International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. #, No.# (Mon-20..)

Z-HandAR: Handheld Augmented Reality Occlusion Handling Framework using WebRTC-based Depth Sensor Streaming

Muhammad Anwar Ahmad[1], Norhaida Mohd Suaib[2], and Ajune Wanis Ismail[1]

[1] ViCubeLab, Faculty of Computing, Universiti Teknologi Malaysia, 81310 Johor, Malaysia
[2] UTM Big Data Center, Ibnu Sina Institute of Scientific and Industrial Research, UniversitiTeknologi Malaysia, 81310 Johor, Malaysia

E-mail address: muhd.anwar135@gmail.com, haida@utm.my, ajune@utm.my

Received ## Mon. 20##, Revised ## Mon. 20##, Accepted ## Mon. 20##, Published ## Mon. 20##

Abstract: In this study, the integration of a handheld device with a ZED Mini depth sensor is explored to produce Z-HandAR, a framework that enhances occlusion handling in handheld Augmented Reality (AR) applications. The aim is to improve the occlusion handling capabilities of the conventional AR depth-based framework by leveraging the powerful stereo vision technology of an RGBD depth sensor. By combining the handheld device with the depth sensor, we can benefit from accurate depth information for more realistic AR experiences. The setup involves the fusion of handheld device with depth sensor via a fusion module involving Unity Render Streaming, a WebRTC-based streaming framework. This allows the handheld device to access the frame data captured by the depth sensor in real-time. For this study, we conducted a pilot test to get the finding where we aim to enhance occlusion handling and reduce flickering in handheld AR applications. We compare our results with one of the state-of-the-art handheld AR frameworks, Lightship ARDK. Our results show improvement over Lightship ARDK's handheld AR occlusion handling. This paper explores a promising potential for improving occlusion handling in handheld AR applications. By leveraging the stereo vision technique of the depth sensor, we present the experiments and framework of this integration could lead to significant advancements in handheld AR technology.

Keywords: Augmented Reality, handheld device, occlusion handling, WebRTC streaming, depth sensor

1. Introduction

Augmented Reality (AR) combines real and virtual objects through specific displays. One of the displays is handheld devices [1]. Handheld AR refers to AR experiences that are accessed through handheld devices such as smartphones or tablets. One of the remaining issues in handheld AR is occlusion [2]. Occlusion occurs when objects closer to viewer obscure the view of object further away along line-of-sight [3]. Users will have the misconception that the real object is further from the viewpoint than the virtual objects when the virtual objects are occluded by the real objects in the scene [4]. Occlusion handling aims to understand the surroundings and how virtual objects are partially or fully occluded by real objects. By definition, occlusion handling is a set of techniques to mitigate the effects of occlusion when doing inference from image [5]. In context of AR, when the system detects overlapping between real and virtual objects, the system needs to optimize rendering accordingly by not drawing what is not visible [6]. Occlusion-free is a scenario where occlusion is no longer a concern. Thus, occlusion-free terminology has known as the solution to occlusion issue.

Depth-based methods for occlusion handling in handheld AR applications utilize depth information to enhance the realism and accuracy of virtual object placement within real-world scenes [7]. Unlike traditional model-based approaches, which rely on predefined 3D models to approximate occlusion, depth-based methods directly analyze depth data obtained from depth sensors or depth estimation algorithms [8]. These methods offer several advantages, including the ability to handle both static and dynamic scenes, adaptability to various environments ranging from simple to complex, and improved occlusion accuracy. By leveraging depth information, these methods enable AR applications to better understand the spatial relationships between virtual and real objects, resulting in more realistic and immersive user experiences [9]. Additionally, depth-based methods facilitate tasks such as lighting estimation, collision

E-mail:author's email

detection, avatar pathfinding, and spatial mapping, further enhancing the capabilities of AR systems. Through a comparative analysis of depth-based approaches, researchers aim to identify the most effective techniques for occlusion handling in handheld AR applications, contributing to advancements in AR technology and user interaction.

This paper proposes a framework that integrates handheld AR with ZED Mini, a stereo vision-based RGBD depth sensor via a fusion module consisting of a WebRTC-based streaming. A pilot test has been developed on handheld AR to obtain the initial results of this framework. This paper also studies about spatial mapping on handheld AR. The spatial mapping is a process to generate spatial map as the output. It can be achieved using specific methods involving algorithms and techniques for depth data and transforming it into a spatial map.

2. Related Works

A. *Depth-based Handheld AR Occlusion Handling*

Depth-based methods for occlusion handling in AR rely on depth information for determining the relative positioning between virtual and real object in the scene. Generally, depth-based handheld AR occlusion handling can be classified with three main approaches [10], which are using embedded depth-sensor, learning-based depth prediction, and monocular depth estimation. Tian and Ma's work [11] used a depth sensor called time-of-flight (ToF) camera for occlusion handling and collision detection on an Oppo R17 Pro smartphone. However, ToF lack geometric details that can hinder certain physics simulations that necessitate detailed geometry. The method from [12] used a stereo vision technique called stereo matching to estimate depth using monocular camera. Their method is now integrated in Google ARCore as Depth API which is now accessible in millions of Android-based phones worldwide [13]. However, the limitation of this method is that it cannot accurately track objects in motion [10]. Lightship ARDK is another state-of-the-art handheld AR framework which utilizes learning-based depth-prediction model for estimating depth and performing occlusion handling. Based on [12], they found that they were able to effectively enable handheld AR application to be occlusion-aware. Depth can be used not only to support the process of occlusion handling, but also for better understanding of the real scene such as 3D shape of the objects and the global illumination condition of the scene [2]. [12] has applied depth-based method and focus on improving occlusion, while Du et al. [14] had used depth-based method has been applied for lighting estimation, collision detection, avatar pathfinding, depth-aware object placement, depth of field. Occlusion problems still can be handled through model-based method; however, the depth-based method can cater for both static or dynamic scene and suitable for simple to complex environment while model-based method is suitable for simple and static scene [6]. However, these approaches have limitations when it comes to accurately estimating the depth such as pose imprecision, low-textured areas, and hardware constraints that can cause rolling shutter artifacts and motion blur [12].

Based on the limitations of the related studies, we explored the potential of harnessing RGBD depth sensor for the purpose of reconstructing spatial mapping occlusion handling in handheld AR. RGBD depth sensors can produce a spatial awareness that allows real time occlusion handling based on a study by Burger et al. [15]. The authors compared three different depth sensors, which are devices that can measure the distance and shape of objects, for use in surgical simulation. From the comparison, it is found that the Intel D405 sensor is the most suitable for close-range applications, such as reconstructing the shape of the heart valve, but it has some limitations, such as not being able to handle reflective surfaces or thin structures. The ZED Mini sensor is the best choice for applications that need high speed and low resolution, such as tracking the movement of the tools. The Intel D415 sensor fails to reconstruct the heart valve models and is not recommended for surgical simulation. The authors also mentioned that ZED Mini's *NEURAL* mode is a good compromise between the Z -accuracy and fill rate which is the fraction of pixels that contain valid measurements within a region of interest. There have been research on integration of RGBD depth sensor with other types of AR use cases. Wolf et al. [16] used head mounted display with ZED Mini to allow the use of the depth sensor's spatial awareness capabilities to measure the mental load of users who wear a video see-through AR. Martini et al. [17] used ZED Mini in a system that combines 3D object detection, depth data from stereo cameras, and VR environment rendering to create an immersive virtual reality experience that avoids collisions with real obstacles and enhances interaction with virtual objects. This study determined the suitable RGBD depth sensor for occlusion handling in AR use case that require movements, which is essential in AR interactions.

B. *Spatial Mapping in AR*

Spatial mapping in AR involves the process of generating a digital representation of the physical environment, often referred to as a spatial map, to enable virtual objects to interact with the real world more effectively. Depth-based methods play a crucial role in spatial mapping by providing accurate depth information that can be used to construct and update the spatial map in real-time. By integrating depth data obtained from depth sensors or depth estimation algorithms, AR systems can create a detailed and precise spatial map that accounts for the 3D geometry of the environment, including the shapes and positions of objects, surfaces, and obstacles [18], [19]. This process enhances the realism and believability of virtual content overlaid onto the real world, as virtual objects can accurately interact with physical surfaces and respond to changes in the environment. By utilizing depth information captured by the device's sensors, these methods allow virtual content to appear realistically anchored to physical surfaces and objects in the environment [20]. Reconstructing the spatial meshes also allows improvement of feature point depth estimates that are obtained by sparse depth information [21].

Depth-based methods play a critical role in spatial mapping by providing the depth data necessary to construct and update the spatial map in real-time. This allows handheld AR applications to create dynamic and interactive experiences where virtual content seamlessly integrates with the user's surroundings. The raw depth data obtained from the sensors is processed to generate a depth map once the session has started, which is a 2D representation of the 3D geometry of the scene where each pixel contains depth information. Lightship ARDK [22] is a handheld AR framework that has a depth-based algorithm integrated based on ManyDepth, which is a deep learning monocular depth estimation model inspired by multi-view stereo (MVS) [23]. This framework supports spatial mapping generated from the depth estimation model. However, in literature there is limited discussion surrounding this framework, thus in this paper we provide a comparison of the spatial mapping and occlusion handling output with our framework.

C. *WebRTC-based Streaming*

Web Real-Time Communication (WebRTC) is an open-source project that allows audio and video communication to work inside web pages [24]. WebRTC is an open standard for real-time communication of video and audio via a web browser. It provides JavaScript Application Programming Interface (API) to realize the audio/video functions and has features such as peer-to-peer communication. In literature, WebRTC-based video streaming has been explored since 2013 [25]. Since then, the technology had improved drastically such that it has been integrated into the popular game engine Unity, known as Unity Render Streaming Framework [26]. This framework allows real-time frame data streaming to HTML5-based browsers, which can be found on virtually any modern handheld device. This allows the devices to leverage the powerful depth sensing capabilities of the depth sensor over the network. Using network-based solution for leveraging depth sensing devices is not new in the handheld AR domain. Previous researchers have used network to integrate Leap Motion hand tracking device to introduce gesture interaction in handheld AR [27]. Thus, this research follows a similar path of using network-based solution for improving occlusion handling on handheld AR.

3. METHODOLOGY

A. *Experimental Design*

This subsection introduces the proposed framework. Fig. 1 shows the architecture of the framework. The handheld device and the depth sensor are integrated via a fusion module that performs the connection between the devices. We chose ZED Mini for the depth sensor. Physically, the ZED Mini is attached to the back of the handheld device, while connected to a PC running Unity game engine. In Unity, the depth sensor's feed is displayed in real time. Then, each frame is then sent to the handheld device via the Unity Render Streaming Framework. On the handheld device, the feed is displayed via the device's browser.

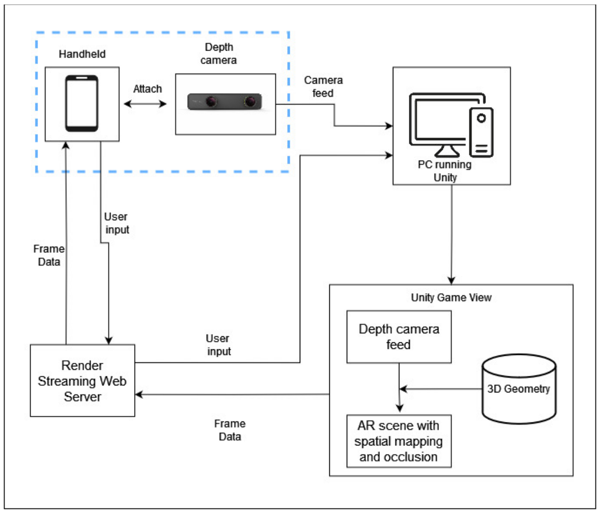

Figure 1. Architecture of the framework

Since the ZED Mini is attached to the handheld device, the depth sensor is now effectively becoming a part of the handheld device. Fig. 2 shows the attachment of ZED Mini to the handheld device. As a result of this attachment, moving the handheld device will also affect the feed displayed in the screen.

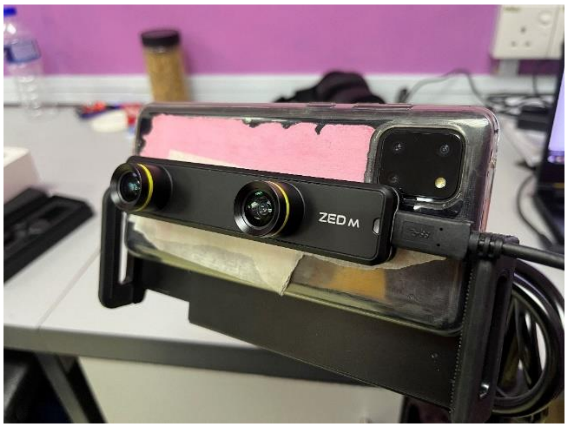

Figure 2. Attachment of ZED Mini to handheld device

B. Fusion of Handheld Device with ZED Mini Module

This subsection describes a module that allows the fusion of handheld device with the ZED Mini RGBD depth sensor. Fig. 3 shows the fusion module hierarchy. The core of this module is the Unity Render Streaming Framework that is based on WebRTC. In the Unity editor, this is where the fusion of the handheld device with the ZED Mini module operates. The ZED_Rig_Mono GameObject provided by the ZED SDK Unity Plugin is now modified to be included in the fusion module with the Unity Render Streaming's prefabs and scripts. In the RenderStreaming prefab, the SignalingManager.cs script manages the communication with signaling servers. It is responsible for sending the frame data output to the receiver interface on the handheld device. It also associates user input on the handheld device with events that is then sent to the server. The Broadcast.cs script contains the list of data that is streamed and received, including the frame data (managed by VideoStreamSender.cs) and input data (managed by InputReceiver.cs). The data is exchanged between this script and the SignalingManager.cs script. The fusion module also contains functions that facilitate the connection between the ZED Mini and handheld device, managed by DeviceDetection.cs script. This function is called before sending or receiving frame data requests.

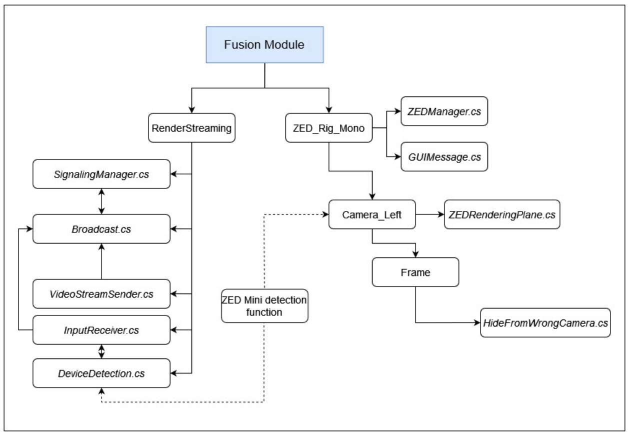

Figure 3. Fusion module

C. WebRTC Connectivity

WebRTC uses a function called bandwidth estimation to ensure real-time video communication. Using bandwidth estimation, the bit rate (the amount of data sent and received per second) can be reduced by automatically lowering the resolution of the transmitted video when bandwidth is low. The user on the handheld device browser can control the application on the server using touch input. The frame data from the depth sensor is streamed to the handheld from the host PC running the Unity Render Streaming Framework at runtime. To enable client connection, the host PC server needs to be set up with OpenSSL certificate, in which the procedure is described in the Unity documentation. The client then needs to enter the IP address of the host server to start the connection. The host server then sends the feed to the client after successfully connected. In Render Streaming a peer-to-peer (P2P) network is created between two peers, and this network sends video/audio/binary data.

The Web server enables communication between two peers. This communication is called signaling. Fig. 4 is an overview of how signaling works [26]. Peer 1 (Sender) and Peer 2 (Receiver) are connected through the signaling server. The sender sends signals to the signaling server, which then forwards the signals to the receiver, establishing a connection for video/audio/binary data exchange directly between the peers. A signaling server is a server that helps two peers find and connect to each other by exchanging information like network data, media data, and session control information. To facilitate the communication between Unity and the Web browser, an application called the web application is needed. It runs on a specific Uniform Resource Locator (URL), which is the location of the Web page that displays the Unity content. The application uses the WebSocket protocol, which is a standard way of exchanging data between a web server and a web client. WebSocket allows for bidirectional and real-time communication, which is essential for interactive and immersive Unity applications. The web application also handles the connection establishment, authentication, and message routing between Unity and the Web browser.

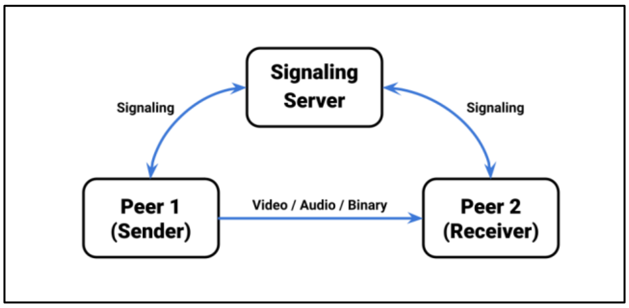

Figure 4. Signaling overview [26]

Peer 1 (Sender) and Peer 2 (Receiver) are connected through the signaling server. The sender sends signals to the signaling server, which then forwards the signals to the receiver, establishing a connection for video/audio/binary data exchange directly between the peers. Fig. 5 provides a more detailed description of peer-to-peer communication using signaling server [26].

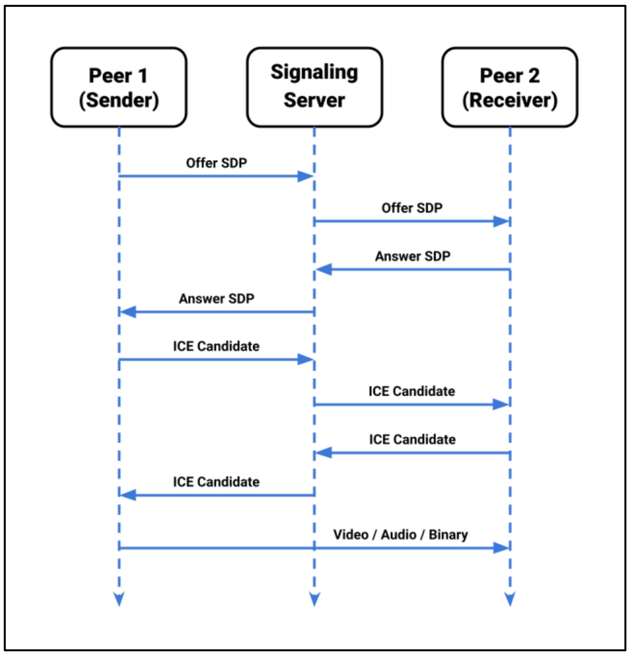

Figure 5. Peer-to-peer communication using signaling server [26]

A signaling server is a server that helps two peers find and connect to each other by exchanging information like network data, media data, and session control information. The figure shows the steps involved in establishing a connection between Peer 1 (Sender) and Peer 2 (Receiver) using WebRTC, a technology that enables real-time communication between browsers and devices. Here are the steps:

- Peer 1 creates an Offer Session Description Protocol (SDP), which is a description of the media capabilities and preferences of Peer 1. Peer 1 sends the offer SDP to the signaling server, which forwards it to Peer 2.
- Peer 2 receives the Offer SDP and creates an Answer SDP, which is a description of the media capabilities and preferences of Peer 2 that are compatible with the Offer SDP. Peer 2 sends the Answer SDP to the signaling server, which forwards it to Peer 1.
- Peer 1 and Peer 2 exchange Interactive Connectivity Establishment (ICE) candidates, which are possible ways of connecting to each other over the internet. ICE candidates include information like IP address, port, and protocol. Peer 1 and Peer 2 send ICE candidates to the signaling server, which forwards to the other peer.
- Peer 1 and Peer 2 use the SDP and ICE candidates to negotiate a direct connection for transferring video, audio, or binary data. The signaling server is no longer needed for the communication.

To facilitate the communication between Unity and the Web browser, an application called the web application is needed. It runs on a specific Uniform Resource Locator (URL), which is the location of the Web page that displays the Unity content. The application uses the WebSocket protocol, which is a standard way of exchanging data between a web server and a web client. WebSocket allows for bidirectional and real-time communication, which is essential for interactive and immersive Unity applications. The web application also handles the connection establishment, authentication, and message routing between Unity and the Web browser.

D. Pilot Test Application

This subsection introduces the applications that have been developed for the pilot test. For our pilot test, we perform comparison between our framework and ARDK framework, therefore there are two applications that have been developed. The first application we implemented the ARDK framework and the second application we implemented our framework. The flow for each application differs slightly since our framework incorporates network elements from the WebRTC. Fig. 6 shows the flowchart for ARDK application. When the application starts, the system will scan the scene and perform the spatial mapping. After that the virtual objects are placed by selecting from a dropdown menu and tapping on the screen of the handheld device. Then the experiment is performed by observing the occlusion handling when the virtual objects are covered by real objects. For our experiment, we use a person's hand and a miniature house to cover the virtual objects.

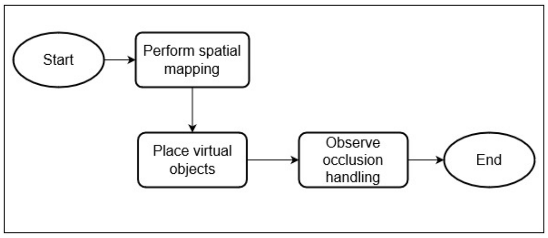

Figure 6. Flowchart for ARDK application

Fig. 7 shows the flow for Z-HandAR framework's application. When the application starts, the system will ask for the IP address of the server to perform the connection. After the connection is successful, the frame data will be start to transmit to the client handheld device and displayed on the screen. Then, the procedure follows the ARDK application's flow.

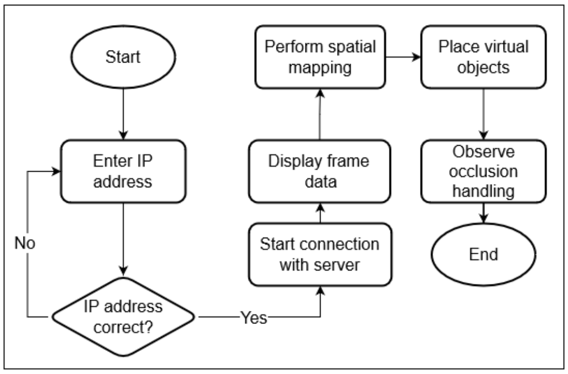

Figure 7. Architecture of the framework

4. Results and Discussions

A. Pilot Test Results

This subsection discusses about the results of the pilot test, with comparison of two applications for spatial mapping and occlusion handling. First, we explain on the user interface (UI) of the applications. We created the same UI for both applications, shown in Fig. 8. The UI of the application consists of a reset button, angle indicator, a virtual object selection dropdown menu, occlusion toggle, and mesh toggle.

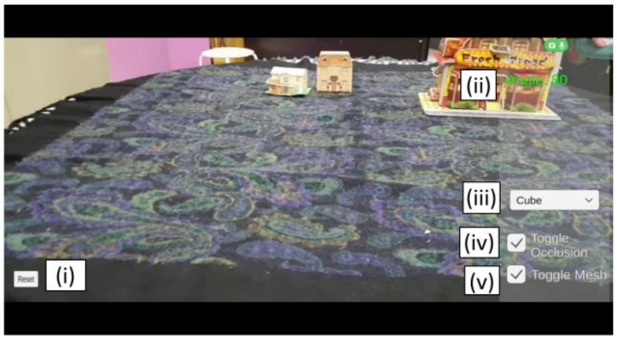

Figure 8. UI of the application

Table 1 shows the functionalities of each UI elements. The reset button is used for resetting the test quickly without closing and restarting the application. The angle indicator is to ensure a similar range of angle for both devices during testing. We chose angles between 75 degrees to 95 degrees for this test. The virtual object dropdown menu is for selecting the AR objects that will be registered in the scene. For this test, we chose basic primitives which are cube, cylinder, and sphere. The occlusion toggle allows switching on and off the occlusion handling effects. When the toggle is on, occlusion handling will occur, and vice versa. Similarly, the mesh toggle allows switching on and off the mesh generated during spatial mapping process. When the toggle is on, mesh will be displayed, and vice versa.

TABLE I. UI Elements Functionalities

UI Element Label	Name	Functionality
(i)	Reset button	resetting the test
(ii)	Angle indicator	ensure similar range of angle for both devices during testing
(iii)	Virtual object dropdown menu	selecting the AR objects
(iv)	Occlusion toggle	switching on and off the occlusion handling effects
(v)	Mesh toggle	switching on and off the mesh generated during spatial mapping process

The results of the spatial mapping process are shown in Fig. 9. In Fig. 9(a), the spatial map of ARDK application is visualized as blue wireframe meshes. In Fig. 9(b), the spatial map of the ZED Mini transmitted to the handheld device using our framework is shown.

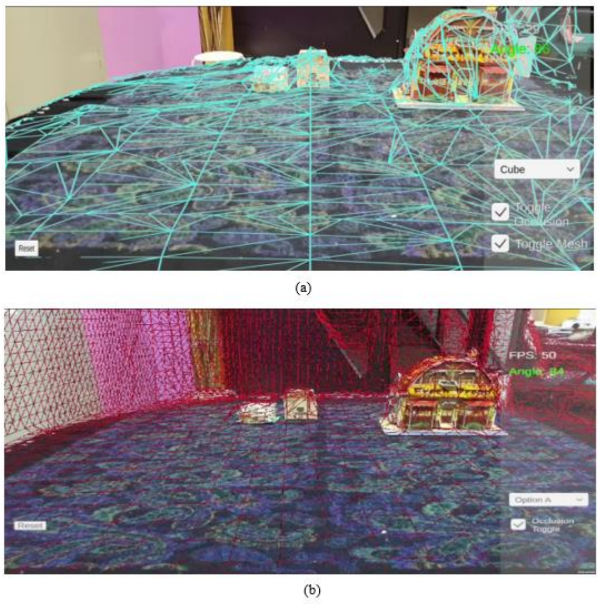

Figure 9. Spatial mapping process of both applications

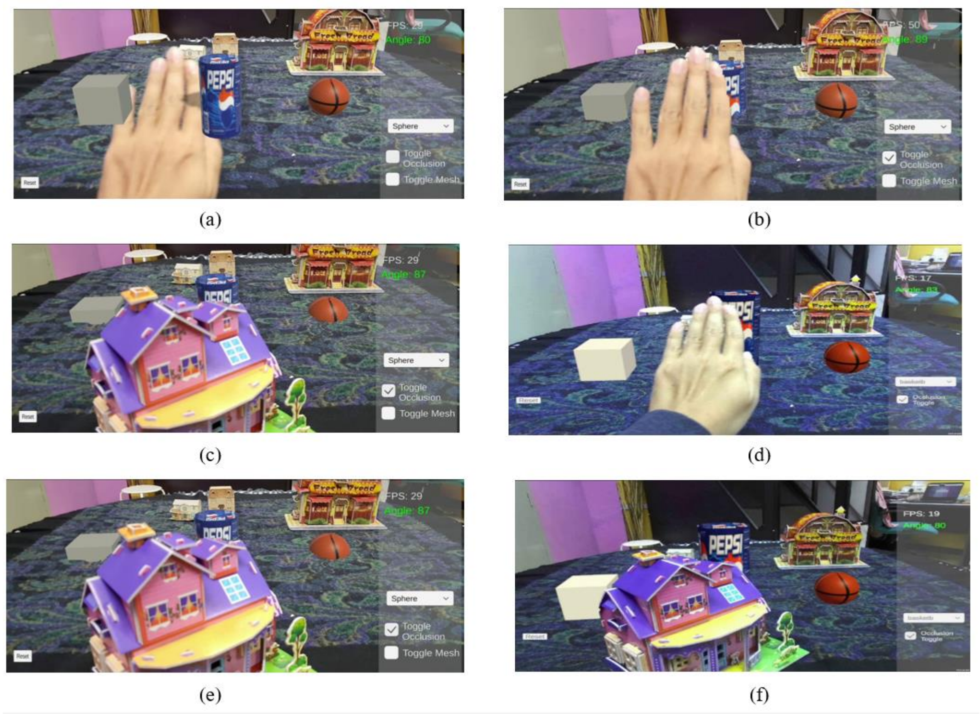

Figure 10. Results comparison between ARDK and Z-HandAR framework

We present the comparative results of the occlusion handling between ARDK and our framework in Figure 9. Fig. 10(a) and Fig. 10(b) show the results of ARDK's occlusion handling. In Fig. 10(a), the Occlusion Toggle is turned off, showing no effects of occlusion handling. In Fig. 10(b), the Occlusion Toggle is turned on, resulting in the pixel of the virtual objects becoming invisible when real objects is obstructing them. However, it can be seen that the boundary between the real object and virtual object are unsmooth. There are also some pixels in other virtual objects that are not within the occluded area that also became invisible, shown in red circle. Figure 10(c) an Fig. 9(d) shows the results of our framework's occlusion handling. In Fig. 10(c), the Occlusion Toggle is turned off, showing no effects of occlusion handling. In Fig. 10(d), the Occlusion Toggle is turned on, once again resulting in the pixel of the virtual objects becoming invisible when real objects is obstructing them. Using this framework, the boundary between the real object and virtual object are smoother. The pixels in other virtual objects that are not occluded were also not affected, showing improvement from the ARDK framework. Fig. 10(e) and Fig. 10(f) shows a side-by-side comparison of the results between the ARDK framework and our framework, using a miniature house as the real object. In Fig. 10(e), the ARDK framework is able to produce a smoother occlusion, however there are still parts of the boundary between the real and virtual object that have visible jaggies, whereas in Fig. 10(f), with our framework the occlusion handling results are smooth on all parts. The issue of other virtual objects that are not affected by the occluding object becoming invisible is also present in the ARDK result.

B. *Discussions*

From the pilot test results, it can be seen that our framework was able to be implemented on handheld AR. The WebRTC-based streaming allows the frame data from the ZED Mini to be transmitted from the server to the handheld device, thereby enabling the handheld device to leverage the ZED Mini's depth sensing features. We observed the spatial mapping and occlusion handling of our framework, and it is an improvement over the ARDK framework.

Since our framework is using a network-based solution, we have to make sure that the internet connection is fast and stable to ensure the frame data is consistently transmitted without hiccups. In this case, ARDK has the advantage of running locally on the device. However, given that the frame data streaming technology we use can be implemented on any device that allows development with Unity, our solution is scalable with future devices.

The Unity Render Streaming documentation described that the network performance does have impact on the resolution quality. Thus, finding a way to optimize the data transmission between the PC server and the handheld device could improve this limitation. Second, the framerate of the proposed framework prototype can sometimes drop and causes lag during the runtime. The lag is mostly obvious during the spatial mapping process. It might be caused by the number of geometries that the streaming server needs to send to the handheld device.

The movement of the user is also restricted due to being connected to the PC. This limitation could be improved by substituting the PC with an embedded computing device. Stereolabs claimed that the Nvidia Jetson Nano embedded computer is compatible with the ZED software development kit (SDK) and able to calculate the depth with the depth sensors, albeit with less performance [28] Thus, this is an interesting research direction to improve this framework.

5. CONCLUSION

Our contribution from this research is the integration of the handheld device with RGBD depth sensor that is able to perform spatial mapping. Spatial mapping generates detailed and accurate representation of the physical environment in the digital space via three-dimensional (3D) meshes. This mesh can used for occlusion handling through specific shaders that hides the meshes' rendering while still keeping the depth information. Since spatial mapping involves understanding of the environment depth, spatial mapping can be called depth-based mapping. RGBD depth sensors are able to perform spatial mapping with high precision, however a high-end PC is needed to perform the depth calculation process. Thus, this research contributed to leveraging the depth sensing power of the depth sensor on handheld device. Network-based streaming method was used to integrate the handheld device with the depth sensor. Thus, the handheld device is now able to perform the spatial mapping.

The integration of the depth sensor also contributed to the design of an improved framework for handheld AR occlusion handling. The framework's design was explained extensively in the methodology to show its underlying principles and operational mechanisms. The practicality of the framework was tested in the pilot test.

Overall, the integration of a handheld device with ZED Mini depth sensor offers promising potential for improving occlusion handling and reducing flickering in AR applications. The stereo vision technology in the ZED Mini allows near accurate depth sensing. Handheld devices usually have limited processing power compared to a desktop PC, which can be observed by the ARDK results. By integrating the ZED Mini that runs on the PC server, computational constraints can be lifted and allow improved handheld AR spatial mapping and occlusion handling. This in turn allows developers to create more immersive and realistic AR experiences that seamlessly blend virtual and real-world elements.

Further experimentation and optimization of this integration could lead to significant advancements in handheld AR technology. Implementing the ZED Mini depth sensor in handheld AR applications presents several challenges. Streaming the frame data from the ZED Mini to the handheld device in real-time requires robust data transfer protocols and efficient processing capabilities to ensure low latency and high throughput. We have observed several times whereby the frame data have delay in transmission, therefore making the streaming lag. The framerate also dropped noticeably during the spatial mapping process. We would propose in the future to explore compression methods for improving the latency issue, as done by [29]. Compressing the data transmission could also optimize power consumption to prolong battery life of the handheld device.

Evaluating handheld AR experiences also require real user use cases. Occlusion handling plays a significant role in affecting the user's depth perception and is primarily a visual phenomenon, thus evaluating it relies on conducting a user evaluation [30]. Thus, in the future, we would conduct user evaluation testing to gather user satisfaction levels on the usage of this handheld AR framework.

ACKNOWLEDGMENT

We would like to thank ViCubeLab at Universiti Teknologi Malaysia for their invaluable support and facilities.

REFERENCES

[1] F. Zhou, H. B.-L. Duh, and M. Billinghurst, "Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR," in *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, 2008, pp. 193–202. doi: 10.1109/ISMAR.2008.4637362.

[2] M. C. de F. Macedo and A. L. Apolinario, "Occlusion Handling in Augmented Reality: Past, Present and Future," *IEEE Trans Vis Comput Graph*, p. 1, 2021, doi: 10.1109/TVCG.2021.3117866.

[3] D. E. Breen, R. T. Whitaker, E. Rose, and M. Tuceryan, "Interactive Occlusion and Automatic Object Placement for Augmented Reality," *Computer Graphics Forum*, vol. 15, no. 3, pp. 11–22, 1996, doi: https://doi.org/10.1111/1467-8659.1530011.

[4] Y. Tian, T. Guan, and C. Wang, "Real-Time Occlusion Handling in Augmented Reality Based on an Object Tracking Approach," *Sensors*, vol. 10, no. 4, pp. 2885–2900, 2010, doi: 10.3390/s100402885.

[5] R. Benenson, "Occlusion Handling," in *Computer Vision: A Reference Guide*, K. Ikeuchi, Ed., Boston, MA: Springer US, 2014, pp. 551–552. doi: 10.1007/978-0-387-31439-6_136.

[6] M. M. Shah, H. Arshad, and R. Sulaiman, "Occlusion in augmented reality," in *2012 8th International Conference on Information Science and Digital Content Technology (ICIDT2012)*, 2012, pp. 372–378.

[7] N. A. B. Abdul Halim and A. W. B. Ismail, "Raycasting method using hand gesture for target selection on the occluded object in handheld Augmented Reality," in *2021 6th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, 2021, pp. 1–6. doi: 10.1109/ICRAIE52900.2021.9704035.

[8] N. M. and I. A. W. Ahmad Muhammad Anwar and Suaib, "Review of Model-Based Techniques in Augmented Reality Occlusion Handling," in *Expert Clouds and Applications*, S. and I. I. Jeena Jacob I. and Kolandapalayam Shanmugam, Ed., Singapore: Springer Nature Singapore, 2023, pp. 629–641.

[9] N. A. A. Halim and A. W. Ismail, “Target selection in handheld Augmented Reality for distant and occluded object,” in *2022 IEEE 7th International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, 2022, pp. 266–271. doi: 10.1109/ICRAIE56454.2022.10054325.

[10] J. Zhang *et al.*, “MobiDepth: Real-Time Depth Estimation Using on-Device Dual Cameras,” in *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, in MobiCom ’22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 528–541. doi: 10.1145/3495243.3560517.

[11] Y. Tian, Y. Ma, S. Quan, and Y. Xu, “Occlusion and Collision Aware Smartphone AR Using Time-of-Flight Camera,” in *Advances in Visual Computing*, G. Bebis, R. Boyle, B. Parvin, D. Koracin, D. Ushizima, S. Chai, S. Sueda, X. Lin, A. Lu, D. Thalmann, C. Wang, and P. Xu, Eds., Cham: Springer International Publishing, 2019, pp. 141–153.

[12] J. Valentin *et al.*, “Depth from Motion for Smartphone AR,” *ACM Trans. Graph.*, vol. 37, no. 6, Dec. 2018, doi: 10.1145/3272127.3275041.

[13] “Depth Adds Realism | ARCore | Google Developers.” Accessed: Apr. 16, 2023. [Online]. Available: https://developers.google.com/ar/develop/depth

[14] R. Du *et al.*, “DepthLab: Real-Time 3D Interaction with Depth Maps for Mobile Augmented Reality,” in *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, in UIST ’20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 829–843. doi: 10.1145/3379337.3415881.

[15] L. Burger *et al.*, “Comparative evaluation of three commercially available markerless depth sensors for close-range use in surgical simulation,” *Int J Comput Assist Radiol Surg*, vol. 18, no. 6, pp. 1109–1118, 2023, doi: 10.1007/s11548-023-02887-1.

[16] D. Wolf, T. Wagner, and E. Rukzio, “Low-Cost Real-Time Mental Load Adaptation for Augmented Reality Instructions - A Feasibility Study,” in *2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, 2019, pp. 1–3. doi: 10.1109/ISMAR-Adjunct.2019.00015.

[17] M. Martini, F. Solari, and M. Chessa, “Obstacle Avoidance and Interaction in Extended Reality: An Approach Based on 3D Object Detection,” in *Image Analysis and Processing – ICIAP 2023*, A. and H. E. Foresti Gian Luca and Fusiello, Ed., Cham: Springer Nature Switzerland, 2023, pp. 111–122.

[18] J. Ferrão, P. Dias, B. S. Santos, and M. Oliveira, “Environment-Aware Rendering and Interaction in Web-Based Augmented Reality,” *J Imaging*, vol. 9, no. 3, 2023, doi: 10.3390/jimaging9030063.

[19] C. Zhou, Q. Yan, Y. Shi, and L. Sun, “DoubleStar: Long-Range Attack Towards Depth Estimation Based Obstacle Avoidance in Autonomous Systems,” 2021, doi: 10.48550/arxiv.2110.03154.

[20] B. Glocker, J. Shotton, and A. Criminisi, “Real-Time RGB-D Camera Relocalization via Randomized Ferns for Keyframe Encoding,” *Visualization and Computer Graphics, IEEE Transactions on*, vol. 21, pp. 571–583, Oct. 2015, doi: 10.1109/TVCG.2014.2360403.

[21] T. Jin, S. Wu, M. Dasari, K. Apicharttrisorn, and A. Rowe, “StageAR: Markerless Mobile Phone Localization for AR in Live Events,” in *2024 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*, 2024, pp. 1000–1010. doi: 10.1109/VR58804.2024.00119.

[22] Niantic, “Lightship ARDK.” Accessed: Apr. 15, 2023. [Online]. Available: https://lightship.dev/

[23] J. Watson, O. Mac Aodha, V. Prisacariu, G. J. Brostow, and M. Firman, “The Temporal Opportunist: Self-Supervised Multi-Frame Monocular Depth,” *CoRR*, vol. abs/2104.14540, 2021, [Online]. Available: https://arxiv.org/abs/2104.14540

[24] C. Jennings, T. Hardie, and M. Westerlund, “Real-time communications for the web,” *IEEE Communications Magazine*, vol. 51, no. 4, pp. 20–26, 2013, doi: 10.1109/MCOM.2013.6495756.

[25] J. K. Nurminen, A. J. R. Meyn, E. Jalonen, Y. Raivio, and R. Garcıa Marrero, “P2P media streaming with HTML5 and WebRTC,” in *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2013, pp. 63–64. doi: 10.1109/INFCOMW.2013.6970739.

[26] Unity Technologies, “About Unity Render Streaming”, Accessed: Nov. 23, 2023. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.renderstreaming@3.1/manual/index.html

[27] A. Ismail, M. Aladin, and M. N. A. Nor’a, “Real Hand Gesture in Augmented Reality Drawing with Markerless Tracking on Mobile,” *International Journal of Computing and Digital Systems*, vol. 12, pp. 1071–1080, Oct. 2022, doi: 10.12785/ijcds/120186.

[28] Stereolabs, “Announcing ZED SDK for Jetson Nano.” Accessed: Feb. 19, 2024. [Online]. Available: https://www.stereolabs.com/blog/announcing-zed-sdk-for-jetson-nano

[29] A. Wanis Ismail, S. Abd Karim Ishigaki, and F. Edora Fadzli, “Improved Real-time 3D Reconstruction Method for Mixed Reality Telepresence,” *International Journal of Computing and Digital Systems*, vol. 20, pp. 2210–142, doi: 10.12785/ijcds/XXXXXX.

[30] M. Alfakhori, J. S. Sardi Barzallo, and V. Coors, “Occlusion Handling for Mobile AR Applications in Indoor and Outdoor Scenarios,” *Sensors*, vol. 23, no. 9, 2023, doi: 10.3390/s23094245.

Muhammad Anwar Ahmad is currently a PhD student at Universiti Teknologi Malaysia, Skudai Johore. He is a member of UTM Vicubelab – a research group of experts in virtual, visualization, and vision. He graduated from UTM with Bachelor’s degree in 2016. He received his MPhil from UTM in 2019, with research focus on 3D facial expression of local cultural dance. For his PhD, his research focus is towards Mixed/Augmented Reality technology. He is also a member of mivielab (mixed and virtual environment research lab).

Norhaida Mohd Suaib is currently a Senior Lecturer in computer graphics and computer vision at the Faculty of Computing, Universiti Teknologi Malaysia, Skudai, Johore. She is a member of UTM Vicubelab – a research group of experts in virtual, visualization, and vision. Her expertise is in Mixed/Augmented Reality technology, computer graphics algorithm & techniques, interactive computer graphics, visualization and computer graphics and cultural heritage.

Ajune Wanis Ismail is a senior lecturer at Johor Universiti Teknologi Malaysia (UTM). She is currently the head of UTM Vicubelab – a research group of experts in virtual, visualization, and vision. She earned her B.Sc., M.Sc., and Dr. Sc. degrees from UTM. She earned a B.Sc. in computer graphics and computer vision and began research on Augmented Reality, which is now her primary research focus. Her expertise is in Mixed/Augmented Reality technology. She is also the head of mivieLab (mixed and virtual environment research lab).