test 3
Accuracy	17,61 °	13,8 °	15,09 °
Precision	15,37 °	20,97 °	15,64 °
Detection Rate	0,45	0,43	0,55
Sampling Rate [1/s]	11,62	11,62	11,63

next section.

3.5. Implementing the simulation

The arm is simulated in the simulation program RViz, which is an implemented complement to the communication program ROS. A urdf file is used to specify the links of the robot model and the joints between each link, so that the robot model represents the set factors. As it can be seen, the robot is modelled in a way that it can be attached to the paralyzed arm and move it. In the simulation neither the collision between the links is activated nor the speed of the executed movement is limited. Thus, the simulation is able to update its angles immediately when the view changes and adjusts its position accordingly. Moreover, the speed at which the position of the arm would be changed in practice by a motion guide is also limited. This is due to the physical limitations, as the position of the arm cannot be updated instantly in practice, and also the safety of the user should be ensured by limiting the speed of a practical device. Fig. 10 shows the robot simulation in a situation where all calculated angles are zero and only the fixed 90° rotations of the coordinate system at the joints "t2″ and "t3″ rotate the links of the robot model.

Fig. 11 shows the robot simulation in different moving states where all angles "t1", "t2″ and "t3″ were changed based on the implemented angle data. Based on the systems identified and the tests results (Tables 3 and 4), it is shown that this method of eye tracking is suitable as a means of controlling robot movement.

3.6. Data of the object-mounted eye tracker implemented on random healthy users

Fig. 12 and Table 5 show the average results of three tests of the eye tracker performed on three random volunteer healthy subjects. In each test, the user looked directly at the camera. The camera was at a distance of 40 cm from the face and a chin rest was used. Each test contains 100 samples. Phi and theta indicate the angular displacement of the eyeball in the horizontal and vertical directions, respectively. The green markings in the graph serve as orientation points in order to facilitate the assessment of the size of the offset. Each mark corresponds to a gaze point 10 cm from the camera lens. All details can be found in the Table 5.

The communication with the robot was not verified in previous works such as [26–28], so the parameters are not optimized to be more suitable for robotic projects. For example, compared to previous work such as [27], this system is not more accurate, but the accuracy is sufficient for the robotic system, and the more important parameter, namely the sampling rate, which affects the communication with the robot, is better in this work.

4. Conclusion

In this study, the ability of eye trackers attached to the object was used as a system to improve the abilities of motion-impaired individuals by controlling a motion guidance device. This system can be used for both prosthetic and external robot arm. The general results of experimental testing, code implementation, and arm simulation can be summarized as follows:

While processing images, motion can be adjusted more quickly when Python is used instead of MATLAB, even if the eye is temporarily occluded. In practice, the speed at which motion guidance adjusts the position of a limb may be limited by the need to ensure safe execution of the limb movement. In addition, both programs can provide sufficient speed for robotic applications. This new algorithm development of the object-mounted eye tracker for this particular application, instead of a commercially available system, provides a robotic optimized open source implementation that is widely available and can be easily set up with a webcam. It can help the medical robots used for paralyzed people to be compatible enough with different situations and enable paralyzed people to control a prosthetic arm or an external robot by eye-tracking.

In general, object-mounted eye trackers can compete with head-mounted eye trackers in an environment where it is possible to set up camera devices. Therefore, the range of motion of object-mounted eye trackers is more limited compared to head-mounted eye trackers. The object-mounted eye tracker has the potential advantage of a longer useful life because the useful life is limited only by the time that the fully charged motion guidance device can support the user's movement, while the time is shorter for head-mounted eye trackers because the energy of the head device is consumed more quickly than that of the motion guide device. Moreover, an object-mounted eye tracker does not require physical contact to the user (glasses may get uncomfortable with time). In general, it can be said that a good combination of these two methods is the most suitable and comfortable environment for paralyzed people.

Future outlook

In order to freely control a prosthetic arm or robot in three-dimensional space, control over a 3rd degree of freedom would need to be implemented. Future work could investigate the forms of additional input that make this possible. Alternatively, an algorithm could be implemented to adjust one joint at a time and then move to the next joint. This stepwise approach, while slow, could be implemented with control over 2 degrees of freedom.

Most of the variance in the measured offset comes from the face recognition algorithm used, whose variances in determining face position extend to the position of the reference points, making them unstable. Future implementations will need to make changes to compensate for this variance which also can distinguish between two different inputs (left or right gaze). Moreover, we are in the process of obtaining ethical clearance to conduct tests on users with motor impairments and then also to test the system as a clinical and home therapy method.

Funding

The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Code and data availability

All data and code will be made available on request to the correspondent author's email with appropriate justification.

Author contributions

All authors contributed to the study. Conceptualization, Alireza Abbasimoshaei and Thorsten A. Kern; Data curation, Marco Winkel; Formal analysis, Marco Winkel; Investigation, Marco Winkel; Methodology, Alireza Abbasimoshaei and Thorsten A. Kern; Project administration, Alireza Abbasimoshaei and Thorsten A. Kern; Software, Marco Winkel; Supervision, Thorsten A. Kern; Validation, Marco Winkel; Writing – original draft, Alireza Abbasimoshaei; Writing – review & editing, Alireza Abbasimoshaei and Thorsten A. Kern. All authors have read and agreed to the published version of the manuscript. All authors read and approved the final manuscript.

Ethics approval

Not applicable.

Consent to participate

Informed consent was obtained from the subject involved in the study.

Consent to publish

The authors affirm that human research participants provided informed consent for publication of the images in Figs. 3 and 4

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] J. Bickenbach, I. Boldt, M. Brinkhof, J. Chamberlain, R. Cripps, M. Fitzharris, B. Lee, R. Marshall, S. Meier, M. Neukamp, P. New, R. Nicol, A. Officer, B. Perez, P. von Groote, P. Wing, Querschnittlähmung – global betrachtet, in: Querschnittlähmung – Internationale Perspektiven, World Health Organization, Geneva, Switzerland, 2013, p. 20.

[2] T. Deemyad, N. Hassanzadeh, A. Perez-Gracia, in: Coupling Mechanisms for Multi-Fingered Robotic Hands with Skew Axes: Proceedings of the 4th IFToMM Symposium on Mechanism Design for Robotics, Springer, 2018, pp. 344–352.

[3] T. Deemyad, O. Heidari, A. Perez-Gracia, Singularity design for rrss mechanisms, in: USCToMM Symposium on Mechanical Systems and Robotics, 2020, pp. 287–297, pages.

[4] R. Moeller, T. Deemyad, A. Sebastian, Autonomous navigation of an agricultural robot using RTK GPS and Pixhawk, Intermount. Eng. Technol. Comput. (IETC) (2020) 1–6.

[5] A. Bannat, J. Gast, T. Rehrl, W. Rösel, G. Rigoll, F. Wallhoff, A multimodal human-robot-interaction scenario: working together with an industrial robot, in: International Conference on Human-Computer Interaction, Springer, Berlin, Heidelberg, 2009, pp. 303–311.

[6] F. Ben Taher, N. Ben Amor, M. Jallouli, EEG control of an electric wheelchair for disabled persons, in: International Conference on Individual and Collective Behaviors in Robotics (ICBR), Sousse, 2013, pp. 27–32, https://doi.org/10.1109/ICBR.2013.6729275.

[7] S. Chandra, G. Sharma, S. Malhotra, D. Jha, A.P. Mittal, Eye tracking based human computer interaction: applications and their uses in Man and Machine Interfacing (MAMI), in: International Conference on IEEE, 2005, pp. 1–5.

[8] S. Wibirama, H.A. Nugroho, K. Hamamoto, Evaluating 3D gaze tracking in virtual space: a computer graphics approach, Entertain. Comput 21 (2017) 11–17.

[9] A. Tonin, A. Jaramillo-Gonzalez, A. Rana, M. Khalili-Ardali, N. Birbaumer, U. Chaudhary, Auditory Electrooculogram-based communication system for ALS Patients in Transition from Locked-in to Complete Locked-in State, Sci. Rep. 10 (2020) 8452, https://doi.org/10.1038/s41598-020-65333-1.

[10] N. Steinhausen, R. Prance, H. Prance, A Three Sensor Eye Tracking System Based ON Electrooculography, SENSORS,IEEE, Valencia, 2014, pp. 1084–1087, https://doi.org/10.1109/ICSENS.2014.6985193.

[11] N. Mir-Nasiri, Camera-based 3D object tracking and following mobile robot, in: IEEE Conference on Robotics, Automation and Mechatronics, Bangkok, 2006, https://doi.org/10.1109/RAMECH.2006.252655.

[12] M. Dirik, O. Castillo, A. Fatih Kocamaz, Gaze-guided control of an autonomous mobile robot using type-2 fuzzy logic, Appl. Syst. (2019), https://doi.org/10.3390/asi2020014.

[13] Maheshwari, A.: Face detection as done in 2001: viola jones algorithm. (2020). visited at: https://iq.opengenus.org/face-detection-using-viola-jones-algorithm/.

[14] E. Wood, T. Baltrušaitis, L.P. Morency, P. Robinson, A. Bulling, in: A 3D morphable eye region model for gaze estimation in European Conference on Computer Vision, Cham, 2016, pp. 297–313.

[15] F. Vicente, Z. Huang, X. Xiong, F. De la Torre, F. Zhang, D. Levi, Driver gaze tracking and eyes off the road detection system, IEEE Trans. Intell. Transp. Syst. 16 (4) (2015) 2014–2027.

[16] F. Lu, Y. Sugano, T. Okabe, Y. Sato, Adaptive linear regression for appearance-based gaze estimation, IEEE Trans. Pattern Anal. Mach. Intell. 36 (10) (2015) 2033–2046.

[17] I.F. Ince, J.W. Kim, A 2D eye gaze estimation system with low-resolution webcam images, EURASIP J. Adv. Signal Process. 40 (1) (2011).

[18] H. Zeng, Y. Shen, X. Hu, A. Song, B. Xu, H. Li, et al., Semi-autonomous robotic arm reaching with hybrid gaze–brain machine interface, Front Neurorobot 13 (2011) 111.

[19] Y-SL-K. Cio, M. Raison, C.L. Menard, S. Achiche, Proof of concept of an assistive robotic arm control using artificial stereovision and eye-tracking, IEEE Trans. Neural Syst. Rehab. Eng. 27 (12) (2019) 2344–2352.
[20] Murthy, L.R.D., Mukhopadhyay, A., Yellheti, V., Arjun, S., Thomas, P., Dilli Babu M., Singh Saluja, K.P., Shree, D.V., Biswas, P.: Eye gaze controlled interfaces for head mounted and multi-functional displays in military aviation environment. arXiv preprint arXiv:2005.13600, (2020).
[21] J. Tang, J. Zhang, Eye tracking based on grey prediction, in: Proc. 1st Int. Workshop Educ. Technol. Comput. Sci., 2009, pp. 861–864.
[22] V. Raudonis, R. Simutis, G. Narvydas, Discrete eye tracking for medical applications, in: Proc. 2nd ISABEL, 2009, pp. 1–6.
[23] V.K. Sharma, L.R.D. Murthy, K.P. Singh Saluja, V. Mollyn, G. Sharma, P. Biswas, Webcam controlled robotic arm for persons with SSMI, Technol. Disabil. 32 (3) (2020) 179–197.
[24] C. Vidrios-Serrano, M. Mendoza, I. Bonilla, B. Maldonado-Fregoso, A generalized vision-based stiffness controller for robot manipulators with bounded inputs, Int. J. Control Autom. Syst. 19 (2021) 548–561, https://doi.org/10.1007/s12555-019-1056-7.
[25] P. Biswas, Appearance-based gaze estimation using attention and difference mechanism, in: In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 3143–3152.
[26] A. Gibaldi, M. Vanegas, P.J. Bex, G. Maiello, Evaluation of the Tobii EyeX Eye tracking controller and MATLAB toolkit for research, Behav. Res. Methods 49 (3) (2017) 923–946. Jun.
[27] K. Semmelmann, S. Weigelt, Online webcam-based eye tracking in cognitive science: a first look, Behav. Res. Methods 50 (2) (2018) 451–465. Apr.
[28] M. Kassner, W. Patera, A. Bulling, Pupil: an open source platform for pervasive eye tracking and mobile gaze-based interaction, in: In Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing: Adjunct publication, 2014 Sep 13, pp. 1151–1160.

Dr. Alireza Abbasimoshaei, is currently a researcher assistant at the Institute for Mechatronics in Mechanics at the Hamburg University of Technology in Germany. Before joining iMEK, he designed and built four robots and controlled them. The last one was at the Technical University of Braunschweig in Germany. He filed two patents for rehabilitation devices and sold a finger rehabilitation robot he developed to a hospital partner. He also developed a new control system for rehabilitation robots in the field of control. He is an expert in mechatronic system design with special emphasis on mechanical and control system design.

Thorsten A. Kern received his Dipl.-Ing. and Dr.-Ing. degrees from Darmstadt University of Technology (TUDA), Darmstadt, Germany in the fields of actuator and sensor development for medical human-machine-interfaces (HMIs) in applications like minimally-invasive surgery and catheterizations. He is currently a director at Hamburg University of Technology, Germany, of the Institute for Mechatronics in Mechanics. He previously worked in Automotive Industry at Continental as a R\&D manager for interior components, leading a team of 300 engineers worldwide. He joined Continental in 2008 covering various functions with increasing range of responsibility in actuator development, motor-development and active haptic device development before shifting towards R\&D management and product-management on Head-Up-Displays. Between 2006 and 2008 he was working in parallel in a startup focussing on medical interventions and was finalizing the 1st edition of „Engineering Haptic Devices". He joined Hamburg University in January 2019. His-interests are specifically focussed on all types of electromagnetic sensors and actuators and their system-integration towards larger motor- or sensor-systems in high-dynamic applications.

Marco Winkel, is currently graduated from the Institute for Mechatronics in Mechanics at the Hamburg University of Technology in Germany. He designed and built an eye tracking system and controlled it for his bachelor thesis Moreover, he has knowledge in mechatronic system design with special emphasis on coding.