DOI:xxxxxxxxxxx

# APPLICATION OF ARTIFICIAL INTELLIGENCE AND COMPUTER VISION TO SOFTWARE INTERFACE CONTROLS

**Danijela Milekić[1*], Lidija Krstanović[2], Bojan Banjac[3]**

*[1]Computer Graphics Chair, Faculty of Technical Sciences, University of Novi Sad, Novi Sad 21000, Serbia; danijelamilekic@uns.ac.rs*
*[2]Computer Graphics Chair, Faculty of Technical Sciences, University of Novi Sad, Novi Sad 21000, Serbia; lidijakrstanovic@uns.ac.rs*
*[3]Computer Graphics Chair, Faculty of Technical Sciences, University of Novi Sad, Novi Sad 21000, Serbia; Intens doo, Bulevar Evrope 28, Novi Sad 2100, Serbia; bojan.banjac@uns.ac.rs*

(*Corresponding author)

**Abstract**

*This study demonstrates how computer vision can manage the game and the user interface of the Unreal Engine program, more specifically the computer's input devices. For these purposes, the Python programming language and the MediaPipe and PyAutoGUI libraries were utilized. Real-time implementation of computer vision-based controls is achieved through communication between Unreal Engine and Python code.*

*The purpose of this study is to use computer vision methods for controlling interactive software's user interface, offering users enhanced game control. The main improvement and result of this study was adapting the system for simultaneous detection and communication of both hand and facial positions, as the initial setup did not support dual tracking effectively. Further research is required to address the complexities of integrating pre-existing commands, which will be the focus of future studies.*

*Keywords: computer vision, GUI (graphical user interface), Python programming language, MediaPipe, Unreal Engine*

## 1 INTRODUCTION

In modern interactive systems, computer vision provides significant opportunities for natural and intuitive control of user interfaces [1]. This paper presents the integration of computer vision and machine learning techniques to manage both the game and user interface within Unreal Engine, using Python, MediaPipe, and PyAutoGUI libraries. The main goal of this study is to develop a system that enables real-time game control through hand gestures and eye blinks [2]. A two-way communication link has been established between the Python script and Unreal Engine, where hand movements control the character's navigation, and eye blinks trigger additional actions such as jumping or switching movement modes. The contribution of this work is the introduction of movement mode switching as a novel feature, along with enhanced simultaneous interaction capabilities. Unlike previous systems where actions like jumping could only be performed from a stationary position, this system allows the character to move while executing them, offering a more dynamic and fluid interaction model.

## 2 METHODOLOGY

This section outlines the methods and technologies used to develop and implement the system for real-time game control through computer vision. It is divided into two key areas: system architecture and communication and application of computer vision for gesture and blink detection.

## 2.1 System Architecture and Communication

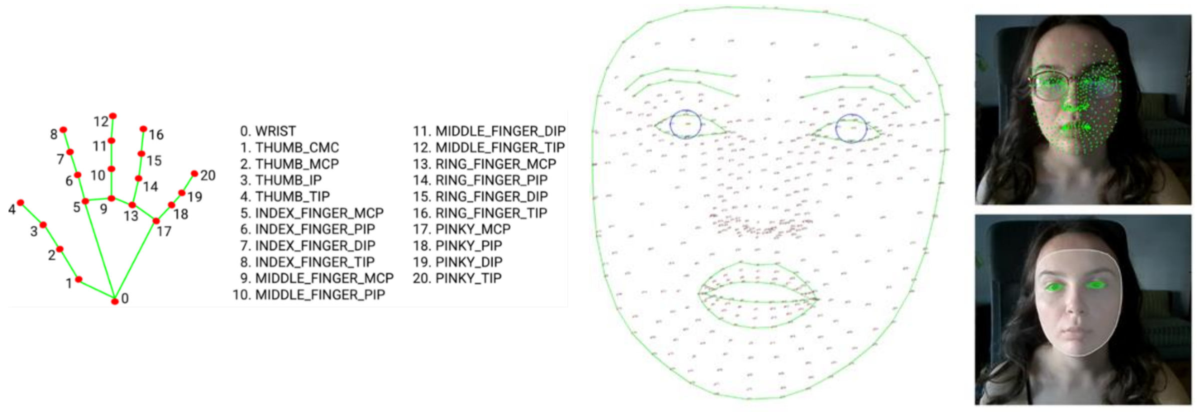
The system architecture is divided into three key stages: detecting hand movements and eye blinks, translating them into controls and implementing these controls within the Unreal Engine environment. The first two stages are handled through Python, while the third phase is implemented in Unreal Engine. In the first stage, the video stream is analyzed to detect the user’s hand movements and eye blinks, differentiating between the left and right hands as well as the left and right eye blinks. The right hand controls input devices, while gestures of the left hand and eye blinks are sent to Unreal Engine as integer values to control character movement. Unlike previous implementations, hand and face detection are processed concurrently, enabling more complex interactions [3].

For input control, specific gestures of the right hand are used. For example, the mouse cursor moves when the index finger is extended, while a left mouse click is activated when the index finger is bent. Additionally, the system adjusts the volume based on the distance between the tips of the thumb and index finger, and a closed fist simulates pressing the "C" key on the keyboard. The gestures of the left hand control the in-game character, such as moving forward, backward, left, or right. Unlike previous implementations, hand and face detection are processed concurrently, enabling more complex interactions. A blink of the right eye triggers the character to jump, while a blink of the left eye activates a flying mode that lasts for ten seconds.

Based on integer values from hand gestures or eye blinks, the corresponding text message is generated to control the character's actions. The motion data is then packed into a Python dictionary and transmitted to Unreal Engine.

## 2.2 Application of Computer Vision

Motion detection is achieved with MediaPipe models. The MediaPipe Hands model tracks up to 21 3D landmarks on a hand in a single frame [4], relying on classical pose-estimation and reconstruction methods [5]. Fig.1a illustrates that two models were used: one detects the palm and creates a bounding box, while the other processes the cropped region for accurate keypoints. Fig.1b shows the MediaPipe Face Mesh with 468 landmarks. The system identifies the face's surface using a single camera and machine learning, without depth information. The pipeline includes models for face detection and landmark localization [6, 7].



(a) (b)

*Fig.1. (a) Features of the hand [3]; (b) Facial features [5]*

For detecting different hand positions, information such as the distance between key landmarks and the angle formed by three finger landmarks was used. The angle is calculated by determining the difference between two angles, each formed by lines connecting the origin and the respective landmarks. These angles indicate how the fingers bend or extend. The distance between specific landmarks, such as the tips of the thumb and index finger, is calculated using the Euclidean distance formula, allowing the system to track finger movements and determine the state of the hand.

Fig.2a shows the method for testing the distance threshold value, with the thumb moved and the distance displayed on the frame, while Fig.2b illustrates the angles, landmarks, and distance used to detect a left mouse click, checking if the thumb and index finger are close enough and if the joint angle meets the criteria.

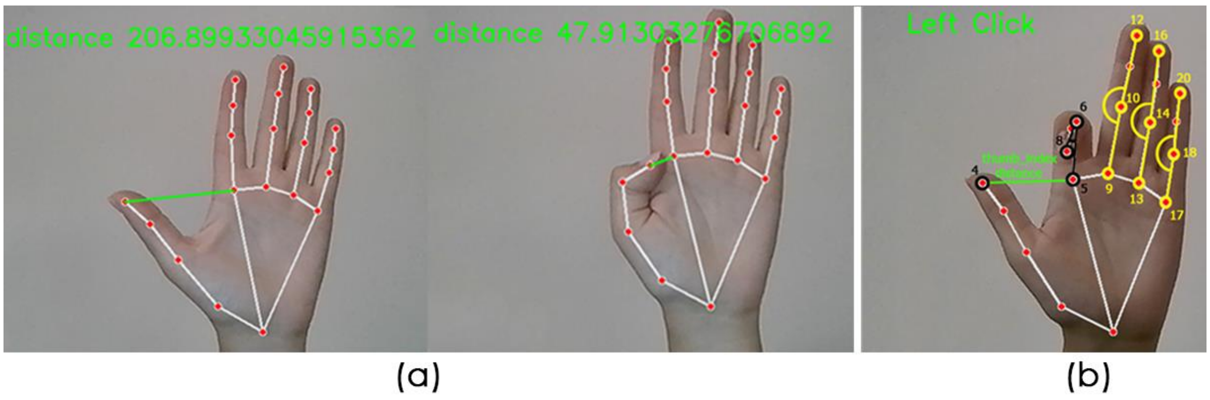


*Fig.2. (a) Testing the distance threshold value; (b) Left mouse click - conditions*

After detecting facial landmarks, the system calculates the distance between key eye points to determine both the horizontal and vertical distances (Fig.3a). These values are then used to compute a ratio that distinguishes between open and closed eyes. A threshold of 5.5 was determined to identify blinks for one or both eyes (Fig.3b). When the ratio exceeds this threshold, actions like jumping or flying are triggered, and a numerical value is returned for further processing.

Due to time and resource constraints, the focus of the project was on developing and validating the system's functionality, rather than conducting an extensive quantitative analysis. The primary goal was to ensure real-time interaction and verify that the system responds accurately to user input, which is the main scope of this study.

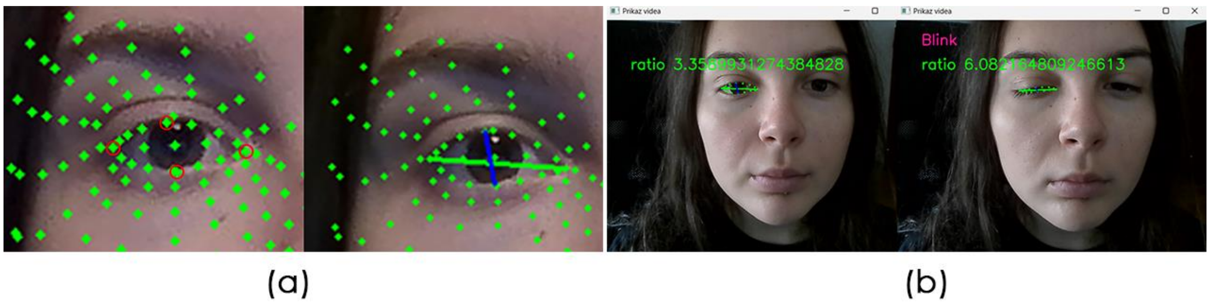
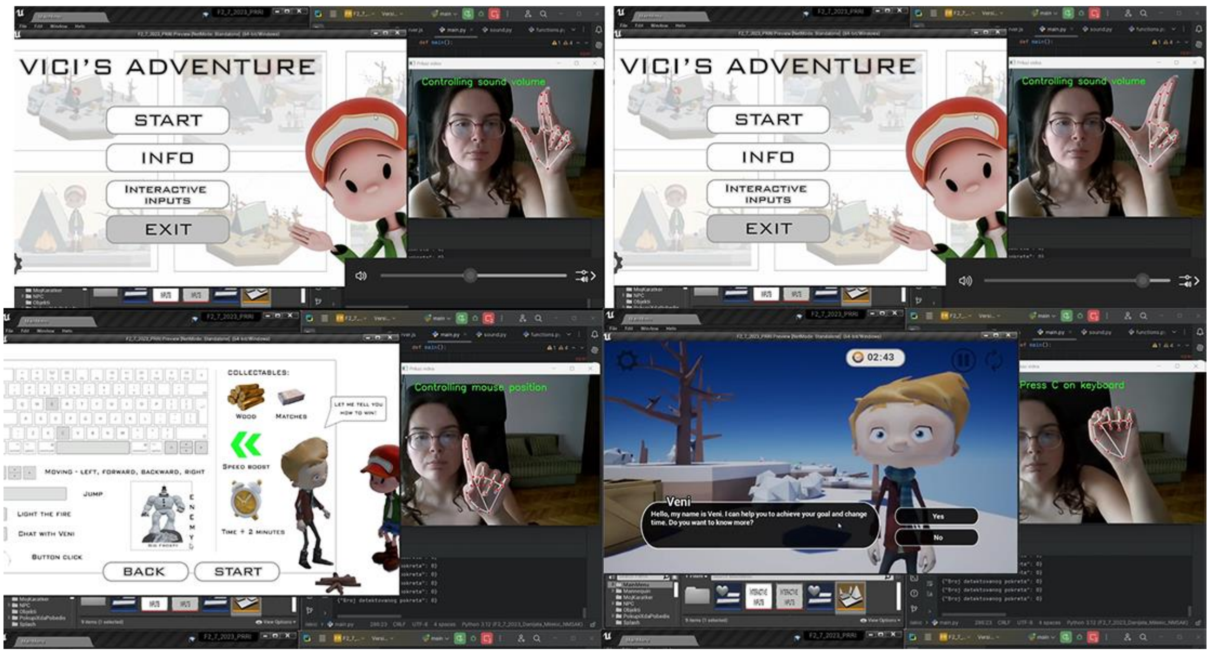


*Fig.3. (a) Features, horizontal and vertical eye line; (b) Testing the threshold value of the ratio*

## 3 UNREAL ENGINE FUNCTIONALITY

The user interface control doesn't require modifications in Unreal Engine, as PyAutoGUI handles the entire system, enabling mouse cursor movement, left click, and the "C" key press within the program. The UI allows users to adjust the in-game sound volume, pause, restart, view information, quit the game, or initiate a conversation with the in-game mentor. PyAutoGUI therefore translates gestures into standard GUI actions, and the resulting interface is illustrated in Fig. 4.



Hello, my name is Veni. I can help you to achieve your goal and change time. Do you want to know more?

*Fig.4. Graphical User Interface Control*

Based on the received messages, which depend on hand movements and eye blinks, character control within Unreal Engine becomes dynamic and responsive. When a hand movement is recognized correctly and has an appropriate value, the corresponding message "Movement forward/ backward/left/right" is sent (Fig.5a). If no recognizable movement is detected, the message "No movement" is sent. This ensures that the character only moves when intentional gestures are made.

In addition to the hand gestures, eye blinks are utilized to trigger specific actions. Based on the value of the blink detection, the string is extended with "+ Jumping" (Fig.5b) or "+ Flying" (Fig.5c). The blink of the right eye triggers a jumping action, while the blink of the left eye activates flying mode.

As the messages are received, the character responds in real-time, executing the appropriate movement or action. This allows for an interactive and immersive experience where the character's behavior directly correlates with the user's gestures.

It should be noted that the system assumes users have full mobility of their upper limbs and facial muscles, as the interaction relies heavily on hand gestures and eye blinks. Users with physical limitations affecting these areas may experience reduced or non-functional interaction capabilities.

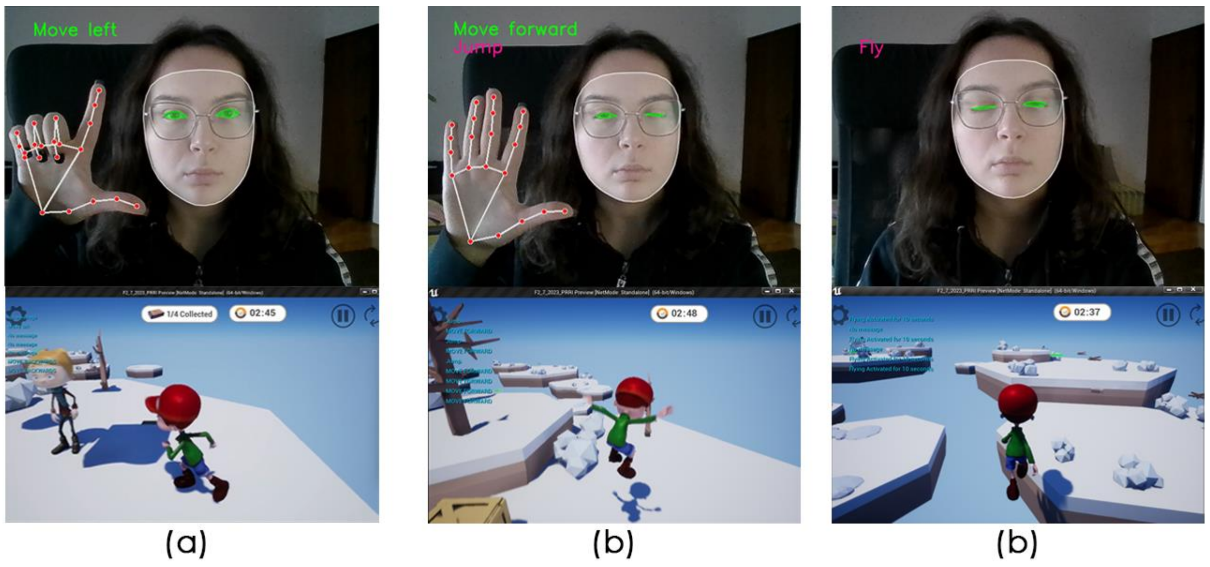


*Fig.5. (a) Simple movement; (b) Simultaneous movement and jumping; (c) Change of movement mode – flying*

## 4 CONCLUSION

This paper explores the use of computer vision to control user interfaces in interactive software. The main challenge was the simultaneous transmission of data about hand movements and eye blinks. The mathematical aspect is challenging, especially regarding methods for determining hand positions, which depend on factors such as distance, hand size, and camera distance. Future improvements could include analyzing relationships between these factors or training a custom model to recognize specific controls, enabling more precise game management and enhanced user interaction. During testing, it was observed that the system's performance was sensitive to lighting conditions. The camera struggled in low-light environments, which significantly reduced the accuracy of detection. These limitations highlight the importance of proper environmental conditions for reliable operation of the system.

## ACKNOWLEDGEMENTS

This research has been supported by the Science Fund of the Republic of Serbia, #7449, Multimodal multilingual human-machine speech communication – AI-SPEAK, and by the Fund of Autonomous Province of Vojvodina for the project: Application of modern methods of computer graphics and simulation for the development and 3D printing of a tactile map of the Campus, University of Novi Sad (No.000661458/2024/09318/004/000/000/001/04/003); and by the Faculty of Technical Sciences, University of Novi Sad, through the project “Scientific and artistic research work of researchers in teaching and associate positions at the Faculty of Technical Sciences, University of Novi Sad 2025” (no. 01-50/295).

## REFERENCES

[1] Clark A.; Moodley D. (2016). A System for a Hand Gesture-Manipulated Virtual Reality Environment.
http://dx.doi.org/10.1145/2987491.2987511

[2] Wong W. K.; Juwono Fi; Khoo B. (2021). Multi-Features Capacitive Hand Gesture Recognition Sensor: A Machine Learning Approach. IEEE Sensors Journal. PP. 1-1.
http://dx.doi.org/10.1109/JSEN.2021.3049273

[3] Romera-Paredes B.; Marín-Jiménez M.; Tzimiropoulos G. (2023). Real-Time Multi-Modal Hand–Face Pose Estimation for Mixed-Reality Interaction. IEEE Transactions on Pattern Analysis and

Machine Intelligence, 45(8), 9141-9155.
http://dx.doi.org/10.1109/TPAMI.2023.3245678

[4] Indriani; Moh H.; Agoes A. (2021). Applying Hand Gesture Recognition for User Guide Application Using MediaPipe.
http://dx.doi.org/10.2991/aer.k.211106.017

[5] Szeliski, R. (2010). Computer Vision: Algorithms and Applications (2nd ed., draft PDF) Available at: https://szeliski.org/Book/ (Accessed: 24. April 2025).

[6] Lê M.; Doan M.; Nguyen D. D.; Nguyen M. S. (2022). Smart Desk in Hybrid Classroom: Detecting student's lack of concentration when studying. 13-18.
http://dx.doi.org/10.1109/NICS56915.2022.10013468

[7] Alanazi, F.; Ushaw, G.; Morgan, G. Improving Detection of DeepFakes through Facial Region Analysis in Images. Electronics 2024,13, 126.
http://dx.doi.org/10.3390/electronics13010126