Unified Representation for XR Content and its Rendering Method

Yongjae Lee
Korea Institute of Science and Technology
Seoul, Korea
Department of Mechanical Engineering, Yonsei University
Seoul, Korea
yongjae.lee@kist.re.kr

Changhyun Moon
Korea Institute of Science and Technology
Seoul, Korea
ckdgus2482@kist.re.kr

Heedong Ko
Korea Institute of Science and Technology
Seoul, Korea
ko@kist.re.kr

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

Byounghyun Yoo
Korea Institute of Science and Technology
Seoul, Korea
yoo@byoo.net

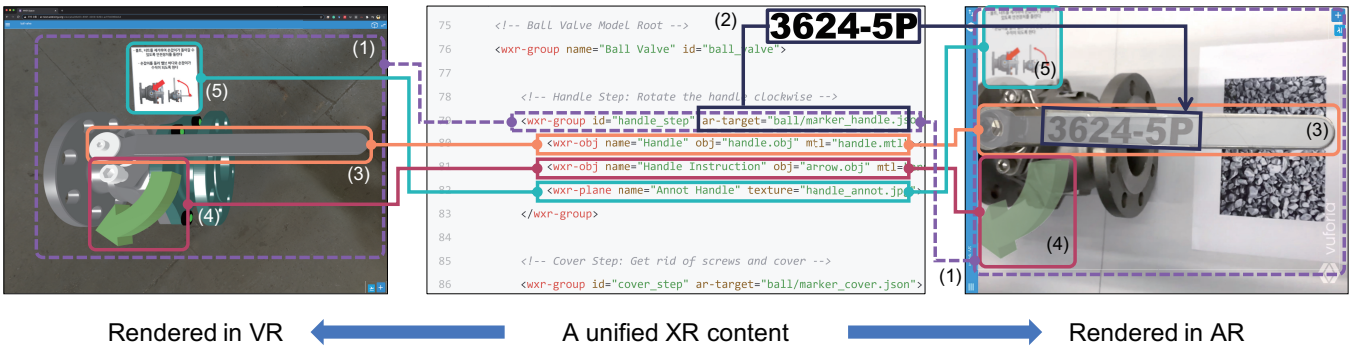

Figure 1: A unified XR content representation (middle) and its rendered results in VR (left) and AR (right). (1) is a virtual object group, which is represented by a wxr-group tag in the code, containing three virtual objects ((3), (4), and (5)). (2) is an ar-target attribute, which makes the wxr-group element work as an AR anchor as well, and its value is a URL that indicates the feature data. The feature data is a string image of '3624-5P' and is attached to the surface of the real handle (right). (3), (4), and (5) are virtual objects of the handle, curved arrow, and annotation, respectively. These virtual objects are rendered normally in VR (left) and augmented in AR (right).

ABSTRACT

Virtual Reality (VR) and Augmented Reality (AR) have become familiar technologies with related markets growing rapidly every year. Moreover, the idea of considering VR and AR as one eXtended reality (XR) has broken the border between virtual space and real space. However, there is no formal way to create such XR content except through existing VR or AR content development platforms. These platforms require the content author to perform additional tasks such as duplicating content for a specific user interaction environment (VR or AR) and associating them as one. Also, describing the content in an existing markup language (e.g., X3D, X3DOM, A-frame) has limitations of that the content author should predefine the user interaction environment (i.e., either of VR and AR). In this study, a unified XR representation is defined for describing XR content, and the method to render it has been proposed. The unified XR representation extends the HTML so that content authored with this representation can be harmoniously incorporated into existing web documents and can exploit resources on the World Wide Web. The XR renderer, which draws XR content on the screen, follows different procedures for both VR and AR situations. Consequently, the XR content works in both user interaction environment (VR and AR). Hence, this study provides a straightforward XR content authoring method that users access anywhere through a web browser

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs International 4.0 License.

Web3D '20, November 9–13, 2020, Virtual Event, Republic of Korea

© 2020 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-8169-7/20/11.
https://doi.org/10.1145/3424616.3424695

regardless of their situational contexts, such as VR or AR. It facilitates XR collaboration with real objects by providing both VR and AR users with accessing an identical content.

CCS CONCEPTS

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

KEYWORDS

Extended Reality, XR, Unified Representation, Virtual Reality, Augmented Reality, Collaboration, Content

ACM Reference Format:
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), November 9–13, 2020, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3424616.3424695

1 INTRODUCTION

The spread of COVID-19 around the world since early 2020 has brought many changes. People had to learn how to distance themselves from their daily lives. Many meetings and social gatherings that were traditionally held face-to-face have turned into virtual meetings. Thus, the demand for video conferencing, VR meetings, and remote AR collaboration services has increased. In particular, VR technology, which can utilize various intuitive audio-visual materials, is receiving much attention as it can provide a richer experience compared to video conferencing. VR Chat [VRChat Inc. 2017], which offers an endless collection of social VR experiences has been steadily gaining popularity with over 25, 000 communities created since its launch in 2017. Many academic events, including IEEE VR 2020 [IEEE 2020], have been held through VR.

While VR enriches the teleconferencing experience, it has certain limitations; for example, it only deals with virtual objects. Therefore, it is difficult for the user to recognize the status of objects that exist in real space. Due to this limitation, remote collaborations use AR technology to carry out work that requires real space. Applications such as Vuforia Chalk [PTC Inc. 2017] or Scope AR[Scope Technologies US Inc. 2018] help collaborators to deliver work instructions remotely by augmenting simple 3D models, such as drawings and arrows on the live video, while workers at the site share their situation over the video via camera-attached devices such as smartphones or AR glasses. However, there is a limit to the ability of remote collaborators to navigate the remote scene actively.

Recently, there is an emergence of the concept of eXtended Reality (XR) or X-Reality [Mann et al. 2018]. This eliminates the distinction between VR and AR and integrates them into a single concept. Studies before the advent of XR considered VR and AR content separately, whereas the concept of XR suggested that both VR and AR could be aggregated into one, by which users could access the content regardless of their interaction environment.

Many VR/AR/XR collaboration methods have been studied in the multi-disciplinary area as well [Huh et al. 2019; Lee et al. 2017, 2019; Poppe et al. 2012; Shen et al. 2010]. They proposed idiosyncratic collaborative systems for their scenario. To integrate their development beyond XR, they should share consensus of a task, then create the content following the scenario on their implementation bases. In other words, integrating them is not accessible due to the absence of a definition for their content. This problem can be solved if there is only one XR content representation on one platform, and if it is properly interpreted and rendered in any user interaction environment across VR and AR. Since the Web has one standard specification, it provides the same user experience in all environments. It can be accessed through a web browser that is already implemented in most devices; hence, it can be stated that the Web is the most appropriate platform for expressing XR content.

Collaboration in XR is featured by allowing users to access collaborative spaces in their user interaction environment, regardless of access to other users' interaction environments. However, collaboration in XR requires duplicated content that responds to each different user interaction environment. Therefore, code redundancy is a crucial problem making it difficult for the content author to maintain consistent content between different user interaction environments. It also creates storage space waste along with other costs.

In this paper, a unified XR content representation is defined and interpreted as VR or AR content, thereby solving the code redundancy problem. This representation extends the HTML, the markup language. It enables content authors to learn XR content authoring easily and express XR content to not depend on specific applications. Because it is HTML-based, it can be read and parsed usually by a web browser engine, and the hierarchy of the content source-block parsed into the document object model (DOM) can be exploited as a scene graph of XR content. Therefore, the code parser does not need to be written, and it has the advantage of accepting the vast content resources of the Web. Moreover, this nature of non-code-duplication and web-friendliness is preferable for Web 3.0, a decentralized web, and it will ensure that Peer to Peer (P2P) based XR collaboration is viable [Huh et al. 2019].

In Section 2, we discuss related work, including traditional methods for describing VR and AR content. Section 3 explains problems in making XR content with traditional strategies and details our approach that defines the definition and grammar of the unified XR representation with simple examples. In Section 4, an implementation of the proposed approach is presented. Finally, Section 5 concludes our research with a brief outline of future work.

2 RELATED WORK

In AR technology, one imperative element is to track information about the real world and the objects of interest [Carmigniani et al. 2011; Ibáñez and Delgado-Kloos 2018]. In general, there are two types of information tracking: location-based and image-based [Ibáñez and Delgado-Kloos 2018]. The former identifies the location where are the device is and the objects in the real world, using navigation systems (e.g., GPS, Beacon). The latter identifies objects using computer vision techniques (e.g., pattern recognition, SLAM), this is more desirable due to remarkable advancements in Artificial Intelligence (AI).

Many software development kits (SDKs) facilitating the approaches mentioned above have been developed and improved [Apple Inc.

2017; Google Inc. 2018; I Love IceCream Ltd. 2020; MAXST Ltd. 2017; PTC Inc. 2011; VisionStar Information Technology Ltd. 2015; Wikitude GmbH 2008]. ARCore [Google Inc. 2018] and ARKit [Apple Inc. 2017], developed by the two major mobile operating system vendors, Google and Apple, respectively, are implemented as native code and show excellent performance. However, they require burdensome work for utilization on the Web, due to lack of support from the web browser. Instead, Apple introduced a 3D model format called USDZ and an AR Quick Look extension, so that some functions of ARKit can be accessed in Safari [Apple Inc. 2019]. However, it remains inadaptable for collaboration-related work with real-world objects, as it is unable to recognize them. The ARToolKit [Kato and Billinghurst 1999], which is a famous AR project that has been ported and widely used in many languages, provides a pure web-based AR experience with the integration of A-Frame [Diego Marcos et al. 2015], a web-based VR framework. 8th Wall [8th Wall Inc. 2018] is another purely web-based AR engine that does not require supplementary installation, to witness the AR experience on the Web.

Unreal [Epic Games Inc. 1998] and Unity [Unity Technologies 2005] are currently the most widely used VR/AR content development platforms. Originally, they were intended to create VR content, but as AR games, including Pokémon Go, became prominent in the market, they also began to support AR content development and established an ecosystem of AR content with AR-supported devices, such as smartphones and Microsoft HoloLens. However, a crucial drawback is that these content development platforms do not support AR content building for the Web. Consequently, the developers must build the project separately for each target platform they intend to deploy. Amazon Sumerian [Amazon Web Services Inc. 2018], which is designed as a web-based VR/AR content development platform, provides everything from creation to deployment of content on the Web. Unlike Unreal or Unity, it is intended exclusively for the Web. Although VR/AR content development platforms help the content developer to implement a design with a quick and easy procedure, a key challenge that remains is creating VR-AR interoperable content.

The most recent VR/AR collaboration service is Spatial [Spatial Systems Inc. 2018]. Spatial provides a virtual workspace that can be accessed by any VR or AR devices. Users can collaborate on this virtual workspace, giving presentations, reviewing 3D models, exchanging documents, and so on. However, this collaboration has been limited to virtual objects and does not focus on real-world objects.

Some studies have emerged describing VR or AR content as human-readable representations. In general, these studies express VR or AR content based on markup language, which is characterized by a schema that is readable and modifiable by humans and is intelligible to computers. However, their content representation schema was designed only for specific VR or AR domains, so additional tasks are required to use in the other user interaction environment. In other words, VR content representation cannot be consumed as AR without any additional modification of content, or vice versa, through their methods. X3D [Web3D Consortium 2001] is a markup language that describes VR content, being created by the Web3D Consortium. X3D succeeds VRML, and its specification version 4.0 is currently being issued. X3DOM [Behr et al. 2009, 2011; Fraunhofer Society 2009] is a library implemented for embedding X3D representation into HTML documents. X3D is defined based on XML, so it is simple to learn and VR content can be written and used through rendering engines that support X3D. Unlike X3D, which uses markup language only to express the content, XML3D [Jankowski et al. 2013; Sons et al. 2010; Sutter et al. 2015] introduces CSS to separate the structure and style of the content. In X3D, the relationship between each virtual object that constitutes the content appears as a hierarchical structure of the document, and the properties of the virtual object are expressed as an attribute of the tag. XML3D considers the relationships between virtual objects and those properties as separable and makes those properties, such as transformations and textures, to be definable by CSS. As a result, XML3D increased the reuse of code by distinguishing between the structure and style of VR content, in the same manner normally achieved by web documents. Depending on the philosophy of expressing VR content using a markup language, such as X3D, A-Frame [Diego Marcos et al. 2015] applied the design pattern of the Entity-Component System to help content creators easily extend and use its functions. These previous studies have increased efficiency and utility in describing VR content; however, an important limitation is that they cannot be utilized in AR, as only VR situations have been considered.

ARML [Open Geospatial Consortium 2010] is a markup language defined by the Open Geospatial Consortium (OGC), which defines standards for data and services related to global space and is designed to express AR content. It is XML-based, like X3D; and describes GIS information, models to augment, and relative positions of anchor-to-anchor or anchor-to-user. KARML [Georgia Tech 2011] is an AR content-authoring language that extends KML for geographic annotation and visualization. Both ARML and KARML specify a location in the real world to augment virtual objects through geographic coordinates and are difficult to use in applications that do not utilize geographic information [Kim et al. 2011]. In order to overcome this limitation, data markup representation for mixed reality content [Kim et al. 2011] is proposed. It describes the real-world object and the virtual object according to 4H1W and endeavors to define the relationship between them. These attempts to express AR content may be used extensively and efficiently within the category of AR, but they are not suitable for describing VR content.

3 METHODOLOGY

The traditional development strategy of VR/AR content does not consider usage in another user interaction environment; content can only be used in the intended user interaction environment at the time of development. There are two main problems if the content is to be used in a different environment than its native one. The first problem is that virtual objects that will be augmented in AR will not be rendered in VR. For example, to create AR content in Amazon Sumerian, the following steps take place. The first step is to define 'ARAnchor', to reflect tracking information on the real-world object of the augmented target, and the second step is to insert the virtual object to be augmented into a child of 'ARAnchor' in the scene hierarchy (Figure 2). Thereby, the AR engine augments the virtual object when it recognizes the real-world object corresponding to

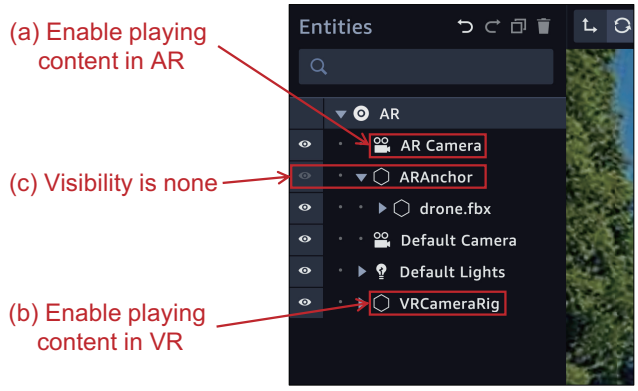

Figure 2: Scene hierarchy authored for AR content in Amazon Sumerian. The AR Camera node enables this content to be played in AR (a), and the VRCameraRig node enables it to be played in VR (b). The ARAnchor node is initially set as invisible to be disappeared until the real-world object is tracked (c).

'ARAnchor', or it does not render the virtual object. On the other hand, if this AR content is used in VR, the virtual objects intended for augmentation will never be rendered because there is no event recognizing the real-world objects in VR (Figure 3). The second problem of cross-environment rendering is that the background in VR also appears in AR and masks the video frame. For VR content, it is common to add a background in the form of a sphere or cube to enhance the sense of reality. When VR content with a background is used in AR, the background will always be rendered upon the video frame, making it impossible to see the video that shows the real world (Figure 4). Not only background but also a virtual object in VR content should not be rendered in AR whenever the corresponding real object, which the virtual object is mirrored from, exists in AR. These problems stem from a fundamental design that only takes into account a single interaction environment, either VR or AR, but fails to take a unified XR approach.

We need to interpret XR content as both VR and AR because it is unknown in advance which environment will be used. This paper proposes the XR representation and rendering method that can be used to interpret the environment of VR and AR with one XR content code. XR representation defines a basic element, **wxr-element**, by extending HTML's basic element, **HTMLElement**, and defines subclasses inheriting **wxr-element** for authoring XR content, as shown in Table 1 (Note that the level of hierarchy means the depth of inheritance.). Because XR representation has extended HTML, we can embed it in general web documents without errors and can parse into DOM by a browser engine. XR representation converted to DOM can be used as the scene tree of the XR scene, as it is in the DOM Tree structure, and the DOM event can be generated and processed using the DOM event handler. The user can also easily access content resources from other sites through the Web.

XR primitives (such as **wxr-world**, **wxr-space**, **wxr-camera**, etc.) inheriting **wxr-element** as root can be used to create XR content. With limited primitive types defined herein, the content

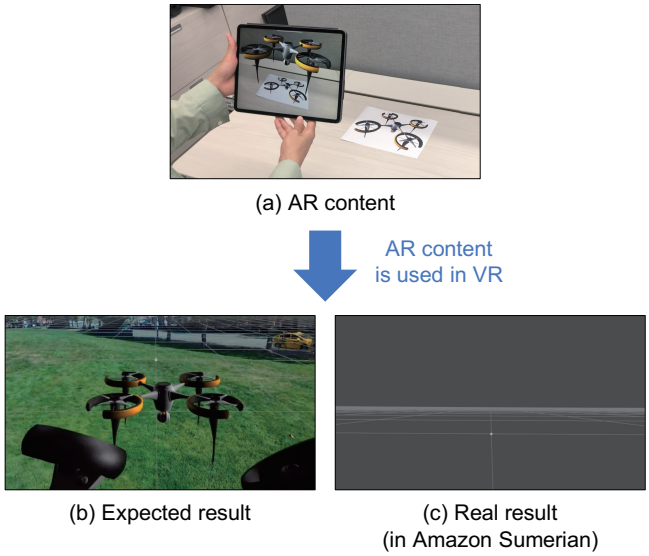

Figure 3: The problem when AR content is used in VR. (a) is the result when AR content (Figure 2) runs in AR. The drone model is augmented above the drone image. (b) is the expected result when the same content without any modification runs in VR. Note that the virtual object under the ARAnchor in the scene hierarchy is never rendered due to no real-world object tracking in (c).

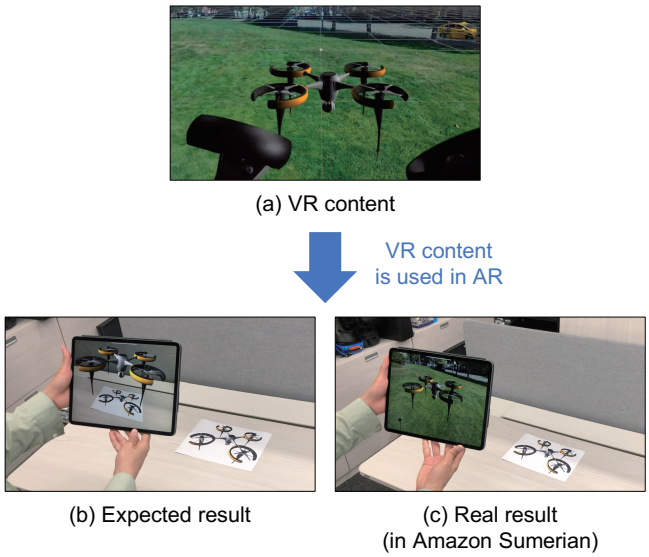

Figure 4: The problem when VR content is used in AR. (a) is the result when VR content runs in VR. The user surrounded by the background, inspects the drone model. (b) is the expected result when the same content without any modification runs in AR. Note that the virtual object, the drone model, is still rendered due to the real-world object is tracked in AR, even though the video frame is hidden by the background image in (c).

Table 1: WXR Tag Hierarchy

WXR Tag hierarchy			Description
Level 1	**Level 2**	**Level 3**	
wxr-element	wxr-world	-	The root tag for XR content. This tag defines the area of XR content in the document and initializes XR content described within this tag.
	wxr-space	-	The unit of XR space. This tag works as a namespace and multiple instances of this tag can be declared in the wxr-world tag. The background attribute of this tag refers to a URL of background image used in VR.
	wxr-camera	-	The XR camera. The user can navigate the virtual world controlling this virtual camera. In AR, the extrinsic and intrinsic parameters of this are synchronized with the physical camera on the device to serve the AR experience to the user properly.
	wxr-light	wxr-light-ambient, wxr-light-directional, wxr-light-spot, wxr-light-point	The XR light. The four basic lights inheriting wxr-light are specified here: directional, ambient, spot, and point light. The user freely defines more complicated light simulation like Rect Area light as required.
	wxr-group	-	The group for XR objects. This organizes several other XR objects as a single XR object. This also works as an AR anchor if the ar-target attribute has a value. The ar-target attribute points to the location where the information of the real-world object is stored. The location is identified using a URL.
	wxr-geometry	wxr-box, wxr-plane, wxr-sphere, etc.	The root tag of geometric objects. This tag contains an algorithm for collision checking and general properties of materials like opacity, texture filter, etc. Every geometric primitive inherits this tag. Moreover, the user can define custom tags inheriting this as per requirement. This tag also can have an ar-target attribute.

cannot be written flexibly. However, the **wxr-element** has extends HTML, allowing user-extension through subclassing of the XR primitive, for specific purposes. Listing 1 shows how to define a new **WXRObj** class with functions to import a 3D model file of .obj extension. **WXRObj** inherits **WXRGeometry** through the custom element interface. The registered custom element can then be seamlessly used as a composition of XR content, with predefined XR primitives.

```
class WXRObj extends WXRGeometry {
    constructor() {
        super();
    }

    static get is() {
        return "wxr-obj";
    }

    /* custom code implementation goes here... */
}

customElements.define(WXRObj.is, WXRObj);
```

Listing 1: XR Element Extension Example

The creation of XR content is similar to creating a web document. Just as a web document is structured through embedding HTML tags, the XR content is structured by embedding XR primitive tags. **wxr-world** is the element that acts as the root node of the XR scene hierarchy. **wxr-world** performs a necessary initialization process when used, such as creating an XR scene root node, camera controllers, and registering basic input event listeners such as mouse clicks and keyboard strokes. The XR content is organized by combining XR primitive tags with a **wxr-world** tag. **wxr-group** elements play the role of a group, as inferred from the tag name, which unite other primitives but play an even more important role in the AR environment. The **wxr-group** element has an attribute named ar-target, and it refers to the information of a target to be augmented as a URL. The AR engine obtains information about the target through the URL specified in the ar-target property and tracks the real-world object based on this. When the AR engine recognizes a real-world object through tracking, it renders a **wxr-group**; otherwise, it does not render the **wxr-group**. Moreover, the **wxr-geometry** tag and its descendants are also acceptable of ar-target property and capable of the same function.

An example of the **wxr-group** tag and ar-target attribute is shown in Figure 1. There are three XR objects (handle part of the ball valve, a curved arrow, directing where the handle is driven, and annotation, delineating how to disassemble the ball valve) declared within a **wxr-group** tag with the ID attribute set to 'handle_step'. This means that the three objects are entangled and manipulated together. Moreover, the **wxr-group** tag has an ar-target attribute referring to the feature information of the '3624-5P' image as a URL. If the AR engine tracks the '3624-5P' image on the handle in the real world, the **wxr-group** is augmented on the surface of the real handle based on the tracking information given by the AR engine. In other words, the renderer draws three virtual objects simultaneously on a real handle. If the **wxr-group** tag did not have an ar-target attribute, the AR engine would not track the object, and consequently, the three virtual objects enveloped in the **wxr-group** tag are not rendered in AR. In summary, the **wxr-group** element acts as a group in the VR and AR environment if there is no ar-target property, but also acts as an ARAnchor of an augmented target if the ar-target property exists. This is an important feature

that allows a renderer interpreting XR content to render a single unified XR code in VR and AR environments.

```
<html>
<style>
    .red {
        --wxr-color: #ff0000
    }
    .left {
        --wxr-transform: translate3d(-1,0,0);
    }
</style>
<body>
    <header>XR Content Example</header>
    <wxr-world>
        <wxr-camera>
        <wxr-space background="background.png">
            <wxr-plane class="red left"
                texture="plane.png"></wxr-plane>
            <wxr-obj obj="book.obj" mtl="book.mtl"
                ar-target="book.feature"
                style="--wxr-transform:translate3d(1,0,0);"></wxr-obj>
        </wxr-space>
    </wxr-world>
</body>
</html>
```

Listing 2: XR Content Example

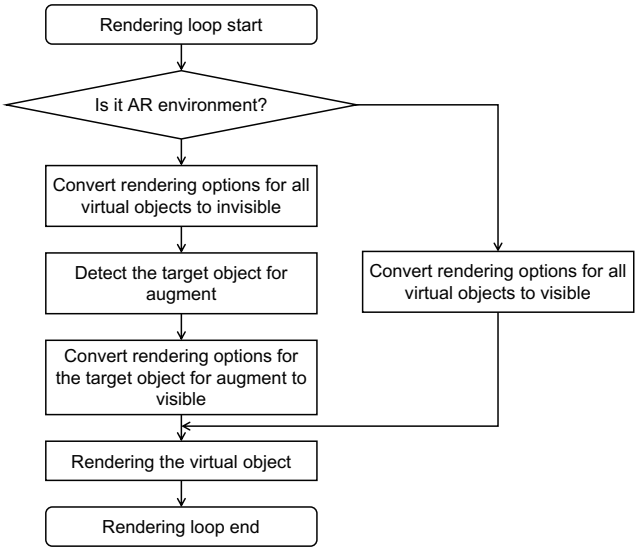

Figure 5: XR interpretation algorithm

Listing 2 shows an example of XR content embedded in the general HTML code of a web document. Much like the typical web document, the structure of XR content is declared through HTML-extended WXR tags, and the style of XR content is declared through CSS. Styles can be applied from inline, internal, and external styles according to HTML standards. By using CSS to separate the presentation and style of the content, it is not necessary to duplicate XR content code to apply different styles to the same content; that is, style management is simplified. The **wxr-space**, in this example, has a background property that shows a background in VR, not in AR. Within the **wxr-space** tag, two exemplar virtual objects are contained. The **wxr-plane** is an object placed at the point of (-1,0,0) with respect to the **wxr-space** element and colored as red. The **wxr-obj** is an object representing a book placed at the point of (1,0,0). Since the **wxr-obj** has an ar-target property that refers to the book's features, it is drawn if the AR engine tracks the feature in AR or not.

The existing VR/AR renderer was designed for only a single user interaction environment and once this is defined, it cannot properly interpret the content code for the other user interaction environment. In order for the proposed XR representation in this paper to be effective, XR content described by the XR representation must be rendered flexibly according to the circumstances that suit the user's environment. There are two possible problems with the cross-use of VR/AR content in different user interaction environments, as seen in the earlier example of Amazon Sumerian.

(1) The problem that virtual objects to be augmented in AR are not drawn in VR (Figure 3).
(2) The problem that the view of the camera in AR is blocked by the remaining background and virtual objects of VR (Figure 4).

The XR renderer considers each problem using the algorithm illustrated in Figure 5. The XR renderer stores the desired user interaction environment option selected by the user and uses it to interpret the XR code properly according to the user's interaction environment. If the user's interaction environment is AR, the XR renderer initially sets all virtual objects not-to-be-rendered and selectively renders only the augmentation object when the AR engine tracks the real-world object. On the other hand, all objects are always rendered regardless of the ar-target property in VR. For the second problem, the XR renderer specifies that the background and virtual objects are not rendered in AR so that the camera video frames are visible to the user. On the other hand, if the user's environment is VR, the renderer provides a realistic immersion to the user by drawing the background and virtual objects. For example, when the XR content code example in Listing 2 is interpreted through this algorithm, all virtual objects, including the background of **wxr-space** element, will be invisible in AR. Then, 'book.feature' is tracked by the AR engine; **wxr-obj**, in which the ar-target property referring to 'book.feature', is rendered. Meanwhile, all virtual objects, including the background of **wxr-space** element, will be visible in VR, allowing the user to grasp the comprehensive context of the entire scene. Figure 1 shows a more clear example. As described earlier, the XR renderer renders all the virtual objects, as shown in the picture on the left. In AR, the XR renderer renders nothing but only the augmented virtual objects (handle and curved arrow, and annotation) embedded within the **wxr-group**, which has **ar-target** attribute referring image feature of '3624-5P', as shown in the picture on the right.

Generally, the procedure to make an XR content takes five steps. The first step is preparing virtual resources such as 3D models and texture images and importing them to the current XR content project. The next step is to arrange them to have an organized structure and relationship. These two steps are commonly employed in making VR content and AR content. The third step is defining and registering a specific feature by which the AR engine tracks the real-world object. Up to this step is the general step for creating AR content. The following steps are for creating XR content. The fourth

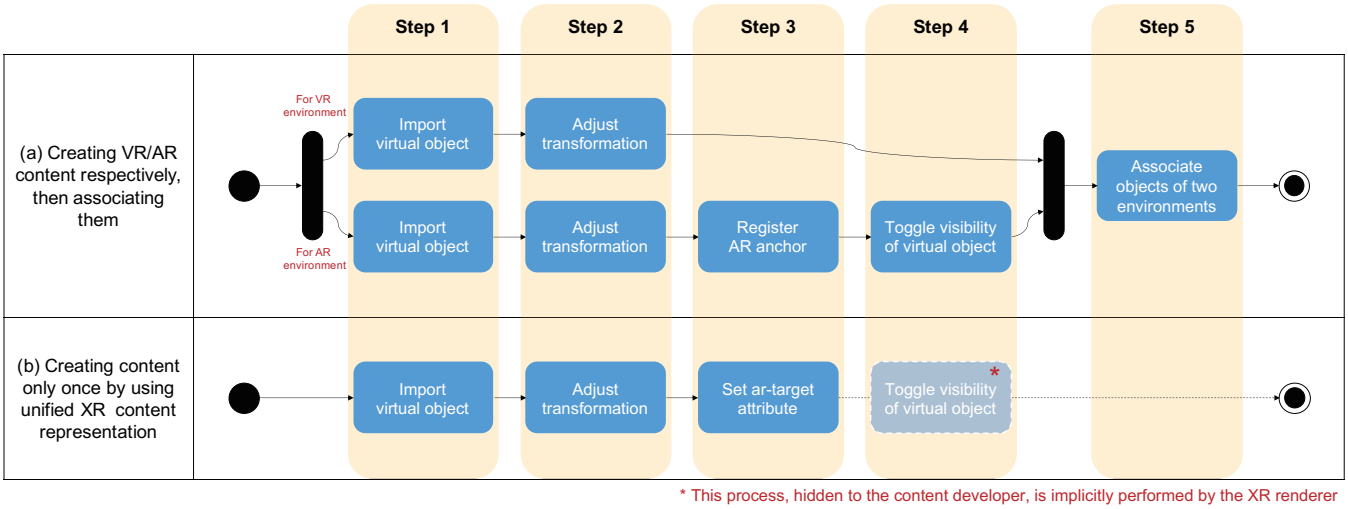

Figure 6: XR content generation approach comparison. (a) is a very primitive way to make XR content. Along with (a), the content developer should make two scenes for VR and AR, and then develop a program to combine the two scenes, to work as one. (b) is the method of using our proposed XR content representation and renderer. The developer working with this method is not tasked with writing any program for the cross-content compatibility between VR and AR. Note that Step 4 in (b) is not demanded due to XR renderer, and there is no Step 5 in (b) due to the singleness of XR scene content.

step is toggling the virtual object's visibility in accordance with its condition; thus, the object is drawn when the corresponding real-world object is tracked in AR. Additionally, the association step, linking the object in VR content with the same object in AR content (e.g., in Figure 1, the handle model of VR and the handle of AR), is needed for completing XR content making. With this step, the linked objects have the same properties as if they were one. Figure 6 shows a comparison of the needed author's tasks to make XR content between the traditional method and the proposed approach. In our method, the toggle step and association step are not needed to the author. The former is conducted automatically by the XR renderer, and the latter is due to the singleness of XR scene content, which is interoperable regardless of the user's interaction environment. The proposed method will dramatically reduce the XR content authoring task, which frequently occurs and is repeated. This is advantageous when there is a large number of XR objects that need to be created.

4 IMPLEMENTATION

A prototype of the XR content development library is implemented, called the WXR Library, for developers to create XR content without consideration of the user's interaction environment. The WXR Library offers two main features: First, an HTML-based markup language tag set (Table 1) for developing XR content; Second, an XR renderer (Figure 5), which interprets the XR content, builds up a scene tree, and draws the XR content according to the user's interaction environment. The WXR Library is implemented in JavaScript, and XR primitives are defined by extending **HTMLElement** through WebComponent technology. The THREE.js library is used to render virtual objects in a web browser. The XR renderer is implemented and embedded in the **wxr-world** element. The XR renderer has property about the user's interaction environment. The user chooses which interaction environment he is getting in by changing the property. According to the property, virtual objects in the scene are selectively drawn. The **wxr-element**, which is the root of the WXR tag hierarchy, has functions about reflecting changes of attributes of XR elements, including changes of CSS and hierarchical relation, on the corresponding THREE.js object or vice versa. Thus, other elements inheriting **wxr-element**, such as **wxr-camera**, are implemented focusing on only their idiosyncratic functions and logics without any further consideration of the attribute changes and reflection to the THREE.js objects. For creating an XR content, what the author has to do is just to import the WXR Library into a web document and to describe XR content along with HTML grammar, as shown in Listing 2.

The WXR Library provides authors with the capability of developing XR collaboration content that manipulates real-world objects. This is a distinctive feature compared to the recent XR collaboration service like Spatial, which only manipulates virtual data. Figure 7 shows the collaboration situation of real-world objects. This situation frequently occurs at an industrial plant where multiple machines are installed. As there are a few people who can control and fix all installed machines in the plant, experts' assistance is required. However, it is difficult for experts to reside in one factory constantly. In this situation, XR collaboration would be beneficial to experts and local workers. Without visiting there, the expert can give the directives to the local workers, based on the state of virtual objects synchronized with the real-world objects in the site through the single XR content. The local workers can catch and follow the directives through augmented virtual objects on real-world ones. The following three examples are provided for a better understanding of XR collaboration using the WXR Library.

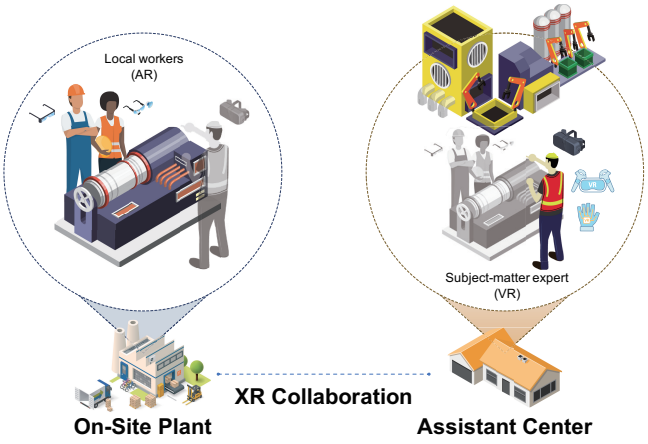

Figure 7: Description of the situation of XR collaboration manipulating a real-world object. The object and local workers participating in the collaboration are on-site in the plant (left-side). They receive the directives for the task through an AR device (i.e., AR glasses) from the expert who is at the assistant center in the remote. The subject-matter expert participates in the collaboration through a VR device (i.e., HMD) (right-side). The expert and local workers can recognize each other through the virtual world or through augmented objects on real-world objects. Note that the translucent gray graphics in the figure represent virtual objects.

Figure 8 shows a collaboration situation between a local worker and a remote expert for replacing a powder keg of a metal 3D printer. If the 3D printer's material has been exhausted, the existing powder keg needs to be replaced with a new one filled with metal powder. However, workers in the field do not have the relevant knowledge. Hence, a collaboration by requesting remote assistance from the expert is a favorable solution to this situation. The remote expert participating through VR shows the required order of work to the local worker by manipulating virtual objects in an environment surrounded by a realistic background, which is a photo of the site area. Because field workers have to deal with real objects, they participate through AR and intuitively learn the sequence of work by augmented objects over the real objects. In the traditional VR/AR content development tool, the content for each of the user interaction environments should be created separately, but it can be written at once via the WXR Library. Users use the XR content by switching the property of XR renderer about user interaction environment as needed.

Listing 3 is the XR content code for the situation of Figure 8. This code-block starting with a **wxr-world** tag is excerpted from an entire HTML document. The **style** tag at the top defines CSS rules for the XR content. Currently, two properties (**–wxr-color**, **–wxr-transform**) are supported in the WXR Library. After the style declaration, the XR content declaration is followed. The attributes of the **wxr-camera** are changed according to the intrinsic properties of the physical camera of the device when the user is in AR. The following **wxr-space** tag conceptually distinguishes it from another XR scene; for example, the XR scene of maintenance for other parts of the 3D printer. An XR scene consists of several elements, including the background. The background property of **wxr-space**, referring to 360 images on the desktop, and all 3D objects including the powder keg (gray cylinder-shaped object), the powder plate (blue colored object under the keg), and the base (compound of dark gray and magenta-colored objects) are rendered in VR (Figure 8a), since all virtual objects are visible in VR (Figure 5). Meanwhile, only objects within the **wxr-group** with ID powder_keg_step are rendered in AR (Figure 8b), since all virtual objects are initially invisible and only the objects tracked by the AR Engine are rendered.

```
<!-- Other HTML tags of an web document... -->
...

<wxr-world>
<style>
    /* Style definitions for XR objects... */
    wxr-camera[name="Main Cam"] {
        --wxr-transform : translate3d(-0.1881,0.8855,0.7293)
            rotate3d(0.01194,-0.1695,0.002);
    }
    #powder_keg_step wxr-obj[name="Powder Keg"] {
        --wxr-transform: translate3d(-0.034,0.1863,0.137)
            rotate3d(0,3.14,0) scale3d(0.9,0.9,0.9);
    }
    #powder_keg_step wxr-obj[name="Powder Plate"] {
        --wxr-transform: translate3d(-0.034,0.078,0.137)
            rotate3d(0,-0.55,0) scale3d(1,1,1);
        --wxr-color: #00ff00;
    }
    ...
</style>

<wxr-camera name="Main Cam" fovy="61" near="0.05" far="1000" fov="49"
    aspect="0.75"></wxr-camera>
<wxr-space background="on_top_of_desk1.png">

    <!-- Environment objects like lights... -->
    <wxr-light-ambient name="Ambient Light"
        style="color:0xf0f0f0"></wxr-light-ambient>
    ...

    <!-- Powder model group -->
    <wxr-group name="Powder Model" id="powder_model">

        <!-- Powder Keg Step: Disassemble powder keg from base -->
        <wxr-group id="powder_keg_step" ar-target="powder/keg">
            <wxr-obj name="Powder Keg" obj="powder.obj"
                mtl="powder.mtl"></wxr-obj>
            <wxr-obj name="Powder Plate"
                obj="plateassy.obj"></wxr-obj>
            <wxr-obj name="Left Arrow" obj="leftarrow.obj"></wxr-obj>
            <wxr-obj name="Right Arrow" obj="leftarrow.obj"></wxr-obj>
            <wxr-plane name="Annot Keg"
                texture="powder_keg_annot.png"></wxr-plane>
        </wxr-group>

        <!-- Bush Step: Pick out bush and shaft from base -->
        <wxr-group id="bush_step" ar-target="powder/bush">
            ...
        </wxr-group>

        <!-- Base model (not augmented) -->
        <wxr-obj name="Base Front" obj="basefront.obj"></wxr-obj>
        <wxr-obj name="Base Back" obj="baseback.obj"></wxr-obj>
        <wxr-obj name="Base Bottom" obj="basebottom.obj"></wxr-obj>
    </wxr-group>
</wxr-space>
</wxr-world>
```

Listing 3: XR code snippet for example 1

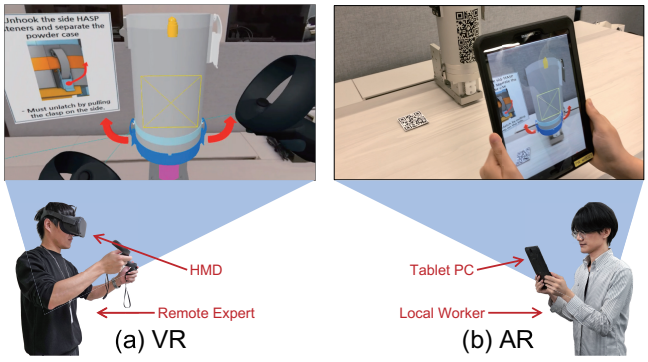

Figure 8: Rendering result of Listing 3 in VR and AR. The expert in VR (accessed with an Oculus Quest HMD) is demonstrating how to disassemble the powder keg (gray cylinder) from the base of a metal 3D printer (assembled part of magenta shaft and dark gray box) (a), and the local worker is watching it through AR with a Samsung Galaxy Tab S3 tablet (b). The object tracked by the AR engine is the wxr-group with ID 'powder_keg_step'; thus, five objects (named as Powder Keg, Powder Plate, Left Arrow, Right Arrow, and Annot Keg) are augmented on the worker's screen. Note that all objects, including the wxr-group with ID 'bush_step' and the base part, are rendered in VR, whereas only objects within the wxr-group with ID 'handle_step' are rendered in AR.

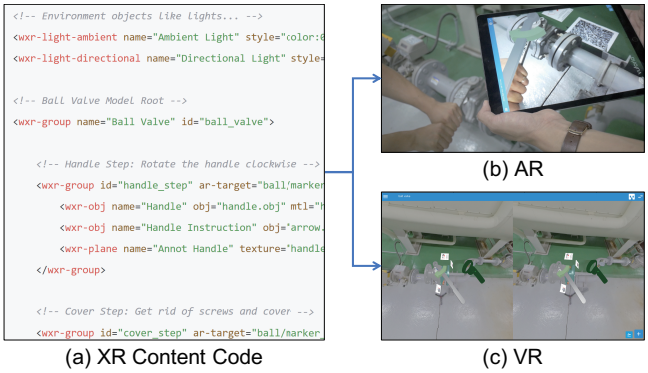

Figure 9: The XR content code snippet (a) for replacing ball in ball valve and its rendering result in both VR and AR. The local worker is closing the valve after receiving instructions from the expert, and another local worker holding the iPad is observing this (b). The remote expert is observing the process in VR with an Acer Windows Mixed Reality headset (c). The tracked object by the AR engine is the wxr-group with ID 'hadle_step'; thus, three objects (named as Handle, Handle Instruction, and Annot Handle) are augmented on the worker's screen. Note that all virtual objects, including wxr-group with ID 'cover_step' and the base part, are rendered in VR, albeit only objects within wxr-group with ID 'handle_step' are rendered in AR.

The ball valve is one of the most common mechanical devices in industrial plants. The ball valve contains a perforated ball that controls fluid passing through pipes. Friction caused by rotating the ball or collision with foreign substances passing through the valve may cause damage to the ball. In order to replace the damaged ball, a remote expert manipulates the virtual object of the ball valve in VR to demonstrate the process of dismantling the ball valve to the workers at the site, and the local workers follow the augmented virtual object's movements. Figure 9 shows the rendering results in each user interaction environment from a single XR code, as in the previous example.

Figure 10 shows the operation of the exhaust valve. Owing to the exhaust valve characteristics through which hot gases pass, the parts inside are prone to corrosion. To replace the parts inside the exhaust valve, the valve must be opened first. However large exhaust valves such as the one in the picture are heavy, making it difficult to control them manually. In the third example, an expert in the remote area informs the local workers of a lever that needs to be manipulated to open the exhaust valve in VR; workers in the field then follow the work instructions through AR and observe the valve opening normally, alongside the expert who is in VR. Similarly, this example content was written in XR code, which the expert and workers used in VR and AR, respectively.

5 CONCLUSION

In this paper, we discussed the limitations of cross-usability (using AR content in VR or vice versa) in the traditional methods describing VR/AR content; and proposed a method for authoring unified XR content and its interpretation algorithm. The unified XR content representation addresses the threshold of cross-usability and duplicated content creation depending on the users' interaction environment. The XR content representation extends HTML to be parsed to DOM without errors in the web browser and can be easily embedded into existing web documents. Besides, content authors can extend the basic primitive defined here to add new features and achieve high scalability. The XR renderer visualizes the XR content code without the two problems of VR/AR cross-use, by employing rendering algorithms of different methods, depending on the user's user interaction environment. These features enable the developer to create XR content interoperable between different devices, e.g., AR Glasses, Smartphone, HMD, that support HTML5.

The implementation and three examples discussed above provide us inspiration for remote cooperation through XR. The workers collaborate on real objects without having to gather in one geographic location. Although they are physically separated from each other, they can still participate in collective activities and tasks using a single XR content. However, there is still scope for improvement. For example, the user in VR has difficulty recognizing the entire situation of the site in AR. The background in VR gives the user an immersive experience, but it may contribute to misunderstanding, as only parts of the environment represent the actual environment in real-time. Therefore, the future work will concentrate on more comprehensive research required to address the impending problem by reflecting the dynamic environmental changes of the site in AR into the background in VR.

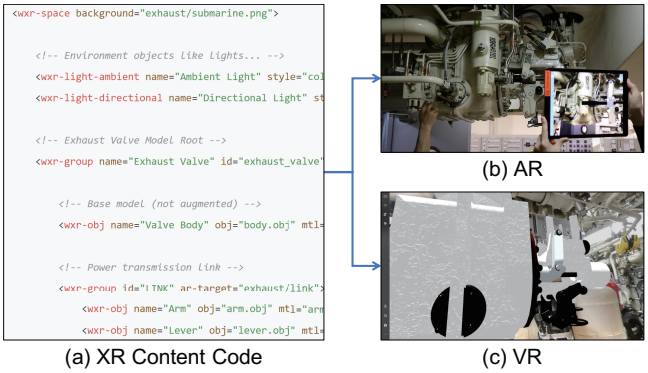

Figure 10: The XR content code snippet (a) for opening/closing exhaust valve and its rendering result in both VR and AR. The local worker is lowering the lever of the valve following the instructions from the remote expert for closing the valve, and another local worker holding the iPad is observing the valve operating (b). The remote expert is observing the process in VR with an Acer Windows Mixed Reality headset (c). The object tracked by the AR engine is a wxr-group with ID LINK; thus, four objects (named with Arm, Lever, and Joint) are augmented on the worker's screen. Note that all objects, including body parts, are rendered in VR, albeit only objects within wxr-group with ID LINK, are rendered in AR.

ACKNOWLEDGMENTS

This work was supported by the Korea Institute of Science and Technology (KIST) under the Institutional Program (Grant No. 2V08640 and 2E30270).

REFERENCES

8th Wall Inc. 2018. 8th Wall. Retrieved July 17, 2020 from https://www.8thwall.com

Amazon Web Services Inc. 2018. Amazon Sumerian. Retrieved July 17, 2020 from https://aws.amazon.com/sumerian

Apple Inc. 2017. ARKit. Retrieved July 20, 2020 from https://developer.apple.com/augmented-reality

Apple Inc. 2019. AR Quick Look. Retrieved July 20, 2020 from https://developer.apple.com/augmented-reality/quick-look

Johannes Behr, Peter Eschler, Yvonne Jung, and Michael Zöllner. 2009. X3DOM: A DOM-Based HTML5/X3D Integration Model. In *Proceedings of the 14th International Conference on 3D Web Technology (Web3D '09)*. Association for Computing Machinery, New York, NY, USA, 127–135. https://doi.org/10.1145/1559764.1559784

Johannes Behr, Yvonne Jung, Timm Drevensek, and Andreas Aderhold. 2011. Dynamic and interactive aspects of X3DOM. In *Proceedings - 16th International Conference on 3D Web Technology, Web3D 2011 (Web3D '11)*. Association for Computing Machinery, New York, NY, USA, 81–87. https://doi.org/10.1145/2010425.2010440

Julie Carmigniani, Borko Furht, Marco Anisetti, Paolo Ceravolo, Ernesto Damiani, and Misa Ivkovic. 2011. Augmented reality technologies, systems and applications. *Multimedia Tools and Applications* 51, 1 (jan 2011), 341–377. https://doi.org/10.1007/s11042-010-0660-6

Diego Marcos, Don McCurdy, and Kevin Ngo. 2015. A-Frame. Retrieved July 17, 2020 from https://aframe.io

Epic Games Inc. 1998. Unreal Engine. Retrieved July 17, 2020 from https://www.unrealengine.com

Fraunhofer Society. 2009. x3dom.org. Retrieved July 17, 2020 from https://www.x3dom.org

Georgia Tech. 2011. KHARMA. Retrieved July 17, 2020 from http://kharma.gatech.edu

Google Inc. 2018. ARCore. Retrieved July 20, 2020 from https://developers.google.com/ar

Seungyeon Huh, Shapna Muralidharan, Heedong Ko, and Byounghyun Yoo. 2019. XR collaboration architecture based on decentralized web. In *Proceedings - Web3D 2019: 24th International ACM Conference on 3D Web Technology*. Association for Computing Machinery, Inc, New York, NY, USA, 1–9. https://doi.org/10.1145/3329714.3338137

I Love IceCream Ltd. 2020. DeepAR. Retrieved July 17, 2020 from https://www.deepar.ai

María Blanca Ibáñez and Carlos Delgado-Kloos. 2018. Augmented reality for STEM learning: A systematic review. *Computers and Education* 123 (aug 2018), 109–123. https://doi.org/10.1016/j.compedu.2018.05.002

IEEE. 2020. IEEE VR 2020. Retrieved July 17, 2020 from https://ieeevr.org/2020

Jacek Jankowski, Sandy Ressler, Kristian Sons, Yvonne Jung, Johannes Behr, and Philipp Slusallek. 2013. Declarative integration of interactive 3D graphics into the world-wide web: Principles, current approaches, and research agenda. In *Proceedings - Web3D 2013: 18th International Conference on 3D Web Technology (Web3D '13)*. Association for Computing Machinery, New York, NY, USA, 39–46. https://doi.org/10.1145/2466533.2466547

H. Kato and M. Billinghurst. 1999. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *Proceedings - 2nd IEEE and ACM International Workshop on Augmented Reality, IWAR 1999*. Institute of Electrical and Electronics Engineers Inc., 85–94. https://doi.org/10.1109/IWAR.1999.803809

Hyejin Kim, Hyoseok Yoon, Ahyoung Choi, Woonhyuk Baek, Ilgu Lee, Dongchul Kim, and Woontack Woo. 2011. Data Markup Representation for Mixed Reality Contents. In *International AR Standards Meeting*. http://www.perey.com/ARStandards/GIST{_}MRContents.pdf

Gun A. Lee, Theophilus Teo, Seungwon Kim, and Mark Billinghurst. 2017. Mixed reality collaboration through sharing a live panorama. In *SIGGRAPH Asia 2017 Mobile Graphics and Interactive Applications, SA 2017*. ACM Press, New York, New York, USA, 1–4. https://doi.org/10.1145/3132787.3139203

Gun A. Lee, Theophilus Teo, Seungwon Kim, and Mark Billinghurst. 2019. A User Study on MR Remote Collaboration Using Live 360 Video. In *Proceedings of the 2018 IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2018*. Institute of Electrical and Electronics Engineers Inc., 153–164. https://doi.org/10.1109/ISMAR.2018.00051

Steve Mann, Tom Furness, Yu Yuan, Jay Iorio, and Zixin Wang. 2018. All Reality: Virtual, Augmented, Mixed (X), Mediated (X,Y), and Multimediated Reality. *arXiv preprint arXiv:1804.08386* (apr 2018). arXiv:1804.08386 http://arxiv.org/abs/1804.08386

MAXST Ltd. 2017. MAXST. Retrieved July 17, 2020 from http://maxst.com

Open Geospatial Consortium. 2010. ARML. Retrieved July 17, 2020 from https://www.ogc.org/standards/arml

Erik Poppe, Ross Brown, Daniel Johnson, and Jan Recker. 2012. Preliminary Evaluation of an Augmented Reality Collaborative Process Modelling System. In *2012 International Conference on Cyberworlds*. IEEE, 77–84. https://doi.org/10.1109/CW.2012.18

PTC Inc. 2011. Vuforia. Retrieved July 17, 2020 from https://www.ptc.com/en/products/augmented-reality/vuforia

PTC Inc. 2017. Vuforia Chalk. Retrieved July 17, 2020 from https://chalk.vuforia.com

Scope Technologies US Inc. 2018. Scope AR. Retrieved July 17, 2020 from https://www.scopear.com

Y Shen, S.K. Ong, and A.Y.C. Nee. 2010. Augmented reality for collaborative product design and development. *Design Studies* 31, 2 (mar 2010), 118–145. https://doi.org/10.1016/j.destud.2009.11.001

Kristian Sons, Felix Klein, Dmitri Rubinstein, Sergiy Byelozyorov, and Philipp Slusallek. 2010. XML3D: Interactive 3D Graphics for the Web. In *Proceedings of the 15th International Conference on Web 3D Technology (Web3D '10)*. Association for Computing Machinery, New York, NY, USA, 175–184. https://doi.org/10.1145/1836049.1836076

Spatial Systems Inc. 2018. Spatial. Retrieved July 17, 2020 from https://spatial.io

Jan Sutter, Kristian Sons, and Philipp Slusallek. 2015. A CSS Integration Model for Declarative 3D. In *Proceedings of the 20th International Conference on 3D Web Technology (Web3D '15)*. Association for Computing Machinery, New York, NY, USA, 209–217. https://doi.org/10.1145/2775292.2775295

Unity Technologies. 2005. Unity. Retrieved July 17, 2020 from https://unity.com

VisionStar Information Technology Ltd. 2015. EasyAR. Retrieved July 17, 2020 from https://www.easyar.com

VRChat Inc. 2017. VRChat. Retrieved July 17, 2020 from https://vrchat.com

Web3D Consortium. 2001. X3D. Retrieved July 17, 2020 from https://www.web3d.org/x3d/what-x3d

Wikitude GmbH. 2008. Wikitude. Retrieved July 17, 2020 from https://www.wikitude.com