International Journal of Scientific Research in Computer Science, Engineering and Information Technology

ISSN : 2456-3307 OPEN ACCESS

Available Online at : www.ijsrcseit.com

doi : https://doi.org/10.32628/IJSRCSEIT

The Integration of OpenCV and Unity for the Development of Interactive Educational Game

Jinish Shah*, Pradhyuman Pandey, Tanmai Kamat, Prachi Tawde

Department of Information Technology, Dwarkadas J. Sanghvi College of Engineering, Mumbai, Maharashtra, India

ARTICLEINFO

Article History:

Accepted: 10 Aug 2023

Published: 30 Aug 2023

Publication Issue

Volume 9, Issue 4

July-August-2023

Page Number

425-431

ABSTRACT

In recent years, online and interactive educational games have become increasingly popular as a tool to improve and accelerate the learning experience for students. Computer vision libraries in python such as OpenCV provide a wide range of tools for image processing, object detection, and face recognition that can be used to create interactive and engaging games. This paper explores the use of OpenCV in Unity 3D for the development of interactive online educational games. Specifically, we discuss how OpenCV can be used to create interactive games that respond to user actions in real-time, such as tracking the movement of a user's hand or face to control in-game actions. It focuses on an unconventional approach to interactive game development using OpenCV combined with a computer network based protocol namely User Datagram Protocol (UDP) as communication channel to transfer real-time data to Unity Game Engine from Python Server. Though being the conventional approach for object and face tracking based games Augmented Reality (AR) requires specialized hardware and high camera quality for accurate tracking of object position and orientation. Given the lack of technical infrastructure and computational resources in majority government schools under the Indian academic education system this approach is not feasible for large scale implementation. This research paper aims to contribute to the development of innovative and effective approaches for interactive and online educational games suitable for large scale implementation in the form of a Vlab (Virtual Lab).

Keywords : OpenCV, Interactive educational game, Unity 3D, Computer Vision, User Datagram Protocol (UDP)

I. INTRODUCTION

Virtual Labs are gaining popularity in educational institutions and research organizations, particularly when physical lab access is limited or impractical. A Vlab (Virtual Lab) is an online platform or piece of software that enables users to access and interact virtually with various laboratory simulations and

Copyright © 2023 The Author(s): This is an open-access article distributed under the terms of the Creative Commons Attribution **4.0 International License (CC BY-NC 4.0)** which permits unrestricted use, distribution, and reproduction in any medium for non-commercial use provided the original author and source are credited.

experiments. It provides a simulated environment in which students, researchers, and professionals can conduct experiments, conduct analyses, and observe results without the need for actual lab equipment. Virtual Labs are intended to enhance learning and research by providing a safe, cost effective, and accessible alternative to conventional hands-on labs. They provide an excellent supplement to conventional laboratory experiences, allowing a wider audience to participate in scientific experiments and research. Additionally, Virtual Labs can be especially useful in environments where access to on-campus labs is difficult for students studying remotely. The learning materials that integrate these technologies are web sites, computer learning packages, virtual field trips, simulations and Vlabs [9]. With the use of this powerful technology to support teaching and learning, the process of teaching science subjects can be made more fun and effective [8]. The following project is a Skeleton Game that was developed as an attempt to establish a virtual laboratory (Vlab) for the purpose of teaching and explaining the fundamentals of human anatomy, with a special focus on the human skeletal system, to students in grades 5 through 8, without the use of real laboratory equipment or models. It entails adding gaming elements to a virtual laboratory in order to provide the user with an interactive learning experience that will make the learning process more fun.

The proposed method utilizes OpenCV Python Server for object tracking and Unity Game Engine for graphics rendering and gameplay. Thus it requires real-time communication between the two isolated systems Unity Game Engine and OpenCV Python Server to create interactive gaming experiences that respond to the user's actions in real-time. The technique involves detecting the target points from the input image or video feed using OpenCV, packaging the target point coordinates as a UDP message, sending the message to Unity, and receiving the message in Unity to control the game objects in the scene. This approach provides a fast and efficient way to transmit data between OpenCV Python server and Unity. The proposed method can provide a different approach to the development of educational games compared to the current convention by allowing for the creation of immersive and interactive games that respond to the user's actions in real-time. However, as with any computer network communication protocol, using User Datagram Protocol (UDP) may have some limitations, such as limited packet size and no error checking. These limitations should be taken into consideration when deciding whether to use UDP or another communication protocol for a specific application. In this paper, we implement and discuss the feasibility of establishing a UDP connection between OpenCV Python server and Unity and evaluate its performance and usability. Overall, the proposed method is an approach for the development of interactive and engaging educational games and Vlabs.

II. LITERATURE SURVEY

A. Review of existing research and tools

Augmented Reality (AR) has gained significant attention in recent years, revolutionizing various fields, including gaming, education, and industrial applications [3]. This section presents a literature survey of different existing technologies and methods to implement AR, including ARCore and ARKit, as well as the integration of OpenCV with AR. Each method is described in detail, highlighting its features, benefits, and limitations. Furthermore, a brief summary of each method is provided, followed by the identification of possible drawbacks in both ARCore and ARKit that are resolved by using OpenCV.

1) ARCore: ARCore, developed by Google, is a widely adopted platform for building AR applications on Android devices [5]. It utilizes three key technologies: motion tracking, environmental understanding, and light estimation. Motion tracking enables the device to

track its position and orientation in the physical world, allowing virtual objects to be placed and anchored in the real environment. Environmental understanding uses feature points and visual odometry to detect flat surfaces and align virtual objects with the real world. Light estimation provides realistic lighting conditions for virtual objects, enhancing their visual integration with the real environment. Thus, ARCore provides robust motion tracking, environmental understanding, and light estimation capabilities, facilitating the seamless integration of virtual objects into the real world. However, a limitation of ARCore is its reliance on predefined feature points and flat surfaces for accurate tracking. In dynamic environments or scenarios with limited surface features, tracking performance may degrade, leading to a less immersive AR experience.

2) ARKit: ARKit, developed by Apple, is a framework for creating AR experiences on iOS devices [7]. It offers features such as world tracking, scene understanding, and rendering. World tracking enables accurate motion tracking of the device, allowing virtual content to be anchored to the real world. Scene understanding analyses the environment, detecting and recognizing objects, and providing spatial mapping information. ARKit also includes advanced rendering techniques, such as lighting and occlusion, to enhance the visual quality and realism of AR experiences. Therefore, ARKit provides robust motion tracking, scene understanding, and advanced rendering capabilities, enabling the seamless integration of virtual content with the real world on iOS devices. Common limitation of ARKit is its reliance on visible feature points and well-textured surfaces for accurate tracking. In challenging lighting conditions or environments with limited features, tracking accuracy may be compromised, affecting the overall AR experience.

3) OpenCV integration with AR: OpenCV, an open-source computer vision library [6]. By integrating OpenCV with AR frameworks like ARCore and ARKit, additional computer vision capabilities can be unlocked. OpenCV provides powerful algorithms for image processing, object detection, and tracking. These functionalities can be utilized to improve the robustness and accuracy of motion tracking, overcome limitations in feature detection, and enable more advanced interaction techniques in AR applications [2]. Thus, integrating OpenCV with AR frameworks enhances the computer vision capabilities of AR systems, enabling more accurate and robust tracking, object detection, and interaction techniques. Key drawback in both ARCore and ARKit is their reliance on predefined feature points and well-textured surfaces for accurate tracking. By integrating OpenCV, which offers robust object detection and tracking algorithms, the dependency on predefined features can be mitigated. OpenCV allows for more robust tracking in dynamic environments and scenarios with limited feature points, improving the overall stability and accuracy of AR experiences. ARCore and ARKit are prominent frameworks for implementing AR experiences on Android and iOS devices, respectively. While both provide robust tracking and scene understanding capabilities, they have limitations in terms of tracking accuracy in dynamic or feature-scarce environments. By integrating OpenCV, the limitations of predefined feature-based tracking can be overcome, improving the robustness and accuracy of AR experiences. The combination of AR frameworks and OpenCV presents a promising direction for advancing the field of augmented reality and creating more immersive and interactive applications.

III. METHODOLOGY

A. Image processing and detection

The use of OpenCV-based Python server for image processing and detection has been widely studied and implemented in various applications. This method involves the use of a computer vision library such as OpenCV to extract relevant information from input images, such as the coordinates of target points [6]. The extracted information is then transmitted to a host

application, such as Unity, for further processing and display. The process of image processing and detection involves the use of various techniques, such as edge detection, colour segmentation, and feature extraction, to extract meaningful information from the input images. These techniques are implemented using OpenCV's vast array of computer vision algorithms, which have been shown to be effective in various applications such as object detection, face recognition, and autonomous navigation. The use of a Python server for image processing and detection allows for real-time analysis of input images, which is essential in many applications. This method has been shown to be effective in applications such as robotics, augmented reality, and computer vision research, where real time processing and detection are necessary. The integration of OpenCV-based Python server with Unity Game Engine allows for the creation of interactive applications that can track and respond to the movement of real-world objects. This technology has been used in various applications, such as virtual reality, 3D modelling, and interactive gaming. The real-time tracking and detection of objects can provide a more immersive and engaging user experience. The use of OpenCV-based Python server for image processing and detection is a powerful tool in various applications where real-time processing and detection are necessary [4].

B. Data Transmission

Socket programming involves using a communication end-point called a socket, which enables two processes to communicate with each other. UDP (User Datagram Protocol) is a connectionless transport layer protocol that operates on top of IP (Internet Protocol) and is commonly used for real-time applications such as video streaming, online gaming, and voice over IP (VoIP) due to its low-latency communication and lack of connection establishment.

For the purpose of connecting the OpenCV Python server to Unity, a UDP socket connection can be employed to transmit image data and other information between the two systems [1]. In Python, the socket library can be used to create a UDP socket server that listens for connections from Unity. Unity, on the other hand, can utilize the Socket class to connect to the Python server and send and receive data. One of the key benefits of using UDP is its low-latency, which makes it suitable for real-time applications such as interactive educational games. In addition, UDP is connectionless, which means that it does not require the establishment of a connection and can transmit data packets quickly and efficiently. However, the use of UDP does have limitations, such as limited packet size and no error checking. These limitations should be taken into account when deciding whether to use UDP or another communication protocol for a specific application.

IV. SYSTEM ARCHITECTURE

The system architecture (refer Figure 1) is extensive and covers three major components of the working of the system Python Server, Unity Server and UDP Socket connection.

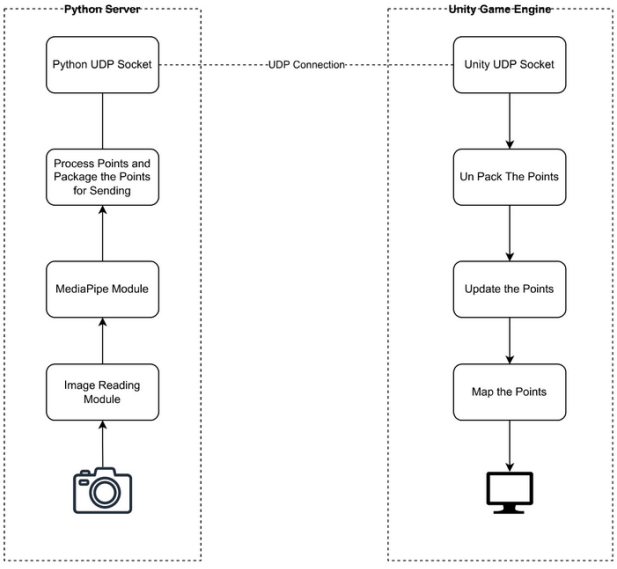

Figure 1. System Architecture

The Python Server consists extensively of the video stream coverage where the Image reading module is applied and the target points are identified through

Mediapipe Module in order to be embedded in the Unity game scene. Next, the target coordinates from the video stream are processed and packaged in UTF-8-byte array format to be sent through Python UDP socket programming connection. Next the Unity Server accepts the incoming byte array and converts it to string format and updates the points in real-time on the Unity game scene. The following connection steps and how they have been performed: For a UDP connection between an OpenCV Python server and a running instance of a game developed using Unity, the following steps are followed: a) OpenCV Image Processing: The first step is to use OpenCV to detect the target points from the input image or video feed necessary for object mapping and graphic rendering. These target points can be the coordinates of the hands, face, or any other object of interest. Once these target points are detected, they can be packaged as a message and sent to Unity. b) Sending the UDP Message: In Python, a UDP server can be created using the socket library. Once the target point coordinates are converted into a UTF-8 format byte array that can be transmitted over the UDP socket, the message can be sent to Unity. In Python, the socket library can be used to send the message to a specified IP address and port number. In this case, the IP address would be the IP address of the device running Unity and the port number would be the port number that the Unity client is listening on. c) Receiving the UDP Message in Unity: In Unity, a UDP client can be created using the Socket class. The client listens for incoming messages on the specified port number. When a message is received, the client can extract the target point coordinates from the message and use them to control the game objects in the scene. d) Embedding the Target Points in the Game Scene: Once the target point coordinates are received in Unity, they can be used to control the game objects in the scene. For example, the target points could be used to control the real-time movement of a character or the position of a virtual object. After the target points are embedded into the scene, the user can interact with the game object in various ways, such as through user input or collision detection. For example, if the target point represents a button, the game can interact with it by detecting user input, such as a mouse click, and trigger a specific action. Alternatively, if the target point represents a moving object, the game can use collision detection to detect if another object collides with it and trigger an appropriate action. The game may also provide feedback to the user based on the detected target points. For example, if the target point represents a collectible object, the game may display a message indicating that the object has been collected. Alternatively, if the target point represents an obstacle, the game may trigger a visual or audio effect indicating that the obstacle has been hit.

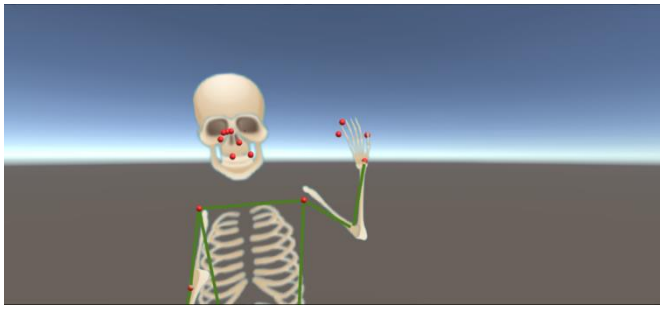

Figure 2. A demonstration of the graphics rendered as Gameobject in Unity3D based on the target points embedded in the game scene.

The UDP connection between the OpenCV Python server and Unity Game Engine allows for real-time communication between the two systems. The UDP connection can be used in integration of real-time image processing and detection in the game. This integration allows for the use of target point coordinates detected by OpenCV to embed objects or other game elements into the scene as shown in Fig. 2.

A. Limitations of UDP

While using UDP there are some limitations that also need to be considered before implementing it for Unity Client and Python Server communication. One of the primary limitations of UDP is its lack of reliability. UDP does not provide error checking, flow control, or retransmission of lost packets, which can lead to data

loss or corruption. This is in contrast to Transmission Control Protocol (TCP), which offers reliable, ordered, and error-checked delivery of data. In our case, this lack of reliability may not be a significant issue for real-time applications like interactive educational games, but in other cases, it may be critical to ensure that data is transmitted and received correctly. Another limitation of UDP is its packet size. UDP has a maximum packet size of around 65.507 kilobyte, which can be a problem for applications that require the transmission of large amounts of data. This can lead to the need to split data into multiple packets, which can increase the overhead and complexity of the transmission.

V. RESULTS

In order to understand the impact of the overall game on the users, an extensive survey on a sample of 30 high school students conducted through google forms after playing the game. The aim of the survey was to study the results of the OpenCV based game on a sample audience. It also served as feedback for potential improvements that can be made to make the game more effective in future applications of educational games. The statistical excerpts of the survey indicated that the game can accurately detect and track objects in real-time, allowing for responsive and interactive gameplay (refer Fig. 3). The performance of the implemented OpenCV based game is evaluated across multiple parameters like user experience, student's understanding of concepts after playing the game, and accuracy of the movement tracking graphics being rendered in real-time. The results are conclusive and indicate the potential of OpenCV in enabling immersive augmented reality experiences, gesture-based controls, and environmental mapping within Unity games for educational initiatives.

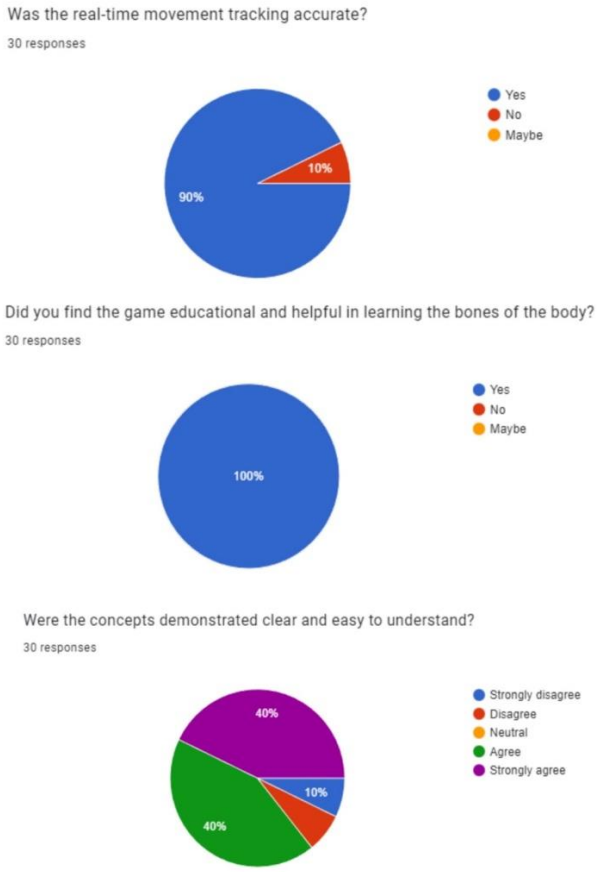

Figure 3. A sample line graph using colours, which contrast, well both on screen and on a black-and-white hardcopy

VI.CONCLUSION

The integration of OpenCV, a powerful computer vision library, with Unity, a popular game development platform, has the potential to enhance game development by enabling real-time image processing and analysis and overcoming the challenges in implementing AR. With the help of UDP based Socket communication between the Unity Server and OpenCV python server, a fast real-time data transfer and subsequent rendering of graphics was achieved which allowed for enhanced learning experience through interactive gameplay based on relevant topics under the academic curriculum. This configuration of system architecture has shown robust and reproducible results and has potential to be applied on other game-based applications where the use of Augmented Reality may not be feasible for real-time graphics rendering.

However, OpenCV integration in Unity has several limitations, including potential performance decreases, complexity, functionality, compatibility, and maintenance issues. This can negatively impact user experience, especially on lower end devices. The integration process is complex and time consuming, requiring expertise in computer vision and Unity programming. Compatibility issues may arise, as OpenCV may not be compatible with all Unity platforms. Maintenance of OpenCV, an open-source library, may require ongoing effort and resources, making it challenging for small development teams or individual developers.

VII. REFERENCES

[1] G. Garg and S. Shivani," Controller free hand interaction in Virtual Reality," 2022 OITS International Conference on Information Technology (OCIT), Bhubaneswar, India, 2022, pp. 553-557, DOI: 10.1109/OCIT56763.2022.00108.

[2] S. L. Kim, H. J. Suk, J. H. Kang, J. M. Jung, T. H. Laine and J. Westlin," Using Unity 3D to facilitate mobile augmented reality game development," 2014 IEEE World Forum on Internet of Things (WF-IoT), Seoul, Korea (South), 2014, pp. 21-26, DOI: 10.1109/WFIoT.2014.6803110.

[3] B. A. Koca and B. Cubukcu," Augmented Reality Application for Preschool Children with Unity 3D Platform," 2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Ankara, Turkey, 2019, pp. 1-4, DOI: 10.1109/ISMSIT.2019.8932729.

[4] R. M. Gurav and P. K. Kadbe," Real time finger tracking and contour detection for gesture recognition using OpenCV," 2015 International Conference on Industrial Instrumentation and Control (ICIC), Pune, India, 2015, pp. 974-977, DOI: 10.1109/IIC.2015.7150886

[5] Introduction OpenCV. Available at: https://docs.opencv.org/4.x/d1/dfb/intro.html (Accessed: 11 July 2023).

[6] Arkit Apple Developer Documentation. Available at: https://developer.apple.com/documentation/arkit (Accessed: 11 July 2023).

[7] Google Developers Overview of arcore and supported development environments — google for developers, Google. Available at: https://developers.google.com/ar/develop (Accessed: 11 July 2023).

[8] E. Smeets," Does ICT Contribute to Powerful Learning Environments in Primary Education" Computers and Education, Vol. 44, No 3, pp 343-355. 2005.

[9] M. Peat and A. Fernandez," The Role of Information Technology in Biology Education: an Australian perspective". Journal of Biological Education. Vol 34, 2000, pp. 69-73.

Cite this article as :

Jinish Shah, Pradhyuman Pandey, Tanmai Kamat, Prachi Tawde, "The Integration of OpenCV and Unity for the Development of Interactive Educational Game", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 9, Issue 4, pp.425-431, July-August-2023. Available at doi : https://doi.org/10.32628/CSEIT2390452
Journal URL : https://ijsrcseit.com/CSEIT2390452