Sharing Ambient Objects Using Real-time Point Cloud Streaming in Web-based XR Remote Collaboration

Yongjae Lee
Korea Institute of Science and Technology
Seoul, Korea
Department of Mechanical Engineering
Yonsei University
Seoul, Korea
yongjae.lee@wrl.onl

Byounghyun Yoo*
Korea Institute of Science and Technology
Seoul, Korea
Division of Nano & Information Technology, KIST School
Korea University of Science and Technology
Seoul, Korea
yoo@byoo.net

Soo-Hong Lee*
Department of Mechanical Engineering
Yonsei University
Seoul, Korea
shlee@yonsei.ac.kr

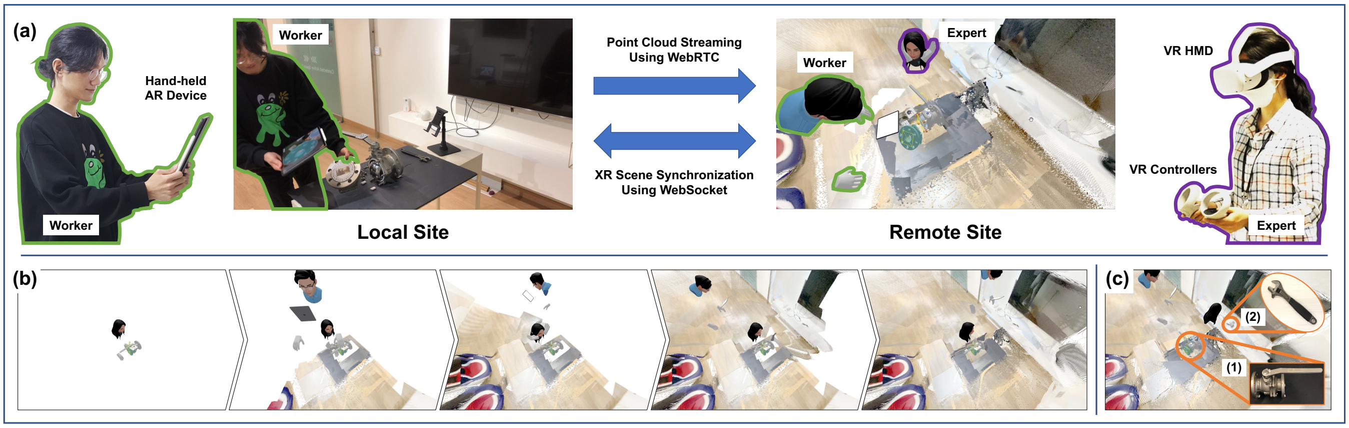


Figure 1: Overview of the proposed XR collaboration method: (a) The worker at the local site shows his workspace by replicating the space as point cloud data and streaming them by WebRTC, and shares the pose information of the ball valve (1), which the worker and expert are collaborating on via WebSocket. The expert at the remote site observes the worker's workspace through the point cloud and understands the status of the ball valve through its pose-synchronized 3D model. (b) The virtual space where the expert resides is being filled with the point cloud from the worker. (c) There are two types of virtual objects in the virtual space. The ball valve (1) is 3D modeled in advance, and the spanner (2) is reconstructed immediately by the worker.

ABSTRACT

Extended reality (XR) collaboration enables collaboration between physical and virtual spaces. Recent XR collaboration studies have focused on sharing and understanding the overall situation of the objects of interest (OOIs) and its surrounding ambient objects (AOs) rather than simply recognizing the existence of OOI. The sharing of the overall situation is achieved using three-dimensional (3D) models that replicate objects existing in the physical workspace. There are two approaches for creating the models: pre-reconstruction and real-time reconstruction. The pre-reconstruction approach takes considerable time to create polygon meshes precisely, and the real-time reconstruction approach requires a considerable time to install numerous sensors to perform accurate 3D scanning. In addition, these approaches are difficult to be used on the collaboration in a location beyond the reconstructed space, making them impractical to an actual XR collaboration. The approach proposed in this study separates the objects that form the physical workspace into OOI and AO, models only the OOI as a polygon mesh in advance, and reconstructs the AO into a point cloud using light detection and ranging technology for collaboration. The reconstructed point cloud is shared with remote collaborators through WebRTC, a web-based peer-to-peer networking technology with low latency. Each remote collaborator collects the delivered point cloud to form a virtual space, so that they can intuitively understand the situation at a local site. Because our approach does not create polygon meshes for all objects existing at the local site, we can save time to prepare

*Corresponding authors: Byounghyun Yoo (yoo@byoo.net) and Soo-Hong Lee (shlee@yonsei.ac.kr).

Web3D '21, November 8–12, 2021, Pisa, Italy
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9095-8/21/11.
https://doi.org/10.1145/3485444.3487642

for collaboration. In addition, we can improve the practicality of XR collaboration by eliminating the need to install numerous sensors at the local site. We introduce a prototype and an example scenario to demonstrate the practicality of our approach.

CCS CONCEPTS

• **Human-centered computing** → *Collaborative and social computing systems and tools*; **Mixed / augmented reality**; **Virtual reality**; **Web-based interaction**.

KEYWORDS

Extended reality, XR, Virtual reality, Augmented reality, Web-based XR, Remote XR collaboration, Real-time point cloud streaming

ACM Reference Format:
Yongjae Lee, Byounghyun Yoo, and Soo-Hong Lee. 2021. Sharing Ambient Objects Using Real-time Point Cloud Streaming in Web-based XR Remote Collaboration. In *The 26th International Conference on 3D Web Technology (Web3D '21), November 8–12, 2021, Pisa, Italy.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3485444.3487642

1 INTRODUCTION

With advances in hardware and vision recognition technology, virtual reality (VR) and augmented reality (AR) have permeated many aspects of our lives. Prospective buyers can tour houses using VR without physically visiting the house [TopVIEW 2021]. Students can effectively learn about the elements of nature through AR and solve quizzes augmented on a see-through screen instead of reading a textbook [Alakärppä et al. 2017]. VR and AR are collectively referred to as extended reality (XR), and XR systems can eventually allow users to choose and alternate between VR and AR according to their preferences. Moreover, XR technology has gained significant attention in the context of the web community, considering that an XR specification for the Web, called WebXR, is being established in the World Wide Web Consortium.

XR technology is also active in remote collaboration because it allows intuitive information delivery by displaying virtual objects instead of delivering information abstractly through speech or writing. Early XR collaboration studies focused only on conveying the information necessary for collaboration. Using their methods, collaboration is conducted by marking parts or attaching annotations of work information on a screen where the local worker needs to concentrate [Gauglitz et al. 2014; Nuernberger et al. 2016]. Recent research has focused on quickly and effectively communicating workers' intentions by sharing various communication cues, such as their gazes, hand gestures, and heart rates [Bai et al. 2020; Dey et al. 2017; Piumsomboon et al. 2017].

Meanwhile, research on improving the spatial awareness of remote workers is being actively conducted [Gao et al. 2020, 2017; Irlitti et al. 2019]. Improved spatial awareness allows a remote worker to easily identify their location relative to another worker or an object of interest (OOI). Two-dimensional (2D) images (including videos) or three-dimensional (3D) models are used as a medium for delivering spatial awareness cues. Using a 2D image has the advantage of allowing the remote worker to view a realistic representation of the local worker's workspace. However, there is no depth in the image, and the remote worker is unable to change the viewpoint of the local worker's camera at will. When using a 3D model, the remote worker explores a virtual space consisting of 3D models that replicate objects existing at the local site. The remote worker can freely navigate the virtual space, changing their viewpoint independently of the local worker camera. However, a reconstructed 3D model is less realistic than a 2D image. Thus, recent studies have conducted photo-realistic 3D rendering using several methods, such as neural nets [Aliev et al. 2020; Dai et al. 2020].

To generate 3D models of objects existing at a local site, devices with embedded optical depth sensor systems, such as Microsoft HoloLens and Microsoft Kinect, are widely used. Using such hardware, the local worker scans the local site before starting the collaboration, to obtain depth maps at various angles. The acquired depth maps are then combined to reconstruct the geometric information of the local site into a single point cloud or polygon mesh [Piumsomboon et al. 2017]. The reconstructed 3D model (point cloud or polygon mesh) is delivered to remote workers and used as a cue to help them gain spatial awareness of the local site in VR. In addition, a method has been studied to reconstruct the 3D model in real time by installing multiple sensors on the local site without scanning around the site [Lindlbauer and Wilson 2018]. Both pre- and real-time 3D model reconstructions in preparation for collaboration (e.g., scanning the local site and installing sensors) are time consuming. Therefore, it is difficult to initiate a collaboration promptly when no preparation has yet been made. Moreover, it is difficult to collaborate outside of the replicated space, even if a 3D model of the local site has been prepared.

We previously studied a system that captures and broadcasts the geometry of a local site as a point cloud in real time [Seo et al. 2017], a methodology for webizing various interaction devices [Seo et al. 2018], an XR collaboration system with distributed web technologies [Huh et al. 2019], a web-based XR content representation method and its rendering algorithm [Lee et al. 2020], and a unified coordinate system for VR and AR [Lee and Yoo 2021]. This study extends the aforementioned concepts. Our method is a hybrid approach combining the conventional pre-reconstruction approach for OOIs and the real-time reconstruction approach for ambient objects (AOs). The reconstructed 3D model for OOIs is a polygon mesh and is used to represent the state of the OOI (e.g., pose information). The reconstructed 3D model for AOs is a point cloud and is utilized as a background for VR, helping a remote worker acquire spatial awareness of the local site. Through this hybrid approach, the preparation time for a collaboration is effectively reduced because only the OOI, as opposed to the entire local site, requires pre-reconstruction. Furthermore, workers can reuse the content in another location by simply scanning again for spatial awareness of that location. This reuse enables flexible content design and efficient XR collaboration.

Contributions of this paper can be summarized:

(1) An XR collaboration system that uses a hybrid approach in sharing a local worker's overall situation
(2) A method for removing points representing an OOI from a reconstructed point cloud
(3) A packet structure that can be used immediately upon receipt in point cloud data streaming with WebRTC

2 RELATED WORK

2.1 XR collaboration

XR collaboration implies that a group of people work together to address a common purpose through VR and AR interfaces. In XR collaboration, the OOI can be either a virtual object or a physical object. If the OOI is a virtual object, the collaboration is conducted only on the virtual object [Grandi et al. 2019; Poretski et al. 2018]. The physical environment on the AR side serves only as the background for placing virtual objects. If the OOI is a physical object, virtual objects are used as a means to convey information about physical tasks [Fakourfar et al. 2016; Kim et al. 2019; Seo et al. 2016; Wang et al. 2020]. A local worker using AR learns the tasks they need to perform by looking at the virtual objects (such as gestures, drawings, and animations) augmented near the OOI. In existing XR collaboration studies, local and remote workers share information about and interact only with the OOI. They do not utilize the environment outside the OOI.

2.2 Sharing physical environment

Previous studies have shared information about the situation of a local site and utilized them in collaboration. The SharedSphere system uses 360° live video streaming to share the situation [Lee et al. 2017]. In the SharedSphere, the local worker shares information about both the OOI and AOs via live video. Other studies have captured a 3D snapshot of the objects that form the local site as a point cloud and share it with the remote worker in VR [Gao et al. 2018, 2017]. In the 360° live video method, the remote worker can understand the situation only from the viewpoint of the local worker's camera. Conversely, using the point cloud method, the remote worker can freely navigate and inspect the local site from any location. However, the point cloud does not reflect dynamic changes occurring during the collaboration because it is a static scene that replicates the situation at the time the local site was scanned. Collaboration systems that use both 360° live videos and point clouds to compensate for the shortcomings of both methods have been proposed [Gao et al. 2020; Teo et al. 2019, 2020]. In these systems, the 3D reconstruction process of the local site and the delivery process of the reconstructed 3D model to the remote worker occur in turn. It is difficult to obtain live feedback from the remote worker in the 3D reconstruction process, which results in inefficiencies, such as scanning of unnecessary objects or dropping details of the required parts.

3 METHODOLOGY

Our primary idea is to separate the objects that form the workspace of the local site into OOI and AO. To represent the OOI, a 3D modeled polygon mesh was used. A remote worker can intuitively teach a local worker about task instructions by demonstrating them with the polygon model. The XR collaboration process using the polygon model of the OOI follows the method presented in our previous work [Lee et al. 2020; Lee and Yoo 2021]. For the AO, the local worker generates a point cloud replicating only AO, and then delivers it to the remote worker via peer-to-peer networking. The remote worker receives the point cloud and uses it as an immersive VR background.

3.1 XR collaboration

In the XR collaboration method of Lee and Yoo [2021], a local worker performing a physical task participates in the collaboration through an AR interface. In their method, it was not intended for a local worker to use a VR interface because it would completely block information from the local worker's surrounding environment, making it difficult for the local worker to interact with physical objects. Once the local worker recognizes the OOI through the AR device, various information (e.g., annotations) aiding the local worker's task is augmented around the OOI. Feature information is required for recognizing the OOI, and it is recorded in an XR content with the annotations. The most commonly used annotation type for OOI is polygon mesh. Recognizing the OOI can be accomplished by various tracking systems (e.g., ARKit [Apple Inc. 2017] or Wikitude [Wikitude GmbH 2008]), which detect the OOI and calculate the OOI's pose (position and orientation) using the feature information. After attaining the pose, the AR device passes them over the network to a remote worker who collaborates through a VR interface. The remote worker also observes the polygon mesh of the OOI in VR. This mesh is synchronized with the delivered pose information, which helps the remote worker identify the pose of the OOI existing at the local site. When the remote worker manipulates and moves the polygon mesh on the VR side, the polygon mesh augmented on the local worker's screen reflects the manipulation. Consequently, the polygon mesh of the OOI is used for the remote worker to identify the OOI's pose and for the local worker to be told how to work on the OOI. Using the proposed XR collaboration method, the remote worker can intuitively describe how to perform the physical task, such as disassembly or assembly, by demonstrating the task instruction on the polygon mesh.

3.2 Sharing ambient objects

The XR collaboration method proposed by Lee and Yoo [2021] does not consider AOs at the local site. However, sharing information about AOs can make collaboration more efficient by allowing a remote worker to grasp the context of the local site quickly. Sharing AO information requires three steps. The first step involves the 3D reconstruction of the geometry of the AOs at the local site into the point cloud (Figure 2(a)). The second step involves masking the part corresponding to the OOI in the generated point cloud (Figure 2(b)). The final step is to deliver the masked point cloud to a remote worker (Figure 2(c)). This procedure is repeated for each frame of depth data gathered from light detection and ranging (LiDAR).

3.2.1 Point cloud reconstruction. A LiDAR measures the distance to an object by measuring the return time of a laser pulse. If the surface of the hit object is very smooth (e.g., polished metal or mirror) and total reflection occurs, the laser pulse may not return [Gao et al. 2021]. The measured distance information is obtained in the form of a 2D depth map, using which we can estimate the geometry of the local site. In the depth map, the intensity of each pixel refers to the distance z_c from the LiDAR to the object (subscript c implies that its value is defined in the camera coordinate system). Using the pinhole camera model, a point (x_c, y_c, z_c) in the real world can be obtained from a pixel point (u, v) and distance z_c in the depth map [Szeliski 2010].

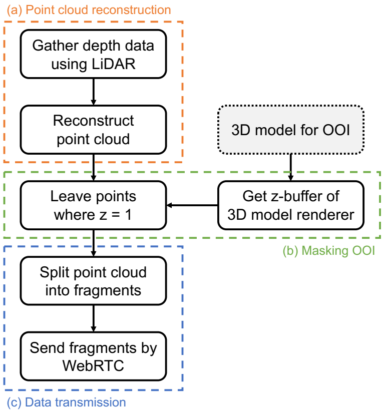

(a) Point cloud reconstruction
Gather depth data using LiDAR
Reconstruct point cloud
3D model for OOI
Leave points where z = 1
Get z-buffer of 3D model renderer
(b) Masking OOI
Split point cloud into fragments
Send fragments by WebRTC
(c) Data transmission

Figure 2: Flowchart of sharing information about ambient objects existing at the local site

3.2.2 Masking OOI. The point cloud represents the geometry of the local site precisely. In general, a point cloud is sufficient to represent the AOs of the local site because they have static or limited movement. Conversely, because the OOI is frequently moved during a collaboration, capturing an OOI as a point cloud disturbs the virtual space that a remote worker is experiencing, making it difficult for the remote worker to understand the situation of the local site.

Figure 3 chronologically shows the result of moving an OOI while reconstructing the point cloud. In the figure, the tablet model and valve flange model (the cyan colored virtual object in the Figure 3(a1)) in front of the male avatar are synchronized with the AR device (held by the local worker) and the real valve flange (above the workbench), respectively (Figure 4(a) depicts the local site configuration corresponding to Figure 3). The OOI for collaboration is the valve flange, and the AO to be reconstructed is the workbench. Because the valve flange is located on the workbench, a point cloud for the valve flange is also generated when the local worker scans the workbench with the LiDAR which is embedded in the AR device. At this point, as the local worker moves the valve flange to the left, afterimages of the flange remain along the path of movement (Figure 3(a)). To prevent these afterimages, the reconstructed point cloud is masked to remove the points in the OOI's area. Consequently, the masked area in the point cloud is left as a blank, and the blank area is substituted by the flange model (Figure 3(b2)). In Figure 3(b3) and (b4), the local worker moves the flange to the left, but no additional afterimages occur as the points in the area of the flange are masked. After moving the flange, the hole once occupied by the flange model is filled with newly generated points.

Because an OOI and its 3D model are known through XR content, which points should be masked from a point cloud can be identified. When comparing the rendered results of the point cloud (Figure 4(b)) and the 3D model (Figure 4(c)), the pixel area representing the OOI (valve flange) is the same for each rendered result. It means that when the points of the point cloud are projected onto the screen, the point is a point to be masked if the pixel position of the projected point is within the area of the rendered 3D model. Whether a pixel is within the area of the rendered 3D model can be determined using the z-buffer of the 3D model renderer. If the z-value of a pixel is not 1 (typically in the OpenGL [The Khronos Group Inc. 2019], the range of a z-value is $[0.0, 1.0]$ and the default depth clearing value is 1.0), it is within the area of the rendered the 3D model. When the z-value equals 1, it indicates that no 3D object occludes that pixel. Finally, the group of remaining points after being masked is defined as follow:

$$Point = \{p | Z_p = 1\} \tag{1}$$

where *Point* denotes the group of remaining points after being masked, p denotes a point in a point cloud, Z denotes the z-buffer of a 3D model renderer, and Z_p denotes the z-value corresponding to p in the z-buffer.

We provided an example code of this section in appendix A.

3.2.3 Data transmission. During the transmission step, the masked point cloud is delivered to other workers participating in the collaboration through a WebRTC data channel. The maximum buffer size of the data channel is 16 KiB (16 384 B)[1]. However, this is generally a tiny size to hold all points of a point cloud. Thus, a point cloud must be split to fit the buffer size.

The position data consist of three 32-bit float values, and the color data consist of three 8-bit unsigned int values. Thus, the size of a single point is 15 B. Consequently, a fragment of a point cloud that fits in the data channel buffer can contain up to $\lfloor 16384/15 \rfloor = 1092$ points. With this fact, we can derive the number of fragments generated from a point cloud. For example, a depth map with a resolution of $192 \times 256 = 49152$ requires $\lceil 49152/1092 \rceil = 46$ fragments to be delivered through the data channel.

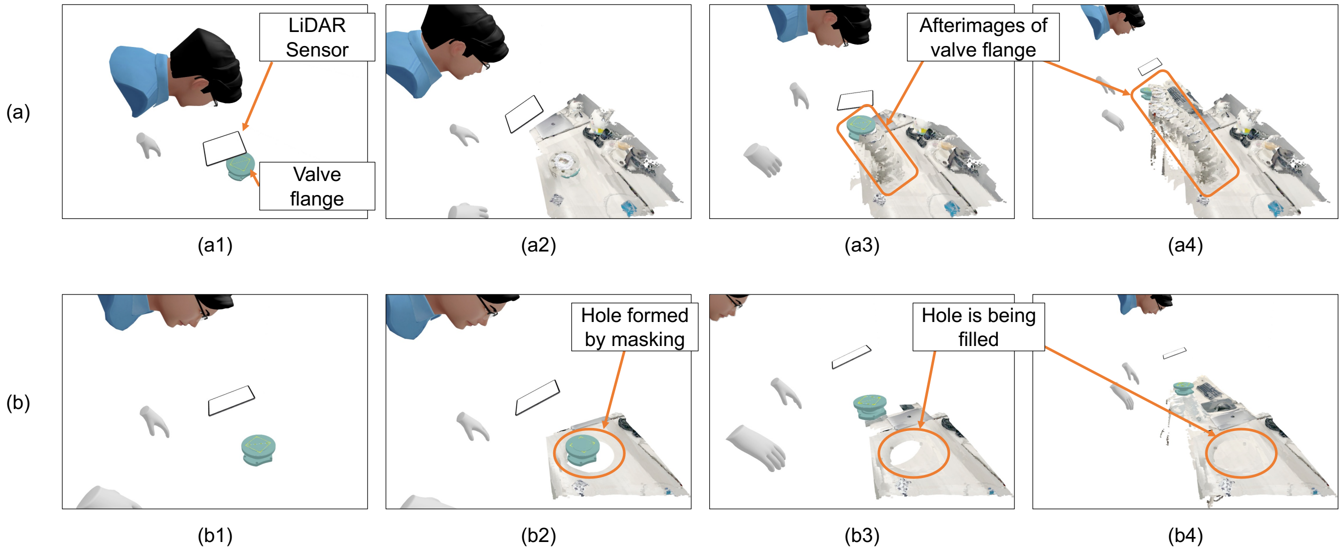
In general, point cloud data are managed separately as position data and color data for efficient memory management. When the point cloud data are divided according to the buffer size of the data channel in the order stored in the memory, the position data and color data for one point are separated into different packets. Rendering a single point requires both position and color data. In other words, packets containing position data and packets containing color data are dependent on each other. Packets can be delivered out of order as they pass through the network (Figure 5(a)). Therefore, a remote worker may receive other packets between two dependent packets. Because both dependent packets must be available, the point cannot be rendered until the receipt of the second packet.

Figure 5(b) presents a structure dividing the point cloud data such that there is no dependence between packets. The position and color data that comprise one point are contained in the same packet. Consequently, the relevant portion of the point cloud can be rendered as soon as each packet arrives.

4 IMPLEMENTATION

We implemented a prototype to demonstrate the proposed XR collaboration concept. Figure 6 shows the architecture of the prototype system. The WXR Library is a JavaScript library that controls and renders XR content. The WXR Library was improved based on A-Frame [Marcos et al. 2020] from a version introduced in our previous study [Lee and Yoo 2021]. The XR content loader downloads static resources, such as XR scenes or textures, from the WXR

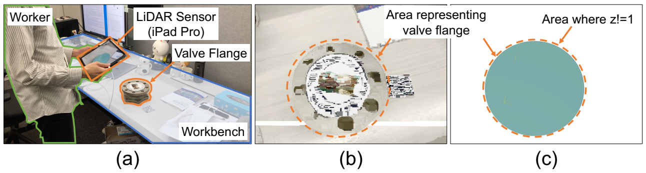
[1] MDN. https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Using_data_channels#concerns_with_large_messages, last accessed on 07/30/21



(a1) (a2) (a3) (a4)

(b1) (b2) (b3) (b4)

Figure 3: Reconstruction result comparison (a) without masking and (b) with masking. The layout of subfigures is chronological from left to right.

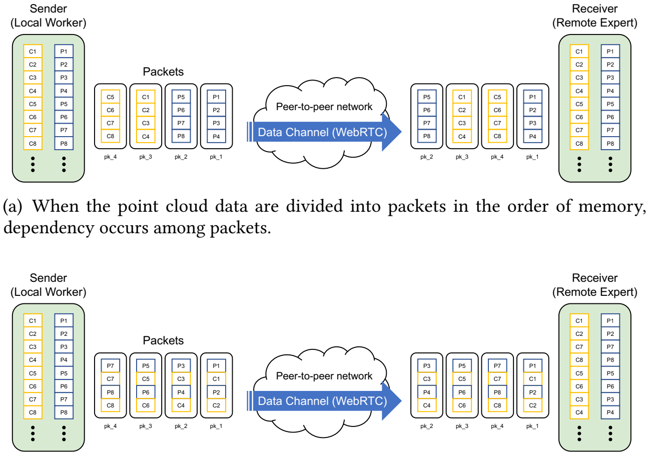


(a) (b) (c)

Figure 4: (a) Configuration of the local site, (b) rendered result of the point cloud, and (c) rendered result of 3D model of the ball valve flange. Note that the holes on the flange were blocked in the 3D model for a better masking result.

workspace server. The event handler applies interaction events (represented as Event Data in Figure 6) to the XR scene and transports them from and to the server for synchronization. The point cloud receiver adds the point cloud delivered through WebRTC to the XR scene. Because the data size of a point cloud is generally big, controlling them loads the server. Thus, we opted to use WebRTC for streaming point cloud data. On the contrary, we chose WebSocket for sending Event Data because the data size of an Event Data is generally tiny, and the logic for handling some Event Data (e.g., joining in or exiting from a collaboration session) is executed in the server.

We created a browser application (WXR Browser) to use the object tracking function and LiDAR functionality in the tablet. The AR engine used in the prototype is ARKit, which provides pose information of the OOI obtained from the object tracking function and a depth map collected from the LiDAR embedded in the tablet. The pose information of the OOI was applied to the 3D model of the OOI in the XR scene via the event handler. The point cloud generator reconstructed a point cloud using depth data from the AR engine and masked the points in the corresponding area of the OOI. Other than that the WXR library rendered the XR scene on the user's screen, the point cloud generator also performed off-screen rendering only on the OOI to obtain its z-buffer. In the point cloud streamer, the masked point cloud was divided by the buffer size of the data channel and passed to other users connected by WebRTC. The process of point-cloud reconstruction could be switched on and off via a separate user interface.



(a) When the point cloud data are divided into packets in the order of memory, dependency occurs among packets.

(b) Dependencies among packets can be eliminated by placing both the position and color data of a point in the same packet.

Figure 5: Dependency among packets. 'P1' denotes position data of the first point in the point cloud. 'C1' denotes color data of the first point. 'pk_n' denotes the *n* th sent packet containing position or color data.

Figure 1 presents the configuration of the collaboration. The local worker with the AR device reconstructs the point cloud of

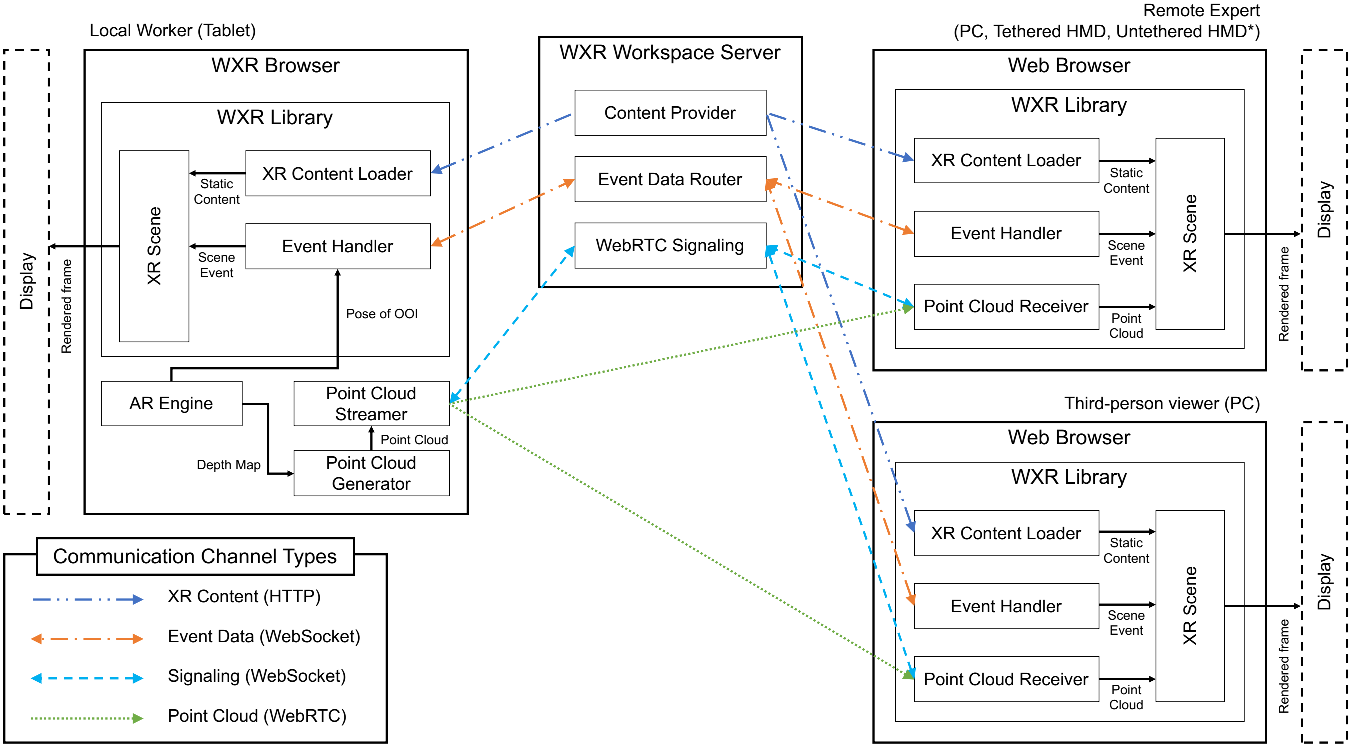
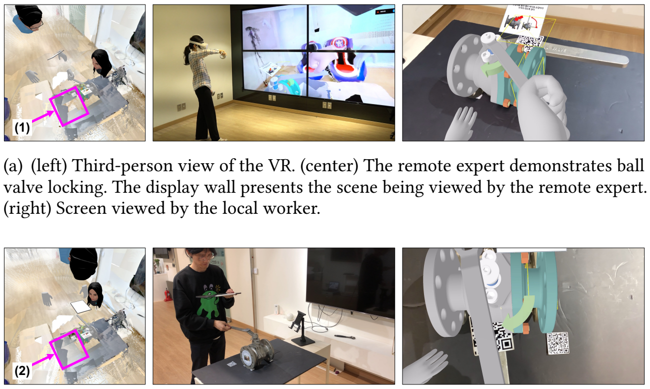


Figure 6: System architecture
*** An untethered HMD, such as the Oculus Quest 2, can be used as a terminal computer. However, in our prototype implementation, we used the Oculus Quest 2 by tethering to a PC for high rendering performance. By conducting several tests, we found that the Oculus Quest 2 must be tethered to render over 100 thousand points of a point cloud.**



(a) (left) Third-person view of the VR. (center) The remote expert demonstrates ball valve locking. The display wall presents the scene being viewed by the remote expert. (right) Screen viewed by the local worker.

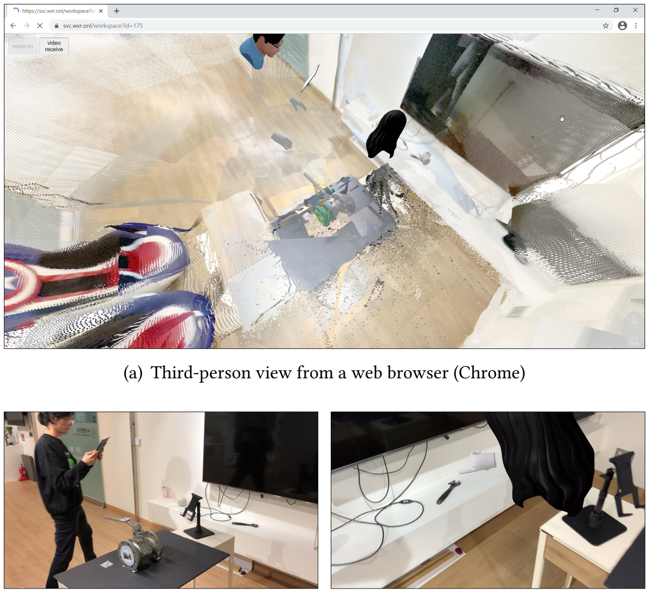
(b) (left) Third-person view of the VR. (center) Local worker performs valve locking. (right) Screen viewed by the local worker.

Figure 7: (a) Work demonstration of remote expert, and (b) the local worker following the remote expert

the AO around him and recognizes the ball valve (Figure 1(c1)), which is the OOI. The remote expert on the VR side can understand the situation of the local site by looking at the 3D model of the OOI and the point cloud of the AO, and deliver task instructions to the local worker. The collaboration process is observed by a third person. The AR device used by the local worker is iPad Pro 12.9-inch (4th generation), and the VR device used by the remote expert is Oculus Quest 2. The Oculus Quest 2 was tethered via Firefox browser on a PC (AMD Ryzen 9 3900X 12-Core 3.8 GHz CPU, 64 GB DDR4 RAM, NVIDIA Quadro RTX 5000 GPU) with a Windows 10 operating system using Oculus Air Link[2]. The third person used Chrome Browser on a laptop (Intel Core i9 8950HK 2.9 GHz CPU, 32 GB DDR4 RAM, Intel UHD Graphics 630 GPU) configured with a Windows 10 operating system. The server program was written using Node.js with MariaDB and run on a PC (Intel Xeon E5-2699 v4 2.2 GHz CPU, 64 GB DDR4 RAM) with a Windows 10 operating system.

In the reconstruction process of the point cloud of the AO, the points collected around the ball valve were masked, and the space of the masked points was occupied by the 3D model of the ball valve. The remote expert accumulated the point cloud delivered at each frame, as shown in Figure 1(b). The accumulated point cloud formed the background of the virtual space that the remote expert was experiencing. Once the reconstruction was completed, the point cloud surrounded the 3D model of the ball valve, allowing the remote expert in VR to grasp the situation of the local site.

[2] Facebook. https://support.oculus.com/link/, last accessed on 07/30/21



(a) Third-person view from a web browser (Chrome)

(b) Local worker watching the remote expert's gesture through the AR device

(c) Screen of the AR device of the local worker

Figure 8: Identifying the location of the spanner through VR used by the remote expert.

The remote expert demonstrated a task in three steps: locking the ball valve, disassembling the flange, and pulling the ball out of the valve. At each step, the remote expert directly showed the local worker how to perform the task using the 3D model of the ball valve part (Figure 7(a)(center)). The 3D model was augmented on the local worker's AR device screen (Figure 7(a)(right)). The local worker could intuitively understand the remote expert's instruction by viewing the 3D model's movement, and perform the task by manipulating the handle based on his understanding of the task instructions (Figure 7(b)(center)). The AR device carried by the local worker recognized the movement of the handle using an object tracking function and synchronized the pose of the handle to the corresponding 3D model (Figure 7(b)(right)). When the local worker rotated the handle, the area (Figure 7(a)(1)), which was initially not reconstructed because of the masking, was exposed to the LiDAR and filled with a newly reconstructed point cloud (Figure 7(b)(2)). In the process of disassembling the flange, a spanner was required to disassemble a set of bolt nuts that combined the ball valve flanges. The remote expert was able to find the spanner in the background consisting of the reconstructed point cloud and inform the local worker of its location (Figure 8). A complete demonstration video of the prototype is available at the following link: https://bit.ly/3yyLR1Q.

5 CONCLUSION

In this paper, we introduced a new XR collaboration concept that separates OOIs and AOs and conveys the situation of a local site in an appropriate manner for each object. This collaboration method uses a LiDAR to reconstruct the point cloud on the fly only for the AOs. Through this, a local worker can manipulate OOIs while sharing their AOs. This method does not require installing a complex sensor system to communicate the situation at the local site. Moreover, this method makes it possible for XR collaboration to proceed in any space. Through the demonstration, we observed that it was possible for the remote expert to recognize AOs (e.g., the spanner) in the local site and induce the local worker to use it. In addition, the proposed method is practical because it does not create virtual replicas for all objects in the workspace, thus saving preparation time for collaboration.

The limitation of this research is that the proposed system is not fully qualified against quantitative user surveys due to the lasting COVID-19 pandemic. Thus, our future work involves conducting a user evaluation of the proposed method to quantitatively verify the practicality and efficiency of our XR collaboration concept against traditional collaboration methods, such as video conferencing.

ACKNOWLEDGMENTS

This work was supported by the Industrial Technology Innovation Program (20012462) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea), the National Research Foundation of Korea (NRF) grant (NRF-2021R1A2C2093065) funded by the Korea government (MSIT) and the KIST under the Institutional Program (Grant No. 2V09004).

REFERENCES

Ismo Alakärppä, Elisa Jaakkola, Jani Väyrynen, and Jonna Häkkilä. 2017. Using Nature Elements in Mobile AR for Education with Children. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '17)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3098279.3098547

Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. 2020. Neural Point-Based Graphics. In *Computer Vision — ECCV 2020*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer International Publishing, Cham, 696–712. https://doi.org/10.1007/978-3-030-58542-6_42

Apple Inc. 2017. ARKit. Retrieved July 30, 2021 from https://developer.apple.com/augmented-reality/

Huidong Bai, Prasanth Sasikumar, Jing Yang, and Mark Billinghurst. 2020. A User Study on Mixed Reality Remote Collaboration with Eye Gaze and Hand Gesture Sharing. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376550

Peng Dai, Yinda Zhang, Zhuwen Li, Shuaicheng Liu, and Bing Zeng. 2020. Neural Point Cloud Rendering via Multi-Plane Projection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 7830–7839. https://arxiv.org/abs/1912.04645

Arindam Dey, Thammathip Piumsomboon, Youngho Lee, and Mark Billinghurst. 2017. Effects of Sharing Physiological States of Players in a Collaborative Virtual Reality Gameplay. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. Association for Computing Machinery, New York, NY, USA, 4045–4056. https://doi.org/10.1145/3025453.3026028

Omid Fakourfar, Kevin Ta, Richard Tang, Scott Bateman, and Anthony Tang. 2016. Stabilized Annotations for Mobile Remote Assistance. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. Association for Computing Machinery, New York, NY, USA, 1548–1560. https://doi.org/10.1145/2858036.2858171

Lei Gao, Huidong Bai, Mark Billinghurst, and Robert W Lindeman. 2020. User Behaviour Analysis of Mixed Reality Remote Collaboration with a Hybrid View Interface. In *32nd Australian Conference on Human-Computer Interaction (OzCHI '20)*. Association for Computing Machinery, New York, NY, USA, 629–638. https://doi.org/10.1145/3441000.3441038

Lei Gao, Huidong Bai, Weiping He, Mark Billinghurst, and Robert W Lindeman. 2018. Real-Time Visual Representations for Mobile Mixed Reality Remote Collaboration. In *SIGGRAPH Asia 2018 Virtual & Augmented Reality (SA '18)*. Association for Computing Machinery, New York, New York, USA, 1–2. https://doi.org/10.1145/3275495.3275515

Lei Gao, Huidong Bai, Rob Lindeman, and Mark Billinghurst. 2017. Static Local Environment Capturing and Sharing for MR Remote Collaboration. In *SIGGRAPH Asia 2017*

Mobile Graphics & Interactive Applications (SA '17). Association for Computing Machinery, New York, New York, USA, 1–6. https://doi.org/10.1145/3132787.3139204

Rui Gao, Jisun Park, Xiaohang Hu, Seungjun Yang, and Kyungeun Cho. 2021. Reflective Noise Filtering of Large-Scale Point Cloud Using Multi-Position LiDAR Sensing Data. *Remote Sensing* 13, 16 (2021). https://doi.org/10.3390/rs13163058

Steffen Gauglitz, Benjamin Nuernberger, Matthew Turk, and Tobias Höllerer. 2014. World-Stabilized Annotations and Virtual Scene Navigation for Remote Collaboration. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. Association for Computing Machinery, New York, New York, USA, 449–459. https://doi.org/10.1145/2642918.2647372

Jeronimo Gustavo Grandi, Henrique Galvan Debarba, and Anderson Maciel. 2019. Characterizing Asymmetric Collaborative Interactions in Virtual and Augmented Realities. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 127–135. https://doi.org/10.1109/VR.2019.8798080

Seungyeon Huh, Shapna Muralidharan, Heedong Ko, and Byounghyun Yoo. 2019. XR Collaboration Architecture Based on Decentralized Web. In *The 24th International Conference on 3D Web Technology (Web3D '19)*. Association for Computing Machinery, New York, NY, USA, 1–9. https://doi.org/10.1145/3329714.3338137

Andrew Irlitti, Thammathip Piumsomboon, Daniel Jackson, and Bruce H Thomas. 2019. Conveying spatial awareness cues in xR collaborations. *IEEE Transactions on Visualization and Computer Graphics* 25, 11 (nov 2019), 3178–3189. https://doi.org/10.1109/TVCG.2019.2932173

Seungwon Kim, Gun Lee, Weidong Huang, Hayun Kim, Woontack Woo, and Mark Billinghurst. 2019. Evaluating the Combination of Visual Communication Cues for HMD-Based Mixed Reality Remote Collaboration. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300403

Gun A Lee, Theophilus Teo, Seungwon Kim, and Mark Billinghurst. 2017. Sharedsphere: MR Collaboration through Shared Live Panorama. In *SIGGRAPH Asia 2017 Emerging Technologies (SA '17)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3132818.3132827

Yongjae Lee, Changhyun Moon, Heedong Ko, Soo-Hong Lee, and Byounghyun Yoo. 2020. Unified Representation for XR Content and Its Rendering Method. In *The 25th International Conference on 3D Web Technology (Web3D '20)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/3424616.3424695

Yongjae Lee and Byounghyun Yoo. 2021. XR collaboration beyond virtual reality: work in the real world. *Journal of Computational Design and Engineering* 8, 2 (2021), 756–772. https://doi.org/10.1093/jcde/qwab012

David Lindlbauer and Andy D Wilson. 2018. Remixed Reality: Manipulating Space and Time in Augmented Reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3173703

Diego Marcos, Don McCurdy, and Kevin Ngo. 2020. A-Frame. Retrieved July 30, 2021 from https://aframe.io

Benjamin Nuernberger, Kuo-Chin Lien, Tobias Hollerer, and Matthew Turk. 2016. Anchoring 2D gesture annotations in augmented reality. In *2016 IEEE Virtual Reality (VR)*. IEEE, 247–248. https://doi.org/10.1109/VR.2016.7504746

Thammathip Piumsomboon, Arindam Day, Barrett Ens, Youngho Lee, Gun Lee, and Mark Billinghurst. 2017. Exploring Enhancements for Remote Mixed Reality Collaboration. In *SIGGRAPH Asia 2017 Mobile Graphics & Interactive Applications (SA '17)*. Association for Computing Machinery, New York, New York, USA, 1–5. https://doi.org/10.1145/3132787.3139200

Lev Poretski, Joel Lanir, and Ofer Arazy. 2018. Normative Tensions in Shared Augmented Reality. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (nov 2018), 1–22. https://doi.org/10.1145/3274411

Daeil Seo, Byounghyun Yoo, and Heedong Ko. 2017. Webized 3D Content Streaming System for Autostereoscopic 3D Displays. In *Proceedings of the 22nd International Conference on 3D Web Technology (Web3D '17)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3055624.3075940

Daeil Seo, Byounghyun Yoo, and Heedong Ko. 2018. Webizing Collaborative Interaction Space for Cross Reality with Various Human Interface Devices. In *Proceedings of the 23rd International ACM Conference on 3D Web Technology (Web3D '18)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3208806.3208808

Daeil Seo, Byounghyun E. Yoo, and Heedong Ko. 2016. Webizing Mixed Reality for Cooperative Augmentation of Life Experience. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion (CSCW '16 Companion)*. Association for Computing Machinery, New York, New York, USA, 401–404. https://doi.org/10.1145/2818052.2869078

Richard Szeliski. 2010. *Computer vision: algorithms and applications*. Springer Science & Business Media. https://szeliski.org/Book/

Theophilus Teo, Louise Lawrence, Gun A Lee, Mark Billinghurst, and Matt Adcock. 2019. Mixed Reality Remote Collaboration Combining 360 Video and 3D Reconstruction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3290605.3300431

Theophilus Teo, Mitchell Norman, Gun A Lee, Mark Billinghurst, and Matt Adcock. 2020. Exploring interaction techniques for 360 panoramas inside a 3D reconstructed scene for mixed reality remote collaboration. *Journal on Multimodal User Interfaces* 14, 4 (2020), 373–385. https://doi.org/10.1007/s12193-020-00343-x

The Khronos Group Inc. 2019. The OpenGL Graphics System: A Specification (Version 4.6 (Core Profile) - October 22, 2019). Retrieved September 26, 2021 from https://www.khronos.org/registry/OpenGL/specs/gl/glspec46.core.pdf

TopVIEW. 2021. Interactive 3D Virtual Tour Specialists. Retrieved July 11, 2021 from https://topview.co.nz/

Peng Wang, Xiaoliang Bai, Mark Billinghurst, Shusheng Zhang, Sili Wei, Guangyao Xu, Weiping He, Xiangyu Zhang, and Jie Zhang. 2020. 3DGAM: using 3D gesture and CAD models for training on mixed reality remote collaboration. *Multimedia Tools and Applications* (sep 2020), 1–26. https://doi.org/10.1007/s11042-020-09731-7

Wikitude GmbH. 2008. Wikitude. Retrieved September 25, 2021 from https://www.wikitude.com

A CODE EXAMPLE

```
// This code is written in Metal.
// This function returns a point having 3D position and color, using
    a depth map from LiDAR and the z-buffer of a 3D object renderer.
    The determination of if the point is representing OOI is included.
    This function executed for each pixel of the depth map in
    parallel.
kernel void compute_world_point_and_color(
        device const int* pixelWidth,
        device const int* colorWidth,
        device const int* depthTextureWidth,
        device const simd_float2* depthToTextureRatio,
        device const simd_float4x4 *perspective,
        device const float* depthTexture,
        device const simd_float2* dpethToImageRatio,
        device const float* depth,
        device const int* color,
        device simd_float3* worldPoint,
        device int* sampledColor,
        device const simd_float3x3* cameraIntrinsicsInversed,
        device const simd_float4x4* cameraTransform,
        uint index [[thread_position_in_grid]])
{
    // Because the data structure of the depth map in memory is a
        1-dimensional array, we need to calculate u, v index of the
        depth map.
    int ix = index % *pixelWidth;
    int iy = index / *pixelWidth;

    // We need to scale u, v index up to fit the image plane
        resolution. Practically, the resolution of the depth map is
        different from the image plane resolution. It is relatively
        small.
    const auto cameraPlanePoint = simd_float3(ix *
        dpethToImageRatio->x, iy * dpethToImageRatio->y, 1);

    // Calculate the world position of a point. Practically, because
        the axis directions of the image coordinate are different
        from those of the world coordinate, we need to multiply the
        flipYZ matrix that inverses the direction of the Y and Z axis.
    const auto localPoint = *cameraIntrinsicsInversed *
        cameraPlanePoint * depth[index];
    float4 _wp = *cameraTransform * flipYZ * simd_float4(localPoint,
        1);
    _wp /= _wp.w;


    // Find the z-value of the pixel corresponding to the point.
    int depthTextureIndex = int(ix * depthToTextureRatio->x) + int(iy
        * depthToTextureRatio->y) * *depthTextureWidth;
    auto modelDepth = depthTexture[depthTextureIndex];

    // Test if the z-value from the z-buffer of a 3D object is 1. If
        it is not 1, we discard the reconstructed point.
    if (modelDepth != 1) {
        _wp = float4(0,0,0,0);
    }

    // Color sampling for the reconstructed point.
    int colorIndex = (int)cameraPlanePoint.x +
        (int)cameraPlanePoint.y * *colorWidth;

    worldPoint[index] = _wp.xyz;
    sampledColor[index] = color[colorIndex];
}
```

Listing 1: Example code of reconstructing point cloud and masking it, in Metal