Eye-tracking technology as a means of ensuring the accessibility of education for people with severe musculoskeletal disorders has been investigated in this study. A number of assistive technologies have been devised up to now that possess significant educational potential, but their autonomous use is mainly associated with such difficulties as dependence on the instability of wireless networks or external computing resources. The need to overcome the technological gap between the capabilities of assistive technologies and their real autonomy is a task addressed in this work.

The study's result is the designed fully autonomous architecture based on virtual reality headsets (Meta Quest), which enables the execution of all eye image processing activities and provides direct interface control using the device's processor. The hardware was improved (frame miniaturization, transition to a wired USB interface were carried out); software was developed in the Rust language in the OpenXR API (Application Programming Interface) layer format.

That has made it possible to get rid of network delays and ensure the system's operation, resistant to any conditions. Direct interaction with the headset hardware is provided by effective optimization of convolutional neural networks and the use of low-level system calls. An immersive graphical user interface (GUI) was designed, successfully tested for system configuration and calibration. The GUI became available directly in the virtual space of the headset.

The results could be used for individual learning at home, as well as at inclusive educational institutions or rehabilitation centers that organize training for people with severe musculoskeletal disorders

Keywords: eye-tracking, inclusive education, motor disorders, VR headset, accessibility of education, adaptive technologies

UDC 004.93:004.5:376
DOI: 10.15587/1729-4061.2026.351773

DESIGN OF AUTONOMOUS GAZE-TRACKING SYSTEM FOR INCLUSIVE EDUCATION

Andrii Symkin*
ORCID: https://orcid.org/0009-0005-2439-3560
Oleksandr Mitsa
Doctor of Technical Sciences, Professor*
Department of Mathematics and Informatics
Ferenc Rákóczi II Transcarpathian Hungarian University
Kossuth sq., 6, Berehove, Transcarpathia, Ukraine, 90202
ORCID: https://orcid.org/0000-0002-6958-0870
Yurii Horoshko
Corresponding author
Doctor of Pedagogical Sciences, Professor, Head of Department**
E-mail: horoshko_y@ukr.net
ORCID: https://orcid.org/0000-0001-9290-7563
Hanna Tsybko
PhD, Associate Professor**
ORCID: https://orcid.org/0000-0002-1861-3003
Viktor Shakotko
PhD, Associate Professor
Department of Technological and Professional Education
Oleksandr Dovzhenko Hlukhiv National Pedagogical University
Kyivska str., 24, Hlukhiv, Ukraina, 41400
ORCID: https://orcid.org/0000-0002-3004-5045
*Department of Informative and Operating Systems and Technologies
Uzhhorod National University
Narodna sq., 3, Uzhhorod, Ukraine, 88000
**Department of Computer Science and Computer Engineering
T. H. Shevchenko National University «Chernihiv Colehium»
Hetmana Polubotka str., 53, Chernihiv, Ukraina, 14013

Received 10.11.2025
Received in revised form 21.01.2026
Accepted 30.01.2026
Published 27.02.2026

How to Cite: *Symkin, A., Mitsa, O., Horoshko, Y., Tsybko, H., Shakotko, V. (2026). Design of autonomous gaze-tracking system for inclusive education. Eastern-European Journal of Enterprise Technologies, 1 (9 (139)), 68–77. https://doi.org/10.15587/1729-4061.2026.351773*

1. Introduction

Among the fundamental human rights, an important place belongs to the right to education. One of the leading directions of modern education development is inclusive education. In the United Nations (UN) Convention on the Rights of Persons with Disabilities, inclusive education is defined as the basic principle of ensuring equal access to education for all people regardless of their physical or intellectual characteristics.

However, for people with significantly impaired mobility, the realization of the right to full education is one of the most difficult problems of modern education. For example, according to the Cerebral Palsy Guide, at the beginning of 2026, cerebral palsy in the world affects from 1.5 to 3 people per 1000 newborns, and many of them have minimized motor properties due to severe forms of the disease. People with neurodegenerative diseases, multiple sclerosis, and the consequences of spinal cord injuries also belong to the specified category. The conventional way of exchanging messages with a computer using a regular keyboard, mouse or joystick, where independent control of hands is required, is significantly complicated or impossible for them.

For the needs of this category of students, traditional teaching methods and standard computer interfaces are unsuitable. The inability to fully participate in the educational process causes social isolation, limited opportunities for self-realization, as well as a decrease in the quality of life. This renders special relevance to scientific research into this area.

2. Literature review and problem statement

The results of the research by a number of scientists in [1, 2] provide a toolkit in the form of effective algorithms for eye tracking. These papers are considered classic for the industry; the authors tried to combine approaches and methods of eye tracking into a clear structure. However, the proposed solutions remained quite expensive and insufficiently adapted to a specific user. In subsequent studies, such as [3], to improve adaptation, an open-source eye tracking toolkit was proposed for reliable collection of gaze data in augmented reality (AR) based on Unity 3D and Microsoft HoloLens 2 (USA).

In study [4], 207 papers reporting research results on eye tracking were analyzed. The authors note the absence of a single approach to the processing of empirical data in them and provide recommendations on the methodology for processing the results of such experiments. At the same time, the similarity of models and algorithms for processing and analyzing recorded eye tracking data was taken into account. Moreover, the availability of hardware and software tools related to eye tracking for the average user continues to remain an unresolved problem, primarily due to their high cost and the need for additional adaptation for a specific user.

An option for overcoming the difficulties related to a more widespread and effective use of eye tracking systems is proposed in studies on possible ways of practical application of developments in eye tracking in education. Thus, in work [5], researchers analyzed the features of teachers' work in the classroom using an eye tracking system. It was determined that there are differences in the attention paid by the teacher to individual students.

In [6], a review of the scientific literature on the use of eye tracking systems to study the features of students' use of information technologies is given. In the considered papers, the use of eye tracking was aimed at studying the interaction between a teacher and a student in the educational process. The problem of designing an autonomous eye tracking system for inclusive education remained unresolved.

Partial ways to overcome the problem of accessibility of eye tracking systems were proposed in [7]. It suggested the architecture of an effective and inexpensive device that uses a camera and software to detect and track human eye movement. The final product is aimed at expanding the capabilities of patients with limited mobility, especially those suffering from amyotrophic lateral sclerosis (ALS) and quadriplegia. The work describes an alternative interface to conventional input methods, such as a keyboard, mouse, touch panel, etc.

Eye tracking systems are actively used in medicine, for example, during the treatment of stroke consequences. Study [8] emphasized that stroke often leads to significant impairments in motor function, cognitive processing, and visual fields, which requires devising effective rehabilitation strategies. In a significant proportion of cases, patients need to be retrained to receive information from the environment. Scientists also emphasize the insufficiency of the conducted studies to determine the most effective conditions for using this method and are confident in its prospects [8]. Another area of application of gaze tracking systems is the control of manipulative robotic systems in production or, for example, for controlling a wheelchair or other assistive devices [9].

In the papers considered above [7–9], a change in the focus of the use of eye tracking on the issue of inclusion is observed but no ready-made software and hardware solutions are proposed to solve the problem.

In [10, 11], the features of the use of virtual and augmented reality tools are mainly concentrated on their application in blended and distance learning or on the development of software for eye tracking systems. For example, in work [12], a variant of such a system is proposed, based on the HOG (Histogram of Oriented Gradients) method and the linear machine learning algorithm SVM (Support Vector Machines) implemented in the Python programming language (The Netherlands) with the connection of the OpenCV (Open Source Computer Vision Library, USA), Dlib (D Library), NumPy (Numeric Python, USA) libraries.

Another area of research is the use of eye tracking systems during control activities in education. The studies reported in [13] tackle this issue. This work is more concerned with continuous authentication of students in the process of remote testing, rather than teaching inclusive students.

To meet the needs in the field of eye tracking, designers of virtual headsets have recently prepared a number of new products that provide significantly greater functionality of these devices and ease of use.

In March 2025, the Bigscreen company (USA) launched the next-generation models in the market: Beyond 2 and Beyond 2e [14]. These models have an improved system for adjusting devices to the features of the facial structure, as well as an expanded field of view, ensuring image clarity in all areas from the center to the edges. Changes have been made to the headset mounting system, which makes the face scanning operation for an individual interface unnecessary.

Headsets by Play For Dream (PRC) have a number of new devices and features:

– VR: Snapdragon XR2+ Gen 2 running Android 15 (USA);

– two micro-OLED (Organic Light-Emitting Diode) displays with a resolution of 3840 × 3552 and a frequency of 90 Hz;

– eye tracking;

– automatic adjustment of IPD (Interpupillary Distance), wired and wireless streaming from a personal computer (PC);

– a Meta Quest Pro-style battery located on the back panel;

– Meta Touch Plus – similar controllers [15].

Also noteworthy are the technological advancements by the following companies:

– HTC (Vive Focus Vision [16]), an updated version of the headset is aimed not only at business but also at ordinary users with an increased amount of RAM, color Passthrough, and built-in eye tracking;

– Samsung, together with Google and Qualcomm (Galaxy XR – the first headset on the Android XR platform [17]), the device is equipped with a Snapdragon XR2+ Gen 2 processor and 16 GB of RAM. The headset has micro-OLED displays with a resolution of 3552×3840 per eye, and a refresh rate of up to 90 Hz;

– Valve (Steam Frame [18]), a standalone VR headset Steam Frame, which runs on the SteamOS operating system with a Qualcomm Snapdragon 8 Gen 3 processor and 16 GB of RAM, has 2160×2160 LCD panels for each eye, pancake optics, variable IPD and a refresh rate of up to 120 Hz.

In addition to new hardware developments, it is worth noting the release of new software for eye tracking systems. Baballonia is a new generation of software from Paradigm Reality Enhancement Labs (USA) for tracking eye and face movements. It is a logical continuation and complete rewrite of the previous EyeTrackVR and Babble applications written in Python [19]. The development team designed Baballonia based on C# and the Avalonia framework, which provides

better performance, cross-platform compatibility, as well as a more flexible architecture for future extensions.

Modern assistive technologies based on digital technologies open up new opportunities for ensuring the accessibility of education. Such technologies include technologies based on voice control programs, electronic (on-screen) keyboards, a set of touch switches, and eye control systems.

As for the eye control system, it can be implemented in several ways. Among these techniques, two are worth highlighting:

– using a webcam connected to a personal computer (mobile or stationary);

– using virtual reality headsets (VR headsets).

The use of a webcam has a number of disadvantages, namely, dependence on the illumination of the room, the need to strictly observe the distance from the webcam, the presence of problems in tracking changes in the angle of view at small angles of change of view, etc.

The critical review of the literature [1–19] allows us to conclude that significant progress has been made in the development of eye-tracking technologies. However, there are issues that need to be resolved:

– technological dependence: most open source systems (for example, basic EyeTrackVR) require video transmission to a personal computer via Wi-Fi, which causes delays and leads to a decrease in student mobility;

– lack of complete autonomy: there is no open solution to ensure the implementation of all stages of eye-tracking (from frame reading to cursor control) only through the VR headset processor without external dependencies;

– limitations in the use of modern eye-tracking technologies in the practice of Ukrainian education, insufficient adaptation of international experience in the implementation of eye-tracking to the Ukrainian educational context.

The reasons for this state of affairs are the complexity of optimizing computer vision algorithms for mobile architectures and the lack of unified standards for integrating third-party hardware into VR environments. Thus, there is a need to design an autonomous architecture that would combine open hardware solutions with high-performance software adapted for mobile operating systems (Horizon OS/Android).

However, the systematization of the identified problems allows us to argue that it is advisable to conduct a study on designing an affordable, autonomous, and open eye-tracking system for mobile VR devices. Such a system should provide minimal latency and high accuracy in actual educational settings without the use of external computing resources.

3. The aim and objectives of the study

The aim of our research is to design a fully autonomous eye-tracking system for mobile VR headsets. This will allow for effective interaction with educational content for individuals with severe motor impairments regardless of the presence of external PCs or network infrastructure.

To achieve this goal, the following tasks were defined:

– to modernize the hardware part of the system (based on the EyeTrackVR project), including miniaturization of camera mounts and ensuring a stable wired connection to the VR headset;

– to develop a server software component in Rust, capable of using real-time neural network inference directly on the Qualcomm Snapdragon XR2 mobile processor (USA);

– to design and implement the OpenXR API layer, which provides transmission of gaze data directly to target educational applications with minimal latency;

– to design an immersive graphical user interface (GUI) for configuring and calibrating the system, accessible directly in the virtual space of the headset.

4. The study materials and methods

The object of our study is eye-tracking technology as a means of ensuring the accessibility of education for people with severe musculoskeletal disorders.

The principal hypothesis assumes that an autonomous eye-tracking system for mobile virtual reality headsets could make it possible to organize interaction with educational content for people with severe motor disorders regardless of the presence of external PCs or network infrastructure.

The most important factor in the development of software and hardware technologies is ease of use for the end user. If the technology causes discomfort to the user, works unreliably, or requires a significant number of actions, then regardless of the level of innovation or benefit of this technology, the user will probably not use it in practice.

That is why, among the main areas of research is the adaptation of the hardware and software components of the eye-tracking system on the VR headset to maximize its autonomous operation. That is, all hardware components are connected directly to the headset, no wireless communications such as Wi-Fi are used, and the eye tracking algorithms run on the headset's processor itself.

The need to use a wired connection between system components is due to a number of advantages they have compared to wireless communication channels:

– the claimed theoretical speed indicators of Wi-Fi connections (Table 1) are in practice much lower due to the presence of interference;

– a large number of neighboring Wi-Fi and other wireless networks generates channel noise, which negatively affects the speed and quality of signal transmission;

– the claimed signal transmission speed in Wi-Fi is divided by all connections (smartphones, laptops, IoT devices);

– delays in Wi-Fi connections are much higher than in USB connections, which can be critical when transmitting a series of images.

Comparative data on data transfer speeds according to different versions of Wi-Fi and USB connections are given in Table 1.

Table 1

Maximum data transfer speeds for various USB and Wi-Fi connection protocols

USB standard	Maximal speed	Wi-Fi standard	Maximal speed
USB 2.0	480 Mbps	Wi-Fi 4 (802.11n)	to 600 Mbps
USB 3.0 / 3.1 Gen 1	5 Gbps	Wi-Fi 5 (802.11ac)	to 6.9 Gbps
USB 3.2 Gen 2	to 10 Gbps	Wi-Fi 6 (802.11ax)	to 9,6 Gbps
USB 3.2 Gen 2x2	20 Gbps	Wi-Fi 6E (802.11ax)	to 11 Gbps
USB4 / Type-C	40–120 Gbps	Wi-Fi 7 (802.11be)	to 23 Gbps

In order for the system to meet the main criterion – ensuring autonomy, it was necessary to develop a server program, which precisely enables autonomy of the software component.

The server in the proposed system is a software component responsible for:

– communication with cameras and receiving an image stream;

– launching a neural CNN (Convolutional Neural Network) model for analyzing the eye image and calculating the direction of gaze;

– processing and filtering the data obtained from the neural network;

– sending the results of all calculations to the final application.

One of the steps to achieving autonomy of the gaze tracking system is compiling the server as an APK (Android Package Kit) application. This is possible due to the fact that Meta Quest headsets run on the Horizon OS operating system, which is built on the basis of the Android OS and is fully compatible with the installation of Android applications distributed in APK format.

The Rust programming language (USA), in which the server is written, and the ecosystem of packages and tools that have developed around this language make it possible to configure the compilation of the same project for different architectures and operating systems.

Naturally, some components of the system must have specific versions for the respective platforms, such as the component responsible for displaying the graphical interface. For example, when compiling as an OpenXR API layer, then the code that is written to display the interface using OpenXR is compiled.

Thus, deploying the server as a standalone Android application allows one to perform all stages of gaze processing without external computational dependences, which is a key step towards full system autonomy.

An OpenXR API Layer was previously designed that receives data about the gaze direction from the above-described server using the OSC (Open Sound Control) protocol and implements the OpenXR extension XR_EXT_eye_gaze_interaction. The Steam Link application was chosen to add this layer to it because it does not have any modification protection systems. The result is that Steam Link successfully recognizes support for tracking the direction of gaze and also makes requests to which the layer responds with the most current information received via OSC.

This approach allows us to implement user-specific setting panels, system status indicators, or notifications directly in the VR space without having to change the application code. The layer does not violate the application's logic but only "extends" the scene, supplementing it with its layer.

So, the graphical interface is designed using the following steps:

– interception of the OpenXR call xrCreateSession, which contains the necessary data structures for interaction and work with the device's graphics accelerator;

– call xrCreateSwapchain to construct a texture in video memory, into which the graphical interface will be rendered, and which will be displayed in space by the composer;

– interception of the call xrEndFrame to transfer to the execution system a list of graphic layers that the composer should display to the user. Adding your graphic layer with your own texture should be done here;

– calls xrCreateActionSet, xrCreateAction, xrSuggestInteractionProfileBindings, xrAttachSessionActionSets, xrGetActionStateBoolean, xrGetActionStateFloat, xrGetActionStateVector2f, xrGetActionStatePose, xrSyncActions, and others. These calls are necessary to work with the user input system according to the OpenXR documentation. Thanks to them, one can read the state of pressing buttons, triggers, stick movements, position and orientation in space, as well as other interactions with the controller.

In this way, user inputs are read and transmitted to subsequent software abstractions that process these inputs at the graphical interface level.

Real-time video streaming is demanding on the bandwidth and stability of the wireless channel. Any interruption in the connection or delay of even a few tens of milliseconds can lead to frame loss, which makes it impossible for the eye tracking algorithms to work stably. Using a wired connection completely eliminates this factor.

For this purpose, a method of directly connecting ESP32S3 microcontrollers to the headset using a USB connection was used.

Let's consider the software and hardware parts of the proposed system.

From a software point of view, the microcontrollers are equipped with the open OpenIris software, the task of which is to communicate with the camera itself, read frames, and then transmit them. In addition to Wi-Fi mode, OpenIris also provides the ability to select USB mode, under which the microcontroller does not launch the Wi-Fi module but transmits images using a virtual serial port.

From the headset side, this communication option is possible because the Android OS has full support for connecting third-party USB devices, as well as virtual serial ports.

Meta Quest headsets, and in general autonomous virtual reality headsets, have only one USB port. Since one needs to connect at least two USB devices (one connection per microcontroller for each eye), one can use a USB hub.

To be able to simultaneously charge the headset and communicate with USB devices, a USB splitter is used, which has two physical ports: for charging and for data transfer, and combines them into one output port, which is connected to the headset.

The transition from wireless to wired communication allows one to completely eliminate the system's dependence on Wi-Fi networks, increase the stability of image transmission, and simplify the setup process.

5. Results of the study on designing an autonomous eye tracking system for inclusive education

5. 1. Modernization of the hardware component for inclusive use

The hardware part is built on the basis of the Meta Quest 2 headset and includes the following:

– frames mounted on the headset lenses for infrared illumination of the eye and installation of a camera in the frame;

– two boards with a Seeed Studio XIAO ESP32S3 Sense microcontroller (China) with a module for connecting a camera;

– two OV2640 camera modules without an infrared filter.

The main goal within the hardware component of the system was to reduce the dimensions and thereby increase comfort for the user.

Improvement of the frames mounted on the headset lenses: the rejection of boards with infrared LEDs, which were included in the "V4 mini Fully Solder-less LED Kit", and the transition to mounting LEDs themselves, allowed us to significantly reduce the size of the system in the plane facing the user's face.

In addition to reducing dimensions, there are other advantages to using LEDs:

1. The ability to turn the LED towards the pupil as the smaller dimensions no longer prevent this. This improves the quality of illumination of the eye.

2. Ease of fixation and formation of a protective layer using superglue.

To switch to LEDs, a new 3D model was built with subsequent manufacturing on a 3D printer.

To fix the LED in place and also as a protective layer, superglue was used (Fig. 1).

The frame itself was also adapted to better fit in areas of unevenness. To do this, a frame model was constructed in the 3D editor Blender. Owing to this, the frame is even further away from the user's face (Fig. 2).

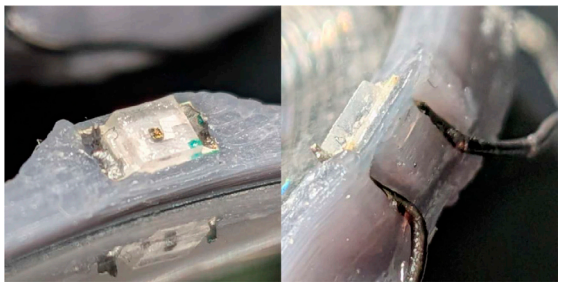

Fig. 1. Infrared LED soldered to the control system and fixed to the frame

Fig. 2. Improved frame model in color lighting on four sides for better visibility of added curvatures and final shape

The camera uses a module based on the OV2640 sensor; the module with a viewing angle of 130° was chosen instead of 160°. The application of this camera module is due to two reasons:

1. The 130° module is much smaller in size than the 160° module, which allows one to place the camera closer to the lens and get a slightly better angle of view of the eye.

In the frame of the 160° module, a lot of excess information enters the neural network input than is needed to determine the direction of gaze. With the 130° module, the image of the eye covers most of the frame, which gives better detail after cropping (Fig. 3).

The disadvantage of the 130° module is the lack of an option without an infrared filter. Therefore, the filter had to be removed.

USB hub for cameras: as a result of searching online catalogs of various USB devices, a set of USB splitters was found that solve the problem of simultaneous data transfer and charging.

In order for the splitter not to increase the horizontal size of the headset, a USB Type-C extension cable was also chosen, the connector of which has a 90° rotation and is connected to the headset to move the splitter away from the front of the headset (Fig. 4).

a *b*

Fig. 3. Comparison of images obtained from the camera module at different viewing angles: *a* — 130°; *b* — 160°

Fig. 4. USB splitter near the back of the headset, which connects on the left side — extension cable to the headset, above — cable to the USB hub, on the right — charger

The next step was to connect two microcontrollers with cameras, as well as an infrared illumination system, to a single USB Type-A port. 5 USB Type-C connectors were soldered to the USB hub, in which a short piece of cable was also added between the USB hub input connector and the hub itself for additional convenience of mounting to the headset. The other four USB connectors are directly soldered to the hub (Fig. 5).

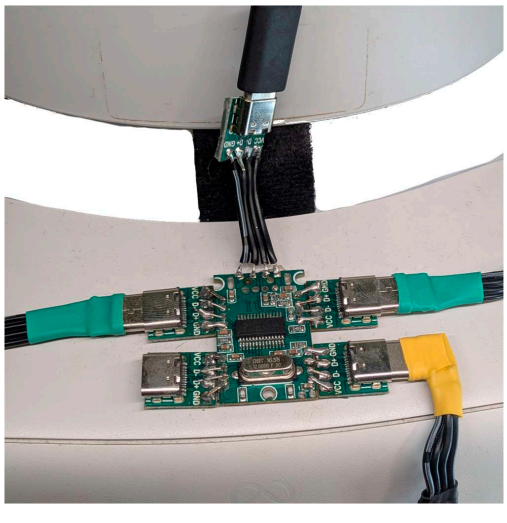

Fig. 5. USB hub that connects to: top — cable to USB splitter, yellow cable — power for infrared illumination, two green cables — ESP microcontrollers with cameras

The general view of the headset with connected additional devices is shown in Fig. 6.

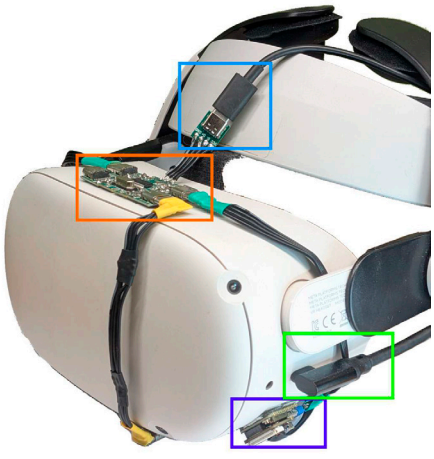

Fig. 6. External hardware

In Fig. 7, the USB hub is highlighted in orange, the cable from the USB hub to the USB splitter is highlighted in blue, one of the microcontrollers is highlighted in purple, and the cable from the headset to the USB splitter is highlighted in green.

In order to reduce the amount of computing resources required to run the model, it was decided to reduce the size of the input image from 128 × 128 to 64 × 64 pixels. Owing to this, the size of the model decreased from 3.15 MB to 675 KB without a significant loss of accuracy in determining the direction of gaze.

5. 2. Design of an autonomous server in Rust

The innovation in this work is the integration of the API layer into the overall server architecture, which allows one server project to be compiled into a single SO (Shared Object) file, which simultaneously plays the role of both the API layer and the server.

This means that when OpenXR requests the direction of view, no network communications are required, and the most current response to the request is already in the process's memory. This provides an advantage in the form of reduced latency because the intermediary (PC) disappears from the communication chain, compared to the previous work (Fig. 7).

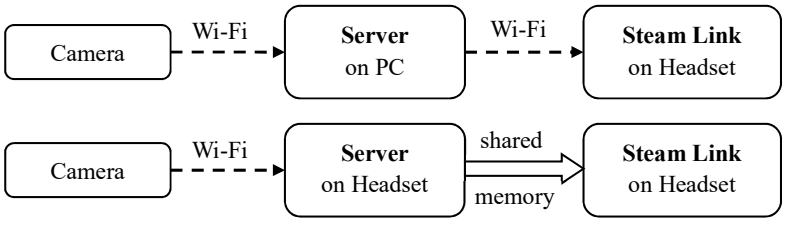

Fig. 7. Schemes of signal transmission from the camera to the Stream Link with the server located on a personal computer (top diagram) or directly in the headset (bottom diagram)

The server, written in the Rust programming language, has been significantly expanded by integrating the API layer and the ability to compile as a .so (shared object) library for Android. To do this, a number of changes were made to the compilation process for different platforms.

A total of 5 compilation options were defined:

1) *inference* is the name of the process of launching an already trained neural network model and obtaining the calculation result. This module adds the ort and ort-sys dependences, and is also responsible for loading the CNN model from disk, launching the calculation process based on ONNX Runtime, and reading the results for further transmission to subsequent modules;

2) *gui* is responsible for displaying the graphical user interface based on the ImGui library for convenient creation of graphical interfaces and *wgpu* for interaction with the video card, as well as *imgui-wgpu* that connects these two technologies. When compiling for Windows, it forms a regular window, and for Android, OpenXR calls are intercepted and their own layer with a graphical interface in VR space is added to them;

3) *openxr-api-layer* is responsible for all necessary entry points for the API layer to work, as well as all the code containing the OpenXR call processing logic;

4) *android* includes jni dependences for working with the Java Native Interface in Rust, as well as *android-usbser* for connecting to serial ports of USB devices within the Android OS;

5) *desktop* targets compilation for PC platforms and includes the nokhwa library that allow one to connect to webcams. This functionality is not used in this work but owing to it, one can use cameras, for example, Bigscreen Beyond 2e.

So, the final project can be compiled both into a PC application and for integration into Android applications as an OpenXR API layer. The difference between these two results is only in changing a few compilation parameters.

The compilation command for a PC application takes the form

```
cargo build --features gui, desktop, inference --bin.
```

The features parameter allows one to specify a list of compilation options that will be enabled for compilation, and bin indicates that the final result should be an executable file.

Instead, to compile the Android version, the following command is executed

```
cargo dinghy -p auto-android-aarch64-api32 build --features openxr-api-layer, android, gui, inference --lib.
```

Therefore, the ability to use the same code for different end applications simplifies development. There is a significant overlap in functionality between them, which would otherwise force code duplication in different projects and their constant updating. Another advantage is the ability to simultaneously obtain software primarily for autonomous headsets that have their own computing capabilities and to which cameras are connected, as well as for headsets that are completely dependent on a PC and do not have their own system.

5. 3. Integration via OpenXR API Layer

The architecture of the bridge between the server components and the API layer is based on the continuous transmission of the gaze state from the data processing subsystem to the OpenXR query response mechanism. The server receives data from the cameras, performs their pre-processing, inference and filtering, and then forms a unified internal representation of the gaze parameters. It consists of a common value of the gaze angle along the vertical axis (pitch) for both eyes, a separate value of the angle along

the horizontal axis for each eye (l_yaw, r_yaw), and a separate value of its openness for each eye (l_eyelid, r_eyelid).

Architecturally, this interaction ensures the independence of the layer from the internal structure of the server part. The API layer does not perform calculations and does not work with camera streams but receives only the already interpreted gaze state. As a result, the server can scale, change inference models or implement additional filtering methods without the need to modify the OpenXR integration itself. On the other hand, the API layer has the ability to precisely and uniformly adapt the provided state to the requirements of OpenXR:

– apply transformations between spaces;

– set validity indicators;

– take into account the time requested by the application;

– generate a response to the request in accordance with the requirements of the extension.

This separation forms a flexible architecture in which the bridge serves as the central mechanism for synchronization and data transfer between the server and the API layer (Fig. 8).

In Fig. 8, the arrows show the general direction of data flow from reading it by the server to displaying the results.

After starting active work on integrating the API layer into the server, it became clear that it was necessary to find a tool that would make it possible to automate the execution of a number of commands within the Cargo build system, which is part of the Rust language. *Cargo-make* was chosen.

Cargo-make reads its configuration from the Makefile.toml file in the root of the project, which describes all the execution steps, as well as the sequence of execution of certain commands. An *apk* script was written within the project, which is conveniently launched from the terminal by calling *cargo make apk*.

As a result, it was possible to obtain a fully automated tool for modifying and installing APKs, which significantly reduced development time and minimized the number of errors. The *cargo make apk* command integrates external tools (Apktool, zipalign, apksigner, adb) into a single, integrated system.

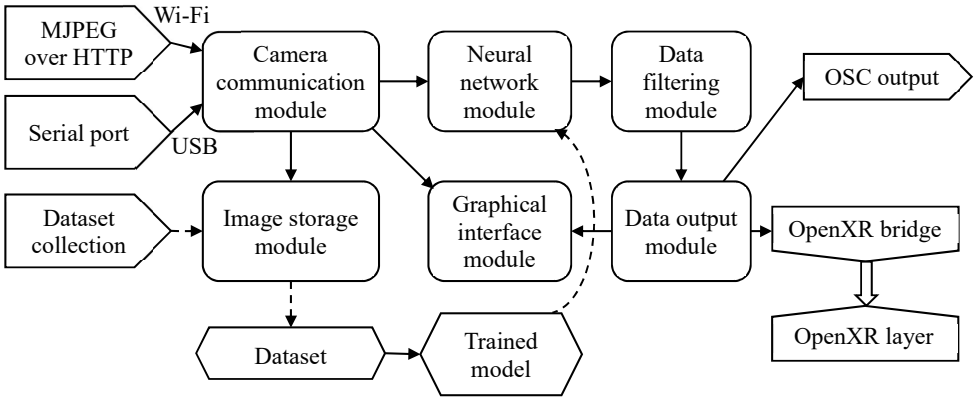

Fig. 8. Server architecture diagram

5. 4. Design of an immersive settings interface

For effective user interaction with the system, a graphical interface (GUI) based on the ImGui library was developed, integrated directly into the VR space (Fig. 9). Since a user with mobility impairments cannot independently switch to an external monitor, all eyetracker controls must be accessible "inside" the headset.

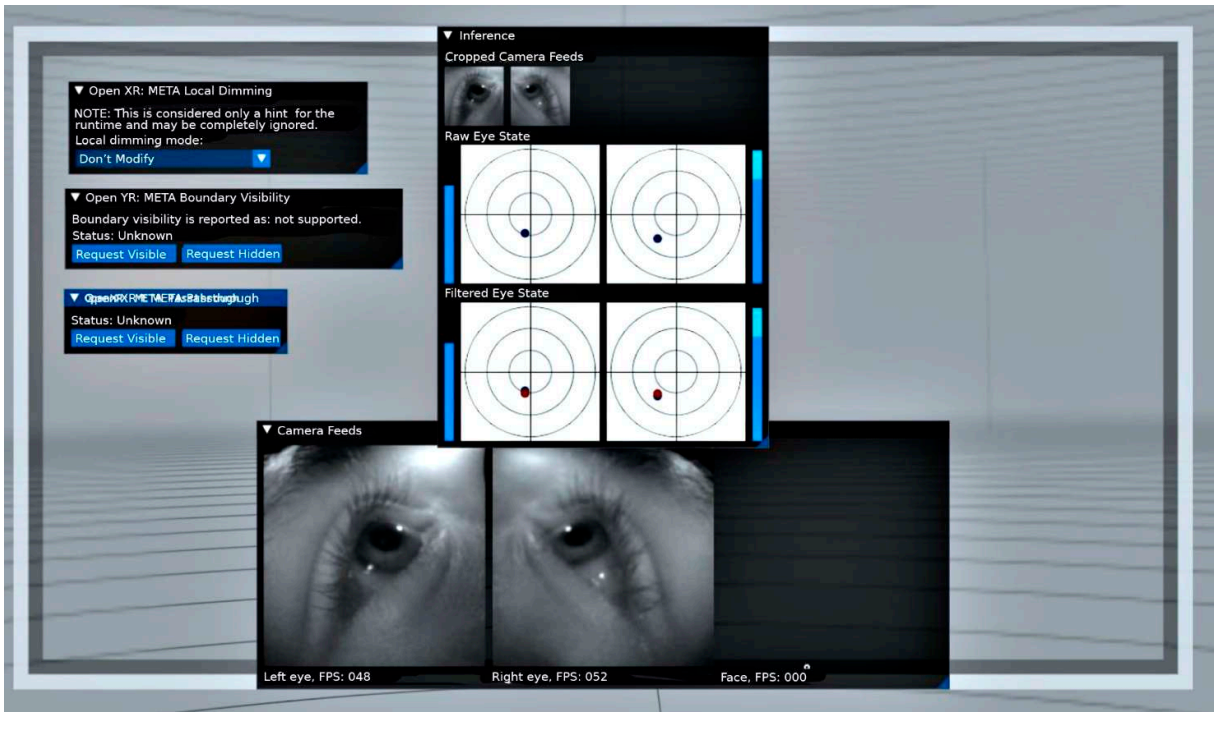

Fig. 9. Screenshot of the headset with the GUI enabled, in which the two central windows show the view from the cameras, the input image for the model, and its results before and after filtering

The GUI system allows the assistant teacher to do the following:

– see "with the camera's eyes" to make sure that the headset is worn correctly;

– run the calibration procedure;

– adjust "dead zones" and cursor sensitivity;

– check the connection status of the left and right eyes.

The implementation of the specified functionality is based on the use of OpenGL ES call interception. A separate graphic layer is superimposed on top of any running application where the interface rendering takes place. This provides convenient access to the system settings at any time of its use.

6. Discussion of the results of design of an autonomous gaze tracking system for inclusive education: results and summary

As a result of our study, an assistive system was designed that is high-tech, based on open software and available hardware components. A significant achievement is enabling the full autonomy of the gaze tracking system. Effective results were achieved due to the following factors.

The autonomy of the system implies that the entire gaze tracking system (hardware and software components) is placed on the headset and does not require the transmission of a signal with images for further processing to a personal computer (Fig. 2, 3, 5, 6). This enables reliability, speed, and the ability to quickly adjust the operation of the gaze tracking system if there is a need to change the location, communication system, computer device, or software with which the system works.

The programming language (Rust) and system architecture were successfully chosen (Fig. 7, 8). The choice of Rust, which avoids the delays typical of garbage-collected languages (such as Java or C#) and provides direct access to hardware accelerators, has led to the high efficiency of the designed system. Ensuring intuitive and natural interaction is based on the fact that instead of a separate application, the OpenXR API Layer is used. This guarantees the transmission of gaze data with the minimum possible delay (<2 ms from the moment of completion of the calculation).

It is worth noting that the OpenXR layer is executed within the same process as the application to which this layer was added. This gives an advantage over the option of a separate APK application since Android systems historically have built-in mechanisms for prioritizing active applications and slowing down or disabling others to save system resources.

Another significant advantage is the server's direct access to the OpenXR runtime. For example, one can find out with the minimum possible delay how many milliseconds in the future a request is made for the user's gaze direction. Or, one can add own graphic elements to the virtual space and check whether the user presses the control buttons on the controller.

This makes it possible to design a graphical user interface in the virtual space of the OpenXR application (Fig. 9). After all, if the layer can make any changes to OpenXR calls without restrictions and make its own, then there are no obstacles to adding convenient ways for the user to interact with the functionality and settings of the layer.

Commercial systems such as Tobii are expensive to use, costing between US$ 4,000 and US$ 12,000 for educational institutions. However, the prototype designed in our research is based on equipment costing up to US$ 200 (excluding the cost of the headset itself). This makes the described technology accessible even to small inclusive centers, schools, and individuals in regions and countries with low levels of education and healthcare funding.

Unlike systems based on the use of, in particular, webcams, the proposed technology in the VR headset does not depend on the lighting in the room since its own IR diodes are used to illuminate the eyes inside the closed space of the headset. Owing to this, the system could be used both in shaded rooms and in bright sunlight from the window. It is also worth noting that the wired connection (USB CDC) turned out to be much more reliable than Wi-Fi, which is often "clogged" in school networks, causing freezes and user disorientation.

The interpretation of the results gives grounds to assert that the chosen path of autonomy based on Rust and OpenXR is promising for the development and implementation of mass and accessible assistive technologies in the future for the educational process.

Along with the successful implementation of the research tasks, certain limitations on the use of the system were identified:

1. The system setup stage requires the participation of specialists who are familiar with the system or have experience in using similar systems. This especially applies to the operation of removing the IR filter. This complicates the process of mass implementation of the system.

2. The need for measures to ensure the confidentiality of data obtained as a result of the use of biometric gaze data, which can be used as access elements to other confidential data. Their processing directly on the device is an advantage; however, when transmitted to the network, it is necessary to ensure their protection in accordance with current legislation.

Using a system based on a VR headset for more than 45–60 minutes can cause eye fatigue and general fatigue of the body.

Future research should be aimed at integrating the system with artificial intelligence to implement adaptive learning. In particular, analysis of reading speed and temporal patterns of gaze movement can provide automatic adjustment of the complexity of text material or dynamic selection of key fragments for students with attention disorders or dyslexia. In this context, the use of cascade neural network architectures for the analysis of temporal patterns of gaze based on two-cascade models is promising [20]. In them, the first cascade extracts the basic features of eye movements, and the second one interprets them in cognitive and educational contexts. Also promising is the transition to cameras with a global shutter in order to eliminate image blurring during rapid saccadic eye movements. In addition, further research could be aimed at integrating assistive VR technologies for gaze control with interactive information systems for public purposes, taking into account the approaches to building visual platforms proposed in [21, 22].

Difficulties in the advancement of research may be associated with problems of financing both the research itself and production and educational institutions for the implementation of the results.

7. Conclusions

1. The hardware part of the system was modernized based on the EyeTrackVR project. First of all, this is the miniaturization of the camera mounting system (construction of models of thin mounting frames in the Blender3D graphic ed-

itor and their manufacture using a 3D printer). A wired communication system using a USB connection was designed, using the same connection to recharge the system, which increased the battery life. Such modernization provided an increase in the signal transmission speed from the cameras to the image processing system by 1.5 times compared to wireless connections and enabled the system's operation under offline mode without connecting to a personal computer.

2. The server part, written in Rust, was successfully adapted for cross-compilation under the Android architecture (AArch64) and integrated into the OpenXR API layer format. This allowed all computational processes (image reading, CNN inference, filtering) to be run directly on the headset processor, minimizing latency and ensuring full system autonomy.

3. The developed API layer successfully intercepts system calls and supports three key OpenXR extensions: XR_EXT_eye_gaze_interaction, XR_FB_eye_tracking_social and XR_FB_face_tracking2. A linear transformation of eye openness and gaze direction data into standardized blendshapes was implemented. An improved rendering system was developed and integrated, which uses intercepted OpenGL ES and xrEndFrame calls to display a full-fledged graphical interface based on ImGui directly in VR space.

4. An immersive graphical user interface (GUI) was designed, which was successfully tested for system configuration and calibration. A feature of the designed interface is that the GUI became available directly in the virtual space of the headset.

Conflicts of interest

The authors declare that they have no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study, as well as the results reported in this paper.

Funding

The study was conducted without financial support.

Data availability

The data will be provided upon reasonable request.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

Authors' contributions

Andrii Symkin: Project administration, Conceptualization, Methodology, Software, Resources, Data Curation, Visualization, Formal analysis, Investigation, Writing – review & editing; **Oleksandr Mitsa:** Project administration, Supervision, Formal analysis, Writing – review & editing; **Yurii Horoshko:** Writing – original draft, Writing – review & editing, Formal analysis, Investigation; **Hanna Tsybko**: Writing – original draft, Writing – review & editing, Validation, Formal analysis, Investigation; **Viktor Shakotko**: Conceptualization, Resources, Writing – original draft, Writing – review & editing, Validation, Formal analysis, Visualization.

References

1. Nyström, M., Holmqvist, K. (2010). An adaptive algorithm for fixation, saccade, and glissade detection in eyetracking data. Behavior Research Methods, 42 (1), 188–204. https://doi.org/10.3758/brm.42.1.188
2. Holmqvist, K., Nyström, M., Andersson, R., Dewhurst, R., Jarodzka, H., van de Weijer, J. (2011). Eye Tracking: A Comprehensive Guide to Methods and Measures. Oxford, 560.
3. Kapp, S., Barz, M., Mukhametov, S., Sonntag, D., Kuhn, J. (2021). ARETT: Augmented Reality Eye Tracking Toolkit for Head Mounted Displays. Sensors, 21 (6), 2234. https://doi.org/10.3390/s21062234
4. Holmqvist, K., Örbom, S. L., Hooge, I. T. C., Niehorster, D. C., Alexander, R. G., Andersson, R. et al. (2022). RETRACTED ARTICLE: Eye tracking: empirical foundations for a minimal reporting guideline. Behavior Research Methods, 55 (1), 364–416. https://doi.org/10.3758/s13428-021-01762-8
5. Šmidekova, Z., Minarikova, E., Janík, M., Holmqvist, K. (2019). Using eye-tracking to investigate differences in teachers' professional vision IN action and ON action. Journal of Eye Movement Research, 12 (7), 260. Available at: https://openurl.ebsco.com/EPDB%3Agcd%3A7%3A10210567/detailv2?sid=ebsco%3Aplink%3Ascholar&id=ebsco%3Agcd%3A142777840&crl=c&link_origin=scholar.google.com
6. Sim, G., Bond, R. (2021). Eye tracking in Child Computer Interaction: Challenges and opportunities. International Journal of Child-Computer Interaction, 30, 100345. https://doi.org/10.1016/j.ijcci.2021.100345
7. El Moucary, C., Kassem, A., Rizk, D., Rizk, R., Sawan, S., Zakhem, W. (2024). A low-cost full-scale auto eye-tracking system for mobility-impaired patients. AEU - International Journal of Electronics and Communications, 174, 155023. https://doi.org/10.1016/j.aeue.2023.155023
8. Sugianto, M., Zhou, Y., Qiu, J. (2025). Eye tracking-based dual task in rehabilitation of motor and cognitive function in post-stroke patients: a literature review. Bulletin of Faculty of Physical Therapy, 30 (1). https://doi.org/10.1186/s43161-025-00295-x
9. Sunny, M. S. H., Zarif, M. I. I., Rulik, I., Sanjuan, J., Rahman, M. H., Ahamed, S. I. et al. (2021). Eye-gaze control of a wheelchair mounted 6DOF assistive robot for activities of daily living. Journal of NeuroEngineering and Rehabilitation, 18 (1). https://doi.org/10.1186/s12984-021-00969-2
10. Bogachkov, Y., Uhan, P. (2025). Models of blended learning in the context of mixed processes. Information Technologies and Learning Tools, 107 (3), 239–260. https://doi.org/10.33407/itlt.v107i3.5920

11. Batsenko, S., Bohachkov, Yu., Burov, O., Horbachenko, V., Korkishko, I., Lytvynova, S., Nosenko, Yu. ta in. (2025). Imersyvni tekhnolohiyi dlia pidtrymky zmishanoho navchannia v zakladakh zahalnoi serednoi osvity. Kyiv: ITsO NAPN Ukrainy, 174. Available at: https://lib.iitta.gov.ua/id/eprint/746953/1/ПОСІБНИК.pdf
12. Sedinkin, O., Derkach, M., Skarga-Bandurova, I., Matiuk, D. (2024). Systema dlia vidstezhennia rukhu ochei na osnovi mashynnoho navchannia. Computer-integrated technologies: education, science, production, 55, 199–205. https://doi.org/10.36910/6775-2524-0560-2024-55-25
13. Barkovska, O., Liapin, Y., Muzyka, T., Ryndyk, I., Botnar, P. (2024). Gaze direction monitoring model in computer system for academic performance assessment. Information Technologies and Learning Tools, 99 (1), 63–75. https://doi.org/10.33407/itlt.v99i1.5503
14. Introducing Bigscreen Beyond 2 with next-gen optics and eyetracking. Available at: https://store.bigscreenvr.com/blogs/beyond/introducing-bigscreen-beyond-2
15. My interview with Play For Dream about their headset, Android XR, enterprise licensing, China, and more! Available at: https://skarredghost.com/2025/06/05/play-for-dream-interview-android-xr/
16. Vive Focus Vision Announced. Available at: https://www.uploadvr.com/vive-focus-vision-announced/
17. Introducing Galaxy XR: Opening New Worlds. Available at: https://news.samsung.com/us/introducing-galaxy-xr-opening-new-worlds/
18. Valve Officially Announces Steam Frame, A “Streaming-First” Standalone VR Headset. Available at: https://www.uploadvr.com/valve-steam-frame-official-announcement-features-details/
19. Project-Babble/Baballonia. Available at: https://github.com/Project-Babble/Baballonia
20. Geche, F., Mitsa, O., Mulesa, O., Horvat, P. (2022). Synthesis of a Two Cascade Neural Network for Time Series Forecasting. 2022 IEEE 3rd International Conference on System Analysis & Intelligent Computing (SAIC), 1–5. https://doi.org/10.1109/saic57818.2022.9922991
21. Lupei, M., Shlahta, M., Mitsa, O., Horoshko, Y., Tsybko, H., Gorbachuk, V. (2022). Development of an Interactive Map Within the Implementation of Actual State and Public Directions. 2022 12th International Conference on Advanced Computer Information Technologies (ACIT). https://doi.org/10.1109/acit54803.2022.9913191
22. Mitsa, O., Sharkan, V., Maksymchuk, V., Varha, S., Shkurko, H. (2023). Ethnocultural, Educational and Scientific Potential of the Interactive Dialects Map. 2023 IEEE International Conference on Smart Information Systems and Technologies (SIST), 226–231. https://doi.org/10.1109/sist58284.2023.10223544