Ingénierie des Systèmes d'Information
Vol. 29, No. 4, August, 2024, pp. 1343-1355
Journal homepage: http://iieta.org/journals/isi

Exploring Attentive User Interface Input via Raspberry Pi, based on Face Landmark Detection, Eye Open-Closed Detection and Head Movements Detection

Muhamad Yusvin Mustar[1,2], Rudy Hartanto[1*], Paulus Insap Santosa[1]

[1] Department of Electrical Engineering and Information Technology, Universitas Gadjah Mada, Yogyakarta 55281, Indonesia
[2] Department of Electrical Engineering, Universitas Muhammadiyah Yogyakarta, Yogyakarta 55183, Indonesia

Corresponding Author Email: rudy@ugm.ac.id

https://doi.org/10.18280/isi.290410

Received: 26 January 2024
Revised: 6 June 2024
Accepted: 30 July 2024
Available online: 21 August 2024

Keywords:
AUI, Raspberry Pi, face landmark detection, eye open-closed detection, head movements detection

ABSTRACT

Attentive User Interface (AUI), especially vision-based AUI systems, is a highly developed area of study. Cameras play a crucial role in modeling AUI inputs to monitor and understand user attention during interaction. This enables the optimization of interaction tasks between humans and machines. Typically, AUI systems are designed for desktop computers, employing various detection methods based on computer vision. This research introduces the exploration of AUI input through a low-cost Raspberry Pi 4 embedded system a small-sized system with limited voltage sources. This exploration serves as a reference for AUI design on embedded systems. The proposed AUI input design generating 8 AUI input categories. These include face detection, open or closed eye condition detection, and detection of forward, up, down, left, and right view directions. This study discusses three scenarios of AUI input modeling using various methods to identify the most effective design through Raspberry Pi 4. Based on experimental results and a series of tested tasks, Scenario 3, based on Mediapipe, yielded the best results with an average FPS value of 10 and 100% accuracy for facial landmark detection, detection of open eye conditions, detection of forward, up, down, right, and left gaze directions at various detection reading angles.

1. INTRODUCTION

AUI is a user interface designed to enhance user attention capacity, allowing for the perception, modeling, and optimization of user attention status. This ensures optimal distribution of information processing resources between the user and the system across a range of interaction tasks [1-4]. Specifically, AUI monitors user behavior, models goals and interests, anticipates needs, provides information, and strategically optimizes communication between humans and machines using sensing, communication, augmentation, control, and human availability [4, 5].

In its evolution, AUI is linked with the Perceptual User Interface (PUI), which combines multimodal input, multimedia output, and human-like perceptual capabilities, facilitating natural interaction. The crux of AUI lies in the modeling of users and the conclusions drawn about them [5]. There are four main types of AUI, including (1) visual attention, (2) turn management, (3) interruption decision interfaces, and (4) visual detail management interfaces [1, 3]. AUI research has found application in various fields such as human-computer interaction (HCI), robot control, and other interaction systems [6-10], with the primary goal of maximizing natural interaction based on user attention.

A prominent contemporary development in AUI systems is the vision-based or computer vision approach. This development is closely tied to advancements in vision-based sensor devices, utilizing camera devices to capture AUI input reflecting user behavior or visual attention.

A prominent contemporary development in AUI systems is the vision-based or computer vision approach. This development is closely tied to advancements in vision-based sensor devices, utilizing camera devices to capture AUI input reflecting user behavior or visual attention [3, 5, 10]. Several previous studies have explored vision-based AUI systems, incorporating input from facial expressions and eye movements [4], eye tracking [11, 12], and eye gaze [13]. Additionally, recent studies have shown that the application of eye gaze-based AUI system input [14, 15] can monitor and understand user behavior and attention during interaction tasks. Consequently, the application of AUI can enhance user interaction with the system compared to interfaces that do not implement AUI. These systems are typically designed for desktop computers, often equipped with graphics processing units (GPU). However, for devices requiring smaller form factors and low-voltage, cost-effective sources, the use of desktop computers proves relatively inefficient. Thus, there is a need for experiments to design vision-based AUI inputs for alternative platforms, so that the application of AUI can be more widely applied as a system that can optimize interaction tasks.

Referring to device considerations, this research proposes an experiment involving a vision-based AUI input design on the Raspberry Pi embedded system, distinguishing it from desktop computers. The Raspberry Pi is chosen for its multiprocessing capabilities, power, programmability with

open-source software, Linux operating system compatibility, low cost, low power consumption, portability, and compact size [16-19]. In addition, several studies explain that the Raspberry Pi supports computer vision-based systems and is considered very reliable [20, 21], so the exploration of AUI input through the Raspberry Pi needs to be done by combining different inputs.

Compared to other studies [4, 11-15], this research proposes an exploration of AUI input design based on different input modalities, encompassing face landmark detection results, open and closed eye detection, and vision-based head movement detection. This design is intended for use as AUI input in the visual attention system design computer vision-based. Therefore, the design experiment for AUI input based on vision for embedded Raspberry Pi systems must be conducted to ensure that it can support various systems that require small media, limited voltage sources, and low cost. Furthermore, the accuracy of AUI input detection in real-time scenarios will be tested with the experiments conducted on the Raspberry Pi 4 embedded system. This is undertaken to assess and determine the optimal performance of the proposed AUI input via Raspberry Pi. Figure 1 illustrates the Raspberry Pi 4 Model B utilized in this experiment.

Figure 1. Raspberry Pi 4 Model B

2. METHODS

2.1 Hardware design

To build and test the proposed AUI input system, various supporting hardware components are required, as illustrated in Table 1. The system design block diagram is depicted in Figure 2.

Table 1. Hardware

No.	Hardware	Specification
1	Raspberry Pi 4 Model B	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz, 4GB LPDDR4-3200 SDRAM
2	Memory Card	SanDisk Ultra Class 10 32 GB
3	Monitor	23 inch
4	Camera	Webcam Logitech C525
5	Power Adapter	5 Volt 3A
6	Mouse and Keyboard	Type Wireless

As depicted in Figure 2, the camera employed to capture visual attention is a C525 Logitech webcam. The Raspberry Pi 4 Model B serves as a single-board computer (SBC), operating on the Raspbian 10 (Buster) operating system, with a 5-volt 3-ampere voltage source. A 23-inch monitor functions as the display screen, while wireless keyboard and mouse are utilized as interface media for program creation, execution of algorithms, and cursor navigation on the Raspberry Pi.

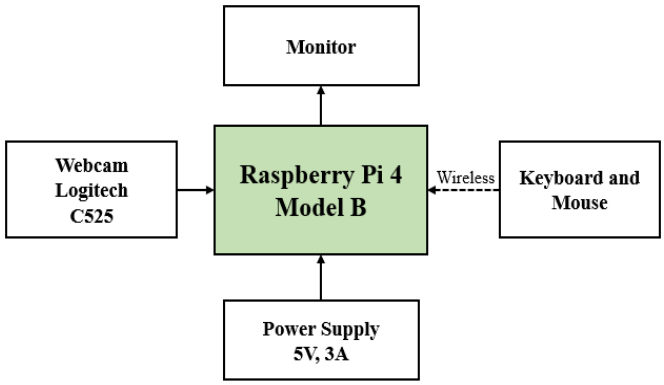


Figure 2. Hardware design

2.2 System design

The system in this research runs on the Raspberry Pi 4 embedded system by developing algorithms and programs that extensively utilize the Python 3.7 programming language, OpenCV 4.1.0 library, Dlib machine learning (ML) library, and Mediapipe ML library. The OpenCV library is the most well-known computer vision library, which builds on algorithms that run in real-time, such as the Viola-Jones technique based on Haar fature-based cascade classifiers for object detection [22, 23]. Dlib is a cross-platform toolkit based on C++ programming that can be used as ML tools and for the creation of algorithms. Dlib includes several modules mainly focused on ML, deep learning, and image processing [24], while Mediapipe is a set of libraries, models, and methods trained using ML so that it can be used for face identification in images, facial landmark detection, body pose tracking, and object recognition [25, 26]. In its current development, the OpenCV library, Dlib ML library, and Mediapipe ML library are widely used as tools in designing computer vision-based systems [27-31].

In this research, various methods employed in the design of facial landmark detection, open and closed eye condition detection, and head movement detection will be compared. These methods include face detection using Haar cascade classifiers and Dlib [32, 33], Dlib face landmark detection [34-36], and Mediapipe face landmark detection [37, 38]. Facial landmark detection plays a crucial role in the design of this AUI input system. Therefore, a comparison between Dlib face landmark detection and Mediapipe face landmark detection will be conducted. This comparison is necessary as the number of facial landmark detections between Dlib and Mediapipe differs. Dlib-based landmark detection results in 68 face landmark detections, while Mediapipe landmark detection results in 468 face landmark detections. However, it is worth noting that the utilization of Dlib face landmarks requires prior face detection, which is different from Mediapipe-based facial landmark detection. Figure 3 illustrates the AUI input design consisting of facial landmark detection, eye open-close detection, and head movement detection.

As depicted in Figure 3, there are 8 AUI inputs, each

assigned a detection code (a to h), indicating the outcome of the AUI input detection process. Facial landmark detection holds a significant role in the proposed design for AUI inputs. Two prominent methods of facial landmark detection, namely using Dlib and Mediapipe, are employed and tested due to their popularity in the field of computer vision. In this study, the outcomes of facial landmark detection serve as input information for AUI detection. Figure 4 illustrates facial landmark detection based on Dlib, while Figure 5 showcases facial landmark detection based on Mediapipe.

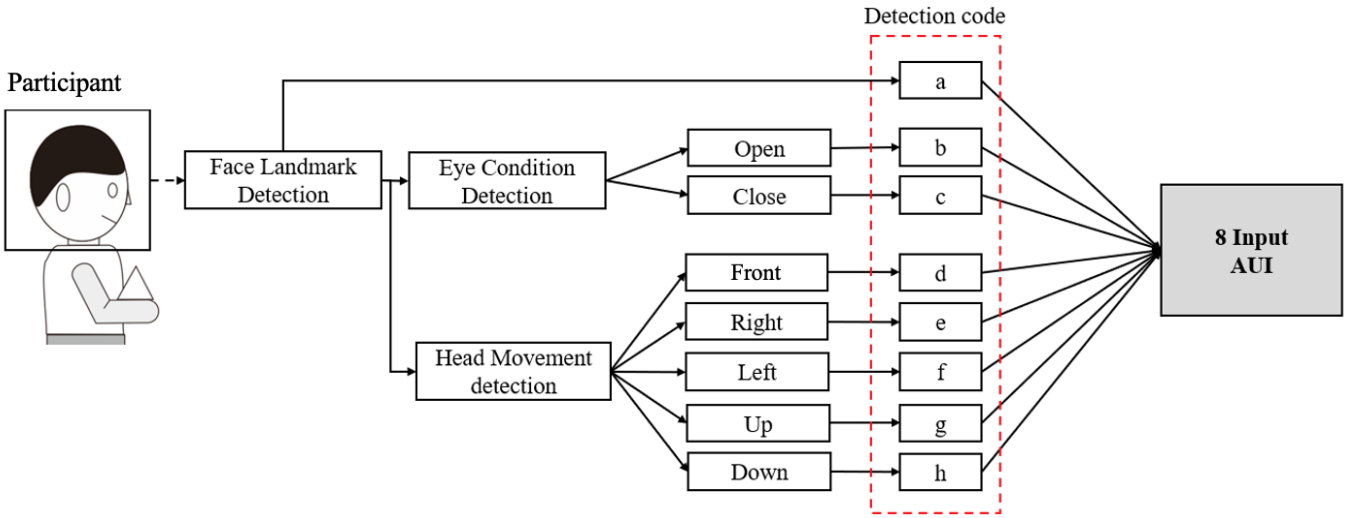


Figure 3. Design of the AUI input

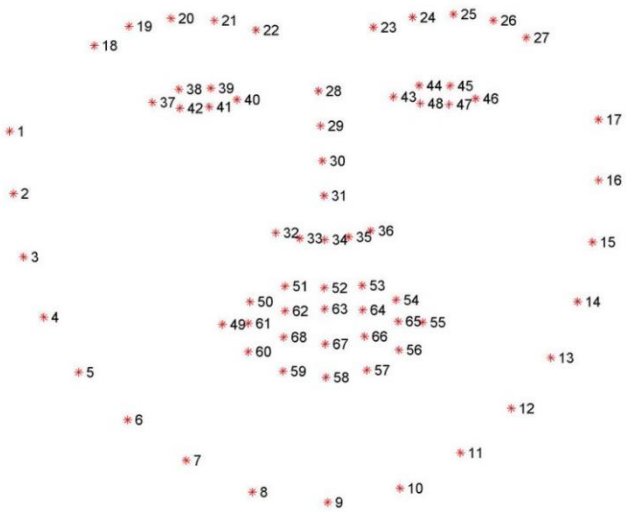


Figure 4. Dlib based 68 landmarks [36]

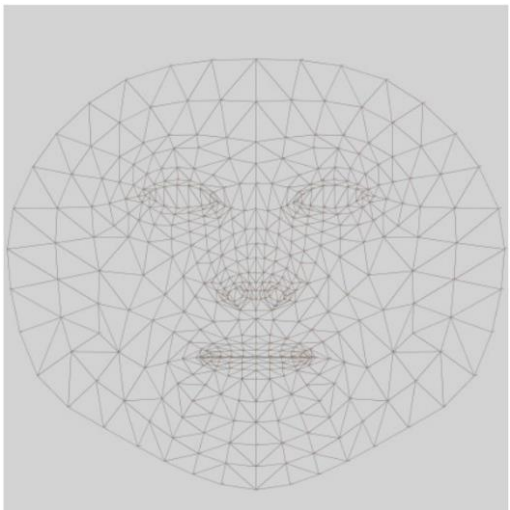


Figure 5. Mediapipe based 468 landmarks [38]

Since the study involves the utilization, testing, and comparison of face landmark detection based on Dlib and Mediapipe as inputs for the AUI, the experiment has been categorized into several scenarios, including:

Scenario 1: AUI input based on Dlib detection of 68 landmarks ("shape_predictor_68_face_landmarks.dat") using Haar cascade Classifier for face detection.

Scenario 2: AUI input based on Dlib landmark detection using Dlib HOG (histogram of oriented gradients) for face detection.

Scenario 3: AUI input based on Mediapipe detection of 468 landmarks.

Both Scenario 1 and Scenario 2 involve the detection of 68 landmarks based on Dlib. However, in this context, the study aims to compare the impact or differences of using face detection methods, namely Haar cascade Classifier and Dlib, as the foundation for the 68 landmark detection. This is due to the fact that Dlib's 68 landmark detection relies on face detection. Additionally, each scenario undergoes AUI input testing as depicted in Figure 3.

Upon successful landmark detection, the subsequent phase involves designing the AUI input for detecting open and closed eye conditions. For the eye condition detection, the researchers utilize the eye aspect ratio (EAR) method. This method calculates the height and width of the eye region [34, 39, 40]. As illustrated in Figure 6 and Equation 1, the EAR method is chosen based on previous research demonstrating its success in detecting open and closed eye conditions, particularly in drowsiness applications [34, 41].

$$EAR = \frac{\| p_2 - p_6 \| + \| p_3 - p_5 \|}{2 \| p_1 - p_4 \|} \quad (1)$$

As shown in Figure 6 and Eq. (1), the computation of the detected eye landmark region utilizes the eye aspect ratio (EAR), which calculates the ratio between the height and width of the eye region. The values P1 through P6 represent the locations of the detected eye landmark points. The EAR

tends to remain relatively constant when the eyes are open and approaches a value close to zero, allowing this method to be used for estimating open and closed eye conditions [34, 41]. In this study, the detection of eye conditions serves as input information for AUI in discerning open and closed eye states.

Regarding head movement detection or monitoring, this experiment serves as the AUI input for view direction detection, encompassing detection of views to the right or forward, views to the right or left, and views up or down. Head movement monitoring contributes to the study of human behavior, enhancing our understanding of a person's attention, focus, and emotional state [42]. For head movement detection, the researchers employ the head pose estimation method, correlating the 2D facial landmark detection results with the 3D face shape [43-45], based on the detection of landmark points on the face, as depicted in Figure 7.

In this experiment, 2D landmark points are derived from landmark detection using Dlib and Mediapipe. The 3D face shape is expressed in world coordinates, with the ability to convert world coordinate points to camera coordinates. In camera coordinates, these points are projected onto the image plane using camera intrinsic parameters, including focal length, optical center, and radial distortion, as outlined in Eq. (2). Matrix A within the equation represents the camera matrix containing intrinsic parameters and key points, while matrix [R|t] signifies the joint rotation-translation matrix that transforms points from the 3D world coordinate system to the 3D camera coordinate system [46].

$$p_i = A\,[R|t]P_i \tag{2}$$

The head pose estimation at 2D-3D landmark points is conducted through the solvePnP (Perspective-n-Point) algorithm in the openCV function. SolvePnP is an algorithm that extracts rotation vectors and translation vectors using DLT (direct linear transform) and Levenberg-Marquardt optimization [46, 47]. Through solvePnP 2D-3D, head movement estimation is obtained, encompassing the roll, pitch, and yaw rotation axis values. However, the experiment primarily focuses on the pitch (vertical) rotation value, representing forward, up, and down views, and the yaw (horizontal) value, representing left and right views. This movement aligns with the head movement classification illustrated in Figure 8 [42].

As depicted in Figure 8, head movement classification serves as a reference for head movement detection, representing the direction of the view. In this experiment, it is utilized as the basis for determining head movement detection.

The comprehensive design of the proposed Attentive User Interface (AUI) input, incorporating facial landmark detection, open and closed eye conditions, and head movement detection, is illustrated in the block diagram presented in Figure 9.

As seen in Figure 9, there are 3 sections labeled A, B, and C that consist of parts of the detection process. As previously explained, face landmark detection is a very important initial part in the process of designing this AUI input system because, through the results of face landmark detection, eye open-close detection and head movement detection can be modeled. On the label of section "A", there are three scenarios of face landmark detection processes that have different methods, and later each face landmark detection result in each scenario will be continued in section "B", namely eye open-close detection and section "C" head movement detection. The purpose of making these three scenarios is to test, know, and compare the performance of each detection scenario when the AUI input system is running on Raspberry Pi 4. Each detection process, including face landmark detection, eye open-close detection, and head movement detection, is programmed with the same algorithm the difference lies only in the face landmark detection method based on the three proposed scenarios. Later, the design of this AUI input system resulted in three application-based Python forms based on scenarios 1, 2, and 3.

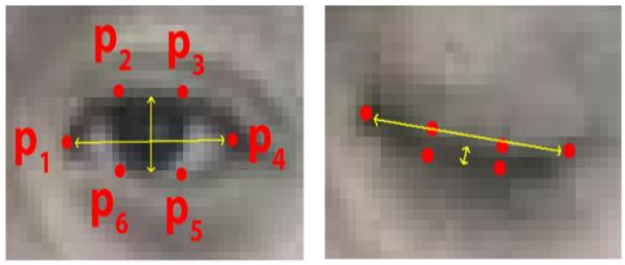


Figure 6. Eye landmark indicators [34]

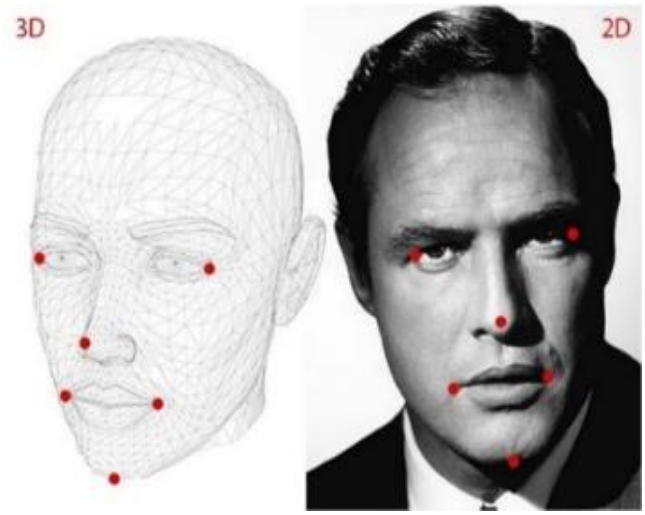


Figure 7. 3D landmarks vs 2D landmarks [44]

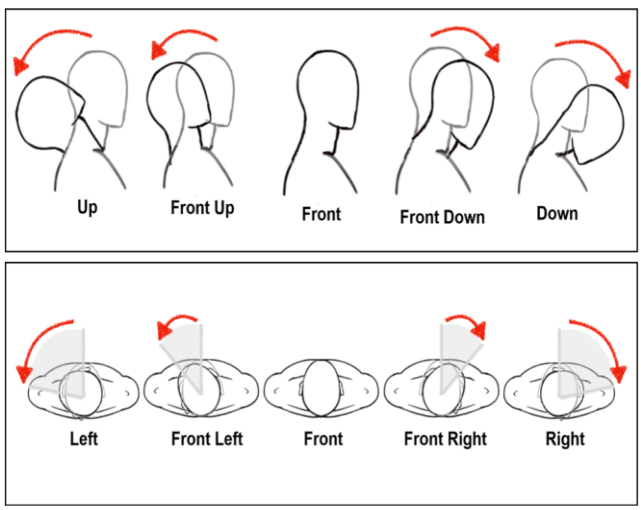


Figure 8. Classification of head movements [42]

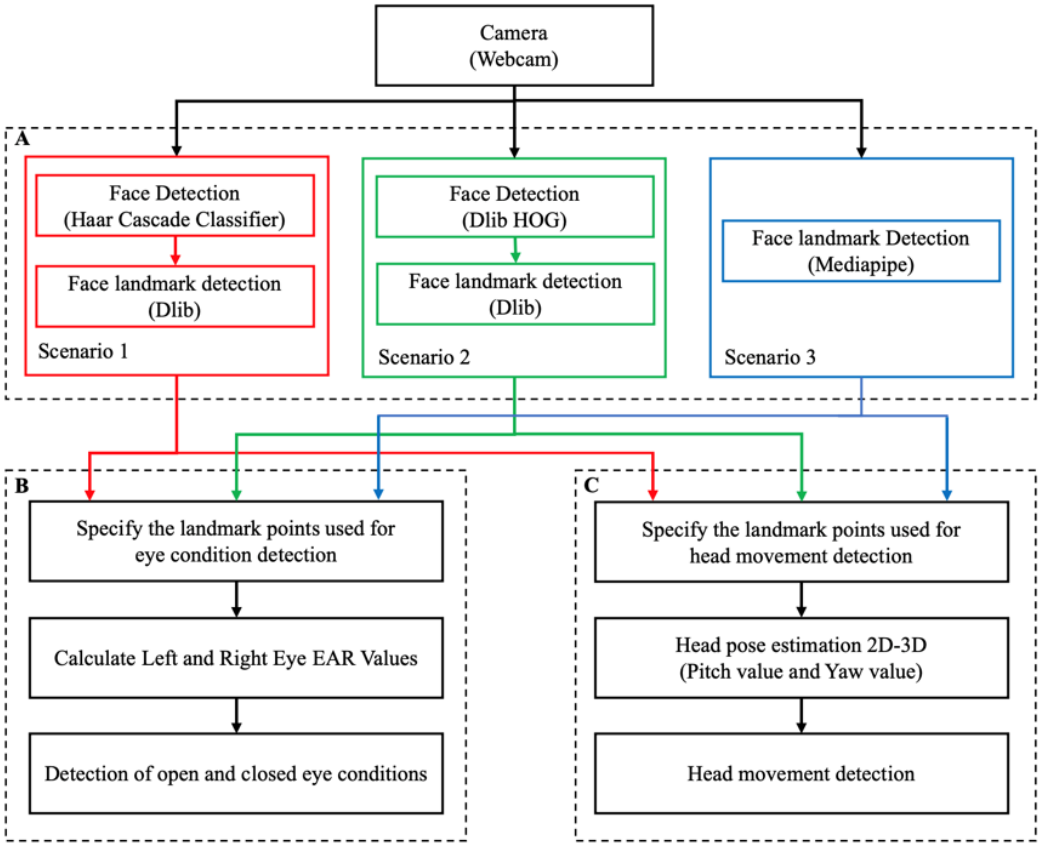


Figure 9. AUI input block diagram: A. Face landmark detection, B. Eye open-close detection, C. Head movement detection

3. PARTICIPANTS AND EXPERIMENT PROCEDURE

In this experiment, 15 participants were voluntarily invited to assess the proposed AUI input system. The group consisted of 13 men and 2 women, aged between 19 and 30 years (M=22.67 years). Additionally, 5 participants wore glasses. It is worth noting that the standard sample size for HCI research is commonly 12 participants, or it aligns with similar previous HCI research studies [48]. Therefore, the inclusion of 15 participants in this experiment was deemed appropriate for testing the proposed AUI input system. Figure 10 presents an overview of the participants involved in testing the AUI input system.

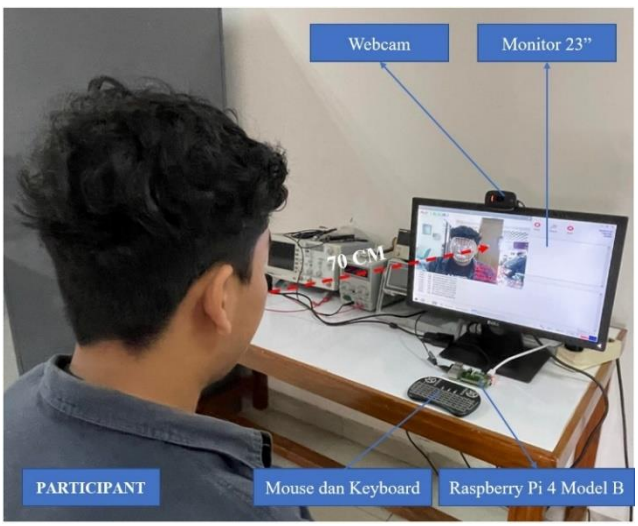


Figure 10. Participants in the AUI input system testing

As depicted in Figure 10, participants will engage in testing AUI input running on the Raspberry Pi 4. Based on the results of the system design, there are three application-based Python input AUIs that will be tested on participants. The three applications have the functions of face landmark detection, eye open-close detection, and head movement detection. It's just that the three applications have differences in facial landmark detection methods based on the three scenarios proposed. As for the steps of this experimental testing procedure, they include:

-In the first step, participants were asked to sit in front of a monitor screen that had a webcam attached to it, with an average distance of 70 cm between the participant and the monitor.

-In the second step, when the participant was ready for testing, the operator would run three application-based Python input AUIs in turn. While running each application, the participant was asked to follow several tasks as listed in Table 2.

-In the third step, each participant performed the test based on the assigned task, observing several indicators as listed in Table 3. In addition, several tools, including a protractor ruler and a protractor application running on a smartphone, were used to measure the detection angle of the AUI input system, which was then recorded and evaluated. The first step to the third step is alternately performed until the 15th participant.

After obtaining the AUI input test data from participants, the next step is evaluation, with the aim of observing the performance of the system when run on a Raspberry Pi 4 and assessing the detection accuracy of the three proposed scenarios. All AUI input detection outcomes were assessed based on the prediction accuracy, as determined by the following Eq. (3):

$$Accuracy = \frac{Number\ of\ correct\ prediction}{Total\ number\ of\ prediction} \times 100\% \quad (3)$$

Subsequently, the test results for each indicator and various test tasks will be compared across the three proposed scenarios. This comparative analysis aims to ascertain the performance and accuracy of each AUI input detection.

Table 2. Participants' tasks during AUI input testing in the three experimental scenarios

Task	Description
Task 1	Involves facial landmark detection. Participants are required to face the front of the monitor screen (front), followed by moving their head up and down (pitch) and left and right (yaw).
Task 2	Focuses on detecting open or closed eye conditions. Participants face the front of the monitor screen (front) and are then instructed to move their head up and down (pitch) and left and right (yaw) for both open and closed eye conditions.
Task 3	Pertains to the detection of head movements influencing the direction of the view. Participants face the front of the monitor screen (front) and are then directed to move their head up and down (pitch) and left and right (yaw).

Table 3. Detection indicator

Indicator	Description
Face landmark detection	Assessment of the system's capability to detect the user's facial landmarks as input information for AUI, enabling the system to detect and recognize the user's face
Eye open-close detection	Evaluation of the system's ability to detect the user's eye condition as AUI input information, facilitating the system's determination of whether the user's eyes are open or closed
Head movement detection	Examination of the system's capability to detect the user's head movement as input information for AUI, allowing the system to ascertain the direction of the user's view, whether it is forward, up, down, left, or right

4. EXPERIMENT RESULT AND DISCUSSION

The presentation of the results for the tested AUI input proposed through the Raspberry Pi 4 is illustrated in Figure 11. It is evident that the three AUI input scenarios are functional, as indicated by the detection codes representing various AUI input indicators, so that scenarios 1, 2, and 3 can display the AUI input detection result indicator with the participant facing front or facing the monitor.

Figure 11 provides insights into the functionality of the system based on the proposed scenarios tested on the participants. The AUI input indicators are displayed in the upper left corner, presenting detection result codes in alignment with the AUI input design depicted in the previous Figure 3. The information in Figure 11 indicates "AUI: a, b, d" with the following meanings:

-Code “a” provides information on detected facial landmarks, enabling the detection of AUI input information related to face detection and recognizing the user's face.

-Code “b” offers information on eye condition detection, with the detection result indicating an open eye. Thus, the AUI input system successfully detects information related to eye condition.

-Code “d” furnishes details on head movement conditions associated with the viewing direction. The result indicates forward view detection, facilitating the AUI input system to detect information on viewing direction.

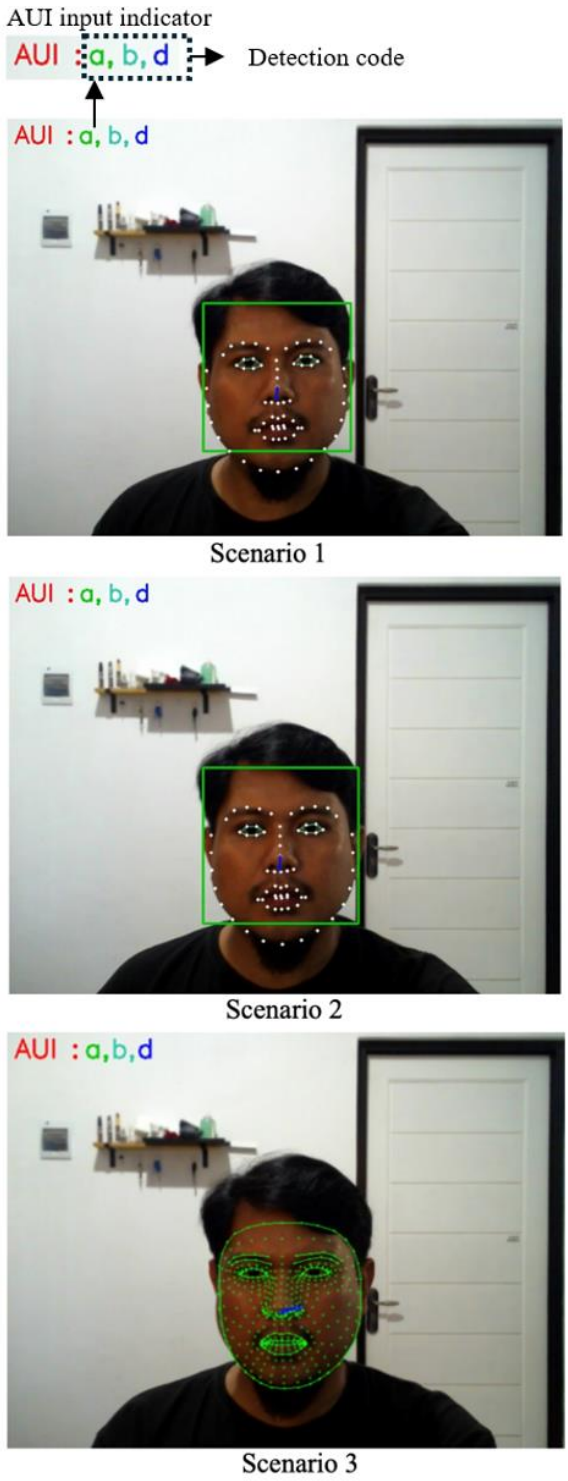


Figure 11. AUI input results using Raspberry Pi 4

The monitoring results are observed through the Raspberry Pi task manager when executing three AUI input system scenarios. The primary focus of the monitoring process is directed towards examining central processing unit (CPU) usage and memory usage. This approach facilitates an assessment of the performance of the proposed AUI input system, as depicted in Table 4.

Table 4. Raspberry Pi 4 task manager results

Testing	Average Value of CPU Usage (%)	Average Value of Memory Usage (MB)
Scenario 1	82.4%	273.7 MB
Scenario 2	38.2%	266.9 MB
Scenario 3	42.6%	233.1 MB

Table 5. FPS comparison on tre AUI input scenarios

FPS Testing	Average FPS	Minimum FPS Value	Maximum FPS Value
Scenario 1	5	5	6
Scenario 2	3	3	4
Scenario 3	10	9	11

Based on Table 4, Scenario 1 of the AUI input system exhibits the highest average CPU and memory usage values when compared to Scenarios 2 and 3. Specifically, Scenario 2 demonstrates the lowest average CPU usage, while Scenario 3 shows the lowest memory usage among the proposed AUI input system scenarios. In addition to task manager monitoring, we also observed the frames per second (FPS) values, as shown in Figure 12 and Table 5.

As shown in Figure 12 and Tables 5, based on the experimental results, scenario 3 has the highest FPS value with an average value of 10 FPS when running on Raspberry Pi 4, while the minimum FPS value is 9 FPS and the maximum FPS value is 11 FPS, so when compared to the FPS values of scenarios 1 and 2, the FPS value of scenario 3 using the Mediapipe-based facial landmark detection method is much better. On the other hand, scenario 2 recorded the lowest FPS value in this test, with an average value of 3 FPS, while the FPS value of scenario 1 is in the middle with an average value of 5 FPS. From the results of this test, it was found that differences in facial landmark detection methods affect the FPS value. Besides that, we found that the FPS value affects the AUI input results in face landmark detection, eye open-close detection, and head movement detection. Higher FPS values correspond to faster AUI input detection results when executed on a Raspberry Pi 4, while lower FPS values result in slower or delayed AUI input detection. In addition to the superiority FPS value, the average CPU and memory usage is also quite low when running on the Raspberry Pi 4, so in terms of FPS values and task manager results, scenario 3 is superior.

Regarding accuracy testing results, this experiment refers to the tasks outlined in Table 2 and detection indicator in Table 3. Each task was tested in three scenarios in turn by the participants, and the pitch and yaw detection angles are measured based on head movements, as illustrated in Figure 8. For pitch detection angles, representation includes front (0°), up (0° to 80°) and down (0° to -80°), while yaw detection angles represent front (0°), right (0° to 90°) and left (0° to -90°). The AUI input test results for facial landmark detection based on Task 1 in the pitch and yaw sections are presented in Tables 6 and 7.

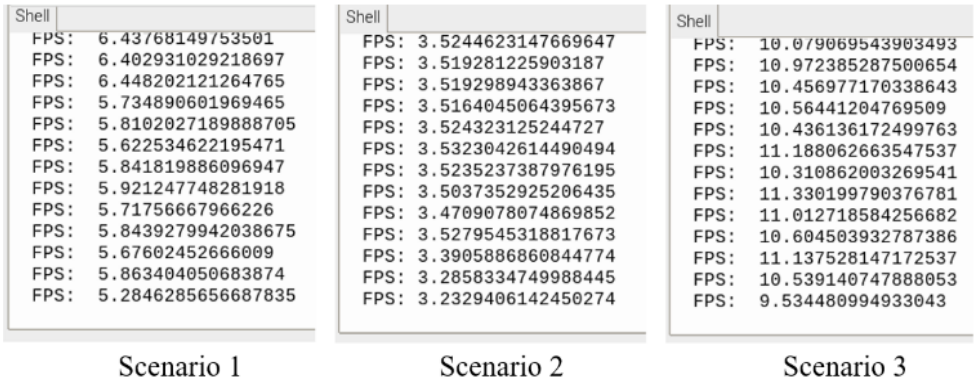


Figure 12. Real-time FPS readings on the Raspberry Pi 4

Table 6. AUI input results on face landmark detection based on task 1 on the pitch side

Testing	Pitch (vertical)						
	Accuracy (%)						
	Angle						
	Down			Front		Up	
	-80°	-60°	-30°	0°	30°	60°	80°
Scenario 1	0%	13%	100%	100%	100%	0%	0%
Scenario 2	0%	100%	100%	100%	100%	100%	0%
Scenario 3	100%	100%	100%	100%	100%	100%	100%

Table 7. AUI input results on face landmark detection based on task 1 in yaw side

Testing	Yaw (horizontal)						
	Accuracy (%)						
	Angle						
	Left			Front		Right	
	-90°	-60°	-30°	0°	30°	60°	90°
Scenario 1	0%	0%	100%	100%	100%	0%	0%
Scenario 2	0%	100%	100%	100%	100%	100%	0%
Scenario 3	100%	100%	100%	100%	100%	100%	100%

As shown in Tables 6 and 7, when 15 participants performed the test with the position facing the monitor screen (front), in turn, scenarios 1, 2, and 3 demonstrated effective detection of facial landmarks, each achieving an accuracy of 100%. This proves that in the front-facing position, all the proposed scenarios can work to display the face landmark detection indicator as AUI input information related to face detection. However, when testing moving the head up and down (pitch), as well as testing moving the head right and left (yaw), there are differences in facial landmark detection results. This test is carried out in stages based on each detection angle accuracy reading result, as listed in the table.

When 15 participants tested moving their heads up and down (pitch) using scenario 1, the accuracy of detecting facial landmarks was only optimal at a reading angle of 0° to 30° for head movement up and 0° to -30° for head movement down, so if the reading of head movement up exceeds an angle of 30^0 and head movement down exceeds an angle of -30°, the accuracy of scenario 1 is not optimal for detection and cannot even detect facial landmarks. While the accuracy results for scenario 2 testing are better when compared to scenario 1, when testing moving the head up and down based on scenario 2, the accuracy of detecting facial landmarks can be more optimal at the reading angle of 0° to 60° for head movement up and 0° to -60° for head movement down. In contrast to testing the detection of facial landmarks using scenario 3, scenario 3 is superior in detection accuracy when compared to scenarios 1 and 2, because when 15 participants tested with scenario 3, the accuracy of detecting facial landmarks could be optimal at reading angles of 0^0 to 80^0 for head movements up and 0^0 to -80^0 for head movements down, so there were no facial landmark detection failures when participants tested with scenario 3.

Similar to the test of moving the head right and left (yaw) as shown in Table 6, the accuracy of detecting facial landmarks using scenario 3 is superior when compared to scenario 1 and scenario 2, the accuracy of detecting facial landmarks can be optimized up to a reading angle of 0° to 90° for right head movements and 0° to -90° for left head movements, and there were no facial landmark detection failures when 15 participants moved their heads left and right when testing using scenario 3. Therefore, based on Task 1 experiments, AUI input using scenario 3 exhibits superior face landmark detection (Mediapipe based) compared to scenarios 1 and 2, as illustrated in Figure 13. This figure provides an example of a sample comparison of scenarios 1, 2, and 3 at a detection angle of -90^0 yaw, indicating that scenarios 1 and 2 fail to detect the AUI input on the side of the -90^0 yaw detection angle, while scenario 3 can still detect the AUI input.

In testing the accuracy of eye condition detection, the test stages are not much different from testing face landmark detection. When participants are asked to move their heads up and down (pitch) and move their heads right and left (yaw), we observe the results of open eye condition detection and closed eye condition detection as described in task 2. The EAR method applied to detect eye conditions was tested in scenarios 1, 2, and 3. The test results show that the accuracy of open eye condition detection in scenario 3 is superior when compared to scenarios 1 and 2, when participants move their heads up and down and move their heads right and left, as shown in Tables 8 and 9.

As for the results of testing the detection of closed eye conditions, scenarios 1, 2, and 3 have the same accuracy value when the participant moves the head up and down and moves the head right and left, as seen in Tables 10 and 11, so that when the head movement up exceeds the angle of 30^0 and the head movement down exceeds the angle -30^0, as well as the right head movement exceeding the angle 30^0 and the left head movement exceeding the angle -30^0, the system cannot detect closed eye conditions.

The use of the EAR method and participant testing enable the classification of open and closed eye conditions. By setting the EAR threshold value to 0.25, a detection result indicating a closed eye condition is established if the EAR value is ≤ 0.25, whereas an open eye condition is indicated if the EAR value is ≥ 0.25, as shown in Figure 14. Additionally, despite the presence of five participants wearing glasses, experimental results indicate that this factor does not impact the accuracy of eye condition detection. The system is thus proposed to reliably detect both open and closed eye conditions as AUI input, even when participants are wearing glasses.

Table 8. AUI input results on open eye condition detection based on task 2 on the pitch side

Testing	Pitch (vertical)						
	Accuracy (%)						
	Angle						
	Down			Front		Up	
	-90°	-60°	-30°	0°	30°	60°	90°
Scenario 1	0%	0%	100%	100%	100%	0%	0%
Scenario 2	0%	0%	100%	100%	100%	0%	0%
Scenario 3	100%	100%	100%	100%	100%	100%	100%

Table 9. AUI input results on open eye condition detection based on task 2 in yaw side

Testing	Yaw (horizontal)						
	Accuracy (%)						
	Angle						
	Left			Front		Right	
	-90°	-60°	-30°	0°	30°	60°	90°
Scenario 1	0%	0%	100%	100%	100%	0%	0%
Scenario 2	0%	100%	100%	100%	100%	100%	0%
Scenario 3	100%	100%	100%	100%	100%	100%	100%

Table 10. AUI input results on closed eye condition detection based on task 2 on the pitch side

Testing	Pitch (vertical)						
	Accuracy (%)						
	Angle						
	Down			Front		Up	
	-90°	-60°	-30°	0°	30°	60°	90°
Scenario 1	0%	0%	100%	100%	100%	0%	0%
Scenario 2	0%	0%	100%	100%	100%	0%	0%
Scenario 3	0%	0%	100%	100%	100%	0%	0%

Table 11. AUI input results on closed eye condition detection based on task 2 in yaw side

Testing	Yaw (horizontal)						
	Accuracy (%)						
	Angle						
	Left			Front		Right	
	-90°	-60°	-30°	0°	30°	60°	90°
Scenario 1	0%	0%	100%	100%	100%	0%	0%
Scenario 2	0%	0%	100%	100%	100%	0%	0%
Scenario 3	0%	0%	100%	100%	100%	0%	0%

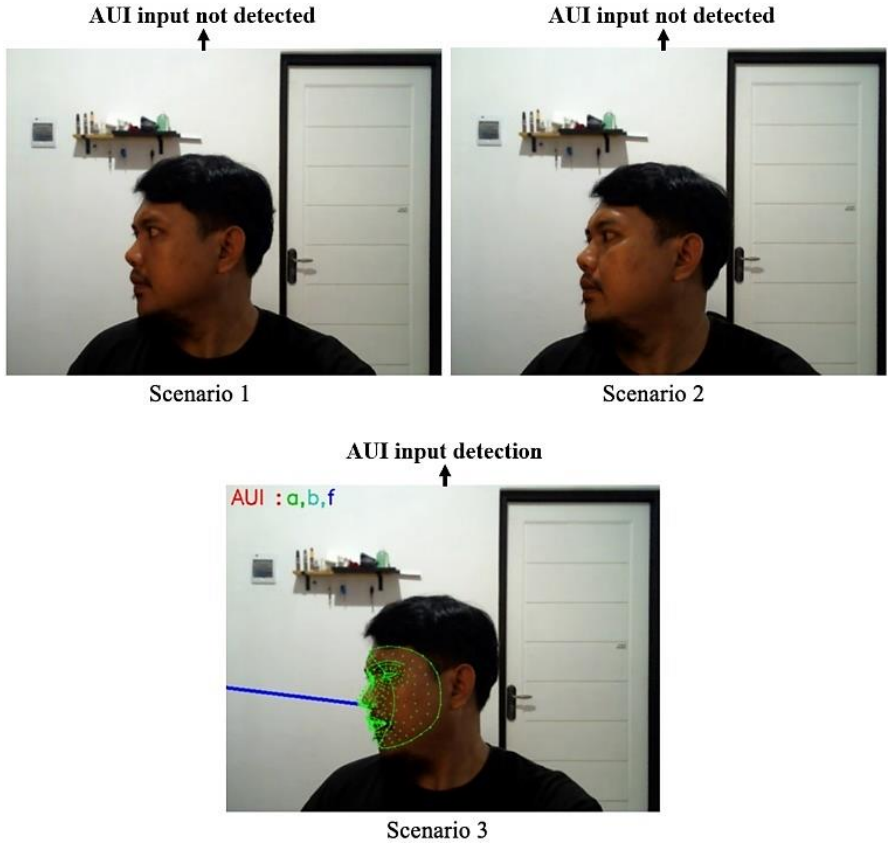


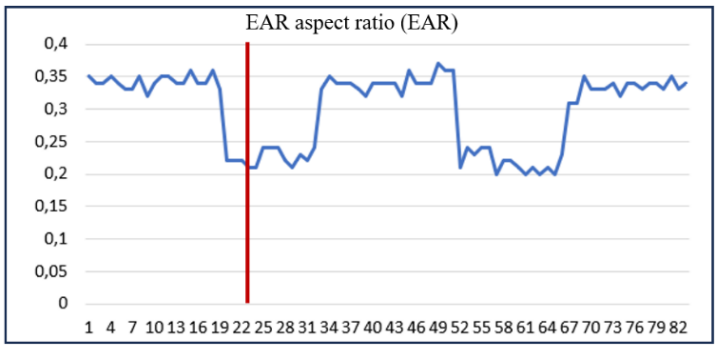
Figure 13. An example of scenario 1, 2 and 3 comparison at -90^0 yaw detection angle

Concerning AUI input testing for head movement detection, representing the direction of view encompassing forward, upward, and downward views (pitch), as well as left and right views (yaw), it is observed that scenario 3 outperforms scenarios 1 and 2. This is evident in Tables 12 and 13, which present the results of participant testing for Task 3, aligning with the results obtained from Task 1 testing on face landmark detection testing on face landmark detection testing.

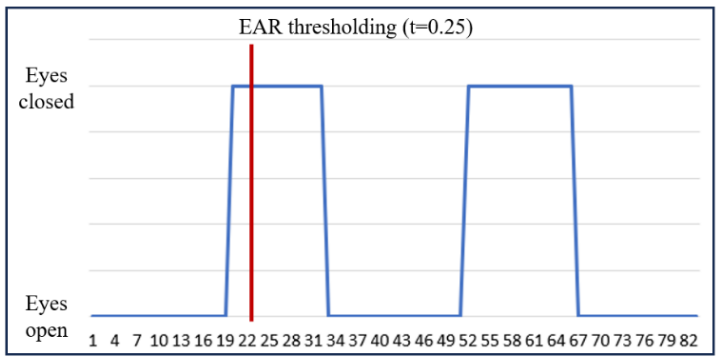
The use of the 2D-3D head pose estimation method and analyzing pitch and yaw rotation axis values, the researchers successfully classified head movements as AUI input for view direction detection, especially by using scenario 3. The researchers defined front view direction detection when both the pitch and yaw rotation axis values were ≤ 15 and ≥ -15. For upward viewing direction detection, they considered pitch axis values ≥ 15, while downward viewing direction detection was defined for pitch values ≤ -15. Similarly, leftward viewing direction detection was designated for yaw values ≥ 15, and rightward viewing direction detection for yaw values ≤ -15. Figure 15 illustrates one of the sample head movement detection results, representing the detection of view direction as an AUI input.

From the experiment results, all three proposed scenarios proved capable of detecting viewing direction as AUI input. Although the scenarios varied in accuracy, scenario 3 demonstrated superior performance in reading head

movements as the direction of view in pitch and yaw angle readings, so that Mediapipe-based facial landmark detection with scenario 3 has the advantage in every test that has been done.



(a) EAR reading sample



(b) EAR thresholding

Figure 14. One of the sample readout graphs of open and closed eye condition detection via the EAR method

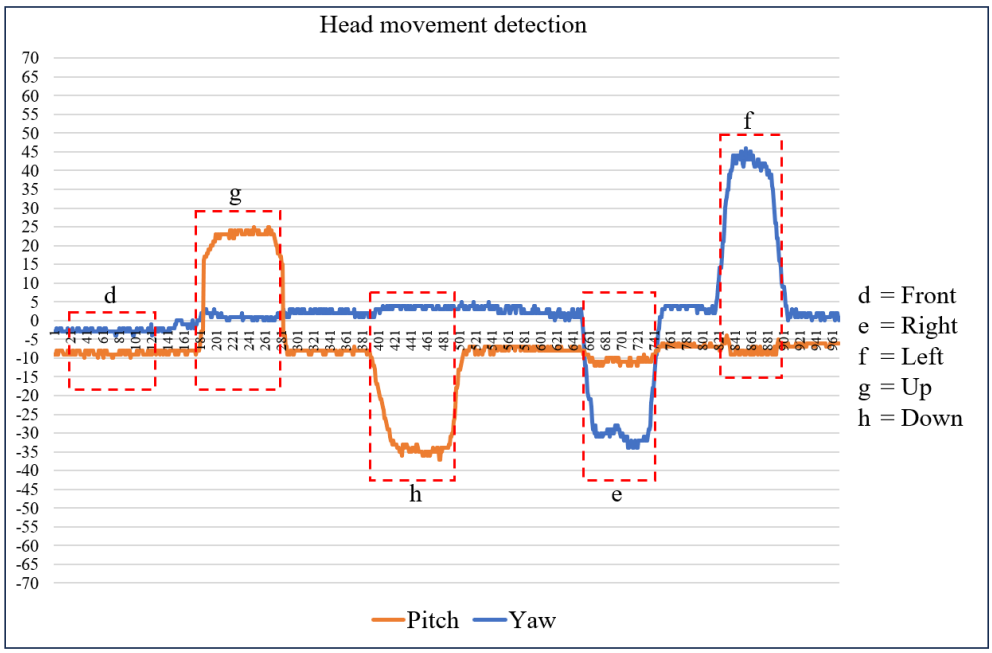


Figure 15. One of the sample head movement detection results representing view direction detection as AUI input

Table 12. AUI input results on head movement detection representing the view direction based on task 3 on the pitch side

Testing	Pitch (vertical) Accuracy (%) Angle						
	-80°	**-60°**	**-30°**	**0° (Front)**	**30°**	**60°**	**80°**
Scenario 1	0%	0%	100%	100%	100%	0%	0%
Scenario 2	0%	100%	100%	100%	100%	100%	0%
Scenario 3	100%	100%	100%	100%	100%	100%	100%

Table 13. AUI input results on head movement detection representing the view direction based on task 3 on the yaw side

Testing	Yaw (horizontal)						
	Accuracy (%)						
	Angle						
	-90°	-60°	-30°	0° (Front)	30°	60°	90°
Scenario 1	0%	0%	100%	100%	100%	0%	0%
Scenario 2	0%	100%	100%	100%	100%	100%	0%
Scenario 3	100%	100%	100%	100%	100%	100%	100%

The exploration of AUI input through the Raspberry Pi 4 covering facial landmark detection, eye opening and closing detection, and head movement detection yielded 8 functional AUI inputs aligned with indicators and predefined tasks. The performance of the Raspberry Pi 4 embedded system exhibited variations based on the employed methods and scenarios. Notably, the test results highlighted that the AUI input system in scenario 3 (Mediapipe-based) outperformed scenarios 1 and 2 in terms of both performance and accuracy. Therefore, the exploration of AUI input through Raspberry Pi 4 with scenario 3 can serve as a reference for designing AUI in diverse forms of interaction, particularly to support the visual attention system design in computer vision-based systems.

Based on the conducted experiments, although the Raspberry Pi 4 embedded system can support input AUI computer vision-based systems, several experiments are needed to obtain a method that can work optimally on the Raspberry Pi, as has been done in this study, so as to obtain the best performance of the system. In addition, the results of this study prove that the AUI system using Raspberry Pi has the advantage of designing a visual attention-based AUI system that can support various systems with small media, limited voltage sources, and low cost. This is crucial for utilizing it as a parameter for user interface design in various forms of interaction, including HCI, HRI, and various other interaction systems. Because, in 2024, the price of the Raspberry Pi 4 from $35 [49], it can be a consideration besides using desktop computers. However, it cannot be denied that desktop computers have a larger CPU and memory when compared to the Raspberry Pi, especially desktop computers equipped with GPU.

5. CONCLUSION

This research successfully explored AUI input through the Raspberry Pi 4 Model B, based on face landmark detection as AUI input information for face detection, eye condition detection as AUI input information for open or closed eye condition detection, and head movement detection as AUI input information for forward, up, down, right, and left gaze direction detection, so as to produce 8 computer vision-based AUI inputs.

In this experiment, we modeled three AUI input scenarios based on several methods with the aim of obtaining the most optimal AUI input system running on a Raspberry Pi. Based on the test results, the three proposed scenarios can work to detect AUI input according to the indicators and tasks that have been determined, but there are differences in performance and accuracy values in each scenario, where the test results involving 15 participants show that the AUI input system in scenario 3 (Mediapipe-based) is superior to scenarios 1 and 2, This shows that the development of the AUI input system through Raspberry Pi 4 using scenario 3 is able to produce the best performance with an average FPS value of 10 FPS with an average CPU usage of 42.6% and an average memory usage of 233.1 MB, with a detection accuracy value of 100% for facial landmark detection, detection of open eye conditions, detection of forward, up, down, right, and left gaze directions at various detection reading angles.

The use of Raspberry Pi as a computer vision-based system has proven to be capable and reliable for the development of vision-based AUI. The findings of this research can be a reference for the development of AUI, especially in the visual attention system, which allows the system to monitor and know the behavior or condition of the user's attention when performing interaction tasks based on 8 AUI inputs that have been proposed, so we believe this research can be utilized to support the user's attention capacity in various interaction tasks, such as HCI, HRI, and various other interaction tasks, especially based on embedded systems. In addition, the use of Raspberry Pi with open source software, low cost, low power consumption, portability, and small size can be an alternative to the use of desktop computers.

During testing, the limitations of this research are only influenced by light intensity, a lack of light can affect the detection results using a webcam. Because the use of a webcam camera can work well when the light needed is sufficient, we had time to measure the light intensity, where the system can work well in light conditions above 20 Lux, if the light is below the value of 20 Lux, the system cannot work optimally in detecting.

Future research on the development of AUI inputs using the Raspberry Pi 4 embedded system can be expanded based on various other inputs that support vision-based interaction tasks, such as facial expression detection, hand gesture detection, and human pose detection.

REFERENCE

[1] Vertegaal, R. (2003). Attentive user interfaces. Communications of the ACM, 46(3): 30-33. https://doi.org/10.1145/636772.636794

[2] Vertegaal, R., Shell, J.S., Chen, D., Mamuji, A. (2006). Designing for augmented attention: Towards a framework for attentive user interfaces. Computers in Human Behavior, 22(4): 771-789. https://doi.org/10.1016/j.chb.2005.12.012

[3] Vertegaal, R., Shell, J.S. (2008). Attentive user interfaces: The surveillance and sousveillance of gaze-aware objects. Social Science Information, 47(3): 275-298. https://doi.org/10.1177/0539018408092574

[4] Singh, G., Bermúdez i Badia, S., Ventura, R., Silva, J.L. (2018). Physiologically attentive user interface for robot teleoperation: Real time emotional state estimation and interface modification using physiology, facial expressions and eye movements. In 11th International Joint Conference on Biomedical Engineering Systems and Technologies, pp. 294-302.

https://doi.org/10.5220/0006733002940302
[5] Maglio, P.P., Matlock, T., Campbell, C.S., Zhai, S., Smith, B.A. (2000). Gaze and speech in attentive user interfaces. In Advances in Multimodal Interfaces—ICMI 2000: Third International Conference Beijing, China, pp. 1-7. https://doi.org/10.1007/3-540-40063-X_1
[6] Bulling, A. (2016). Pervasive attentive user interfaces. Computer, 49(1): 94-98. https://doi.org/10.1109/MC.2016.32
[7] Xu, P., Sugano, Y., Bulling, A. (2016, May). Spatio-temporal modeling and prediction of visual attention in graphical user interfaces. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, San Jose California USA, pp. 3299-3310. https://doi.org/10.1145/2858036.2858479
[8] Castellanos-Cruz, J.L., Gómez-Medina, M.F., Tavakoli, M., Pilarski, P.M., Adams, K. (2019). Comparison of attentive and explicit eye gaze interfaces for controlling haptic guidance of a robotic controller. Journal of Medical Robotics Research, 4(3-4): 1950005. https://doi.org/10.1142/S2424905X19500053
[9] Tavares, A., Silva, J.L., Ventura, R. (2023). Physiologically attentive user interface for improved robot teleoperation. In Proceedings of the 28th International Conference on Intelligent User Interfaces, Sydney, Australia, pp. 776-789. https://doi.org/10.1145/3581641.3584084
[10] Zhang, X., Sugano, Y., Bulling, A. (2019). Evaluation of appearance-based methods and implications for gaze-based applications. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, Glasgow, Scotland, UK, pp. 1-13. https://doi.org/10.1145/3290605.3300646
[11] Tonsen, M., Steil, J., Sugano, Y., Bulling, A. (2017). Invisibleeye: Mobile eye tracking using multiple low-resolution cameras and learning-based gaze estimation. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 1(3): 106. https://doi.org/10.1145/3130971
[12] Hummel, D., Toreini, P., Maedche, A. (2018). Improving digital nudging using attentive user interfaces: Theory development and experiment design. In 13th International Conference on Design Science Research in Information Systems and Technology, Chennai, India.
[13] Cruz, J.L.C. (2019). Development of an attentive user interface for controlling a telerobotic haptic system to support play in children with physical disabilities. Doctoral dissertation, University of Alberta. https://doi.org/10.7939/r3-qz5z-q284
[14] Castellanos-Cruz, J.L., Gómez-Medina, M.F., Tavakoli, M., Pilarski, P., Adams, K.D. (2022). Preliminary testing of eye gaze interfaces for controlling a haptic system intended to support play in children with physical impairments: Attentive versus explicit interfaces. Journal of Rehabilitation and Assistive Technologies Engineering, 9: 20556683221079694. https://doi.org/10.1177/20556683221079694
[15] Shen, H., Asiry, O., Babar, M.A., Bednarz, T. (2023). Evaluating the efficacy of using a novel gaze-based attentive user interface to extend ADHD children's attention span. International Journal of Human-Computer Studies, 169: 102927. https://doi.org/10.1016/j.ijhcs.2022.102927
[16] Mo, R., Shaout, A. (2016). Portable facial recognition jukebox using fisherfaces (FRJ). International Journal of Advanced Computer Science and Applications, 7(3): 9-14. https://doi.org/10.14569/IJACSA.2016.070302
[17] Aliff, M., Hanisah, N.F., Ashroff, M. S., Hassan, S., Nurr, S.F., Sani, N.S. (2022). Development of underwater pipe crack detection system for low-cost underwater vehicle using Raspberry Pi and canny edge detection method. International Journal of Advanced Computer Science and Applications, 13(11): 456-464. https://doi.org/10.14569/IJACSA.2022.0131152
[18] Jolles, J.W. (2021). Broad-scale applications of the Raspberry Pi: A review and guide for biologists. Methods in Ecology and Evolution, 12(9): 1562-1579. https://doi.org/10.1111/2041-210X.13652
[19] Molloy, D. (2016). Exploring Raspberry Pi: Interfacing to the Real World with Embedded Linux. John Wiley & Sons.
[20] Abdulhamid, M., Odondi, O., Muaayed, A.R. (2020). Computer vision based on Raspberry Pi system. Applied Computer Science, 16(4): 85-102. https://doi.org/10.23743/acs-2020-31
[21] Seelam, V., kumar Penugonda, A., Kalyan, B.P., Priya, M.B., Prakash, M.D. (2021). Smart attendance using deep learning and computer vision. Materials Today: Proceedings, 46: 4091-4094. https://doi.org/10.1016/j.matpr.2021.02.625
[22] Lashkov, I., Kashevnik, A., Shilov, N., Parfenov, V., Shabaev, A. (2019). Driver dangerous state detection based on OpenCV & Dlib libraries using mobile video processing. In 2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), New York, NY, USA, pp. 74-79. https://doi.org/10.1109/CSE/EUC.2019.00024
[23] Kumar, A., Chaudhary, S., Sangal, S., Dhama, R. (2022). Face detection and recognition using OpenCV. International Journal of Computer Applications, 184(11): 23-32. https://doi.org/10.5120/ijca2022922085
[24] Shen, K., Ramli, R., Chuah, J.H., Chai, G.M.T. (2023). Driver fatigue detection using opencv and Dlib library. In 2023 IEEE Asia-Pacific Conference on Computer Science and Data Engineering, Nadi, Fiji, pp. 1-6. https://doi.org/10.1109/CSDE59766.2023.10487775
[25] Savin, A.V., Sablina, V.A., Nikiforov, M.B. (2021). Comparison of facial landmark detection methods for micro-expressions analysis. In 2021 10th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, pp. 1-4. https://doi.org/10.1109/MECO52532.2021.9460191
[26] Suherman, S., Suhendra, A., Ernastuti, E. (2023). Method development through landmark point extraction for gesture classification with computer vision and MediaPipe. TEM Journal, 12(3): 1677-1686. https://doi.org/10.18421/TEM123-49
[27] Boyko, N., Basystiuk, O., Shakhovska, N. (2018). Performance evaluation and comparison of software for face recognition, based on Dlib and opencv library. In 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine, pp. 478-482. https://doi.org/10.1109/DSMP.2018.8478556
[28] Reza, M.A.N., Hamidi, E.A.Z., Ismail, N., Effendi, M. R., Mulyana, E., Shalannanda, W. (2021). Design a landmark facial-based drowsiness detection using Dlib

and Opencv for four-wheeled vehicle drivers. In 2021 15th International Conference on Telecommunication Systems, Services, and Applications, Bali, Indonesia, pp. 1-5. https://doi.org/10.1109/TSSA52866.2021.9768278
[29] Kesharwani, H., Mallick, T., Gupta, A., Raj, G. (2022). Automated attendance system using computer vision. In 2022 Second International Conference on Computer Science, Engineering and Applications, Gunupur, India, pp. 1-5. https://doi.org/10.1109/ICCSEA54677.2022.9936266
[30] Heakl, A., Youssef, F., Parque, V., Gomaa, W. (2024). DroneVis: Versatile computer vision library for drones. arXiv preprint arXiv:2406.00447. https://doi.org/10.48550/arXiv.2406.00447
[31] Li, L., Huang, H., Zeng, S., Cao, H., Zheng, R., Lin, S. (2022). AI body detection and teaching system based on Mediapipe machine learning platform and OpenCV computer vision library. Learning & Education, 10(8): 183-184. https://doi.org/10.18282/l-e.v10i8.3115
[32] Choi, C. H., Kim, J., Hyun, J., Kim, Y., Moon, B. (2022). Face detection using haar cascade classifiers based on vertical component calibration. Human-centric Computing and Information Sciences, 12: 11. https://doi.org/10.22967/HCIS.2022.12.011
[33] Jadhav, A., Lone, S., Matey, S., Madamwar, T., Jakhete, S. (2021). Survey on face detection algorithms. International Journal of Innovative Science and Research Technology, 6(2): 291-297.
[34] Soukupova T., Cech, J. (2016). Real-time eye blink detection using facial landmarks. In 21st Computer Vision Winter Workshop, Rimske Toplice, Slovenia. https://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf.
[35] Wassef, E., Abd El Munim, H.E., Hammad, S., Ghoneima, M. (2021). Robust Real-time head pose estimation for 10 Watt SBC. International Journal of Advanced Computer Science and Applications, 12(7): 578-585. https://doi.org/10.14569/IJACSA.2021.0120766
[36] Golob, O. (2022). Analysis of face detection, face landmarking, and face recognition performance with masked face images. arXiv preprint arXiv:2207.06478. https://doi.org/10.48550/arXiv.2207.06478
[37] Lugaresi, C., Tang, J., Nash, H., et al. (2019). Mediapipe: A framework for perceiving and processing reality. In Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR).
[38] Thaman, B., Cao, T., Caporusso, N. (2022). Face mask detection using Mediapipe facemesh. In 2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO), Opatija, Croatia, pp. 378-382. https://doi.org/10.23919/MIPRO55190.2022.9803531
[39] Băiașu, A.M., Dumitrescu, C. (2021). Contributions to driver fatigue detection based on eye-tracking. International Journal of Circuits, Systems and Signal Processing, 15: 1. https://doi.org/10.46300/9106.2021.15.1
[40] Ngasri, M.A., Isa, I.S., Sulaiman, S.N., Soh, Z.H.C. (2019). Automated stand-alone video-based microsleep detection system by using EAR technique. In 2019 9th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Penang, Malaysia, pp. 78-83. https://doi.org/10.1109/ICCSCE47578.2019.9068566
[41] Maior, C.B.S., das Chagas Moura, M.J., Santana, J.M. M., Lins, I.D. (2020). Real-time classification for autonomous drowsiness detection using eye aspect ratio. Expert Systems with Applications, 158: 113505. https://doi.org/10.1016/j.eswa.2020.113505
[42] Jiang, Y., Sadeqi, A., Miller, E.L., Sonkusale, S. (2021). Head motion classification using thread-based sensor and machine learning algorithm. Scientific Reports, 11(1): 2646. https://doi.org/10.1038/s41598-021-81284-7
[43] Mallick, S. (2016). Head pose estimation using OpenCV and Dlib. https://learnopencv.com/head-pose-estimation-using-opencv-and-Dlib/.
[44] Bailey, W. (2020). Pose estimation using OpenCV facial detection and tracking, CPSC 6820 Sprin. http://andrewd.ces.clemson.edu/courses/cpsc482/spring20/projects/reports/jwb4.pdf.
[45] Hosamani, A., Phirke, M. (2020). Real-time head pose estimation based on face geometry. In Proceedings of the 5th International Conference on Multimedia and Image Processing, Nanjing, China, pp. 38-42. https://doi.org/10.1145/3381271.3381296
[46] Reinoso, L.L.R., López, F.L.G., Gutiérrez, J.C., Bressan, G., Ruggiero, W.V. (2020). Real-time head pose estimation with SVM model for frontal face classification. In IADIS International Conference Sustainability, Technology and Education, pp. 63-67. https://doi.org/10.33965/its_ste2020_202001C008
[47] Jeong, M., Kim, D., Park, S., Paik, J. (2022). Drowsy status monitoring system based on face feature analysis. In 2022 International Conference on Electronics, Information, and Communication (ICEIC), Jeju, Korea, Republic of, pp. 1-4. https://doi.org/10.1109/ICEIC54506.2022.9748269
[48] MacKenzie, I.S. (2012). Human-Computer Interaction: An Empirical Research Perspective. Morgan Kaufmann Publishers Inc.
[49] Buy Raspberry Pi 4 Model B Raspberry Pi. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/.

NOMENCLATURE

AUI	Attentive User Interface
HCI	Human-Computer Interaction
GPU	Graphics Processing Units
HRI	Human-Robot Interaction
SBC	Single-Board Computer
EAR	Eye Aspect Ratio
DLT	Direct Linear Transform
FPS	Frames Per Second
CPU	Central Processing Unit