XSpace: An Augmented Reality Toolkit for Enabling Spatially-Aware Distributed Collaboration

JAYLIN HERSKOVITZ, University of Michigan, USA
YI FEI CHENG, Carnegie Mellon University, USA
ANHONG GUO, University of Michigan, USA
ALANSON P. SAMPLE, University of Michigan, USA
MICHAEL NEBELING, University of Michigan, USA

Augmented Reality (AR) has the potential to leverage environmental information to better facilitate distributed collaboration, however, such applications are difficult to develop. We present XSpace, a toolkit for creating spatially-aware AR applications for distributed collaboration. Based on a review of existing applications and developer tools, we design XSpace to support three methods for creating shared virtual spaces, each emphasizing a different aspect: shared objects, user perspectives, and environmental meshes. XSpace implements these methods in a developer toolkit, and also provides a set of complimentary visual authoring tools to allow developers to preview a variety of configurations for a shared virtual space. We present five example applications to illustrate that XSpace can support the development of a rich set of collaborative AR experiences that are difficult to produce with current solutions. Through XSpace, we discuss implications for future application design, including user space customization and privacy and safety concerns when sharing users' environments.

CCS Concepts: • **Human-centered computing → Interface design prototyping**; **Mixed / augmented reality**.

Additional Key Words and Phrases: Augmented reality, Distributed collaboration, Toolkit

ACM Reference Format:
Jaylin Herskovitz, Yi Fei Cheng, Anhong Guo, Alanson P. Sample, and Michael Nebeling. 2022. XSpace: An Augmented Reality Toolkit for Enabling Spatially-Aware Distributed Collaboration. *Proc. ACM Hum.-Comput. Interact.* 6, ISS, Article 568 (December 2022), 26 pages. https://doi.org/10.1145/3567721

1 INTRODUCTION

Augmented reality (AR) can enable a wide range of collaborative applications by supporting interaction with physical environments and conversational grounding through shared virtual landmarks [38, 48]. Prior research has described many benefits of shared environments for collaborative work, including creating a persistent context for ongoing activity, enabling peripheral awareness of others, facilitating chance encounters, and promoting usability via spatial metaphors [4]. AR has the potential to extend these benefits to situations where collaborators are spatially distributed by sharing each user's unique environmental context, which would help address key limitations of traditional video conferencing systems.

Authors' addresses: Jaylin Herskovitz, University of Michigan, Ann Arbor, MI, USA, jayhersk@umich.edu; Yi Fei Cheng, Carnegie Mellon University, Pittsburgh, PA, USA, yifeic2@andrew.cmu.edu; Anhong Guo, University of Michigan, Ann Arbor, MI, USA, anhong@umich.edu; Alanson P. Sample, University of Michigan, Ann Arbor, MI, USA, apsample@umich.edu; Michael Nebeling, University of Michigan, Ann Arbor, MI, USA, nebeling@umich.edu.

2573-0142/2022/12-ART568 $15.00
https://doi.org/10.1145/3567721

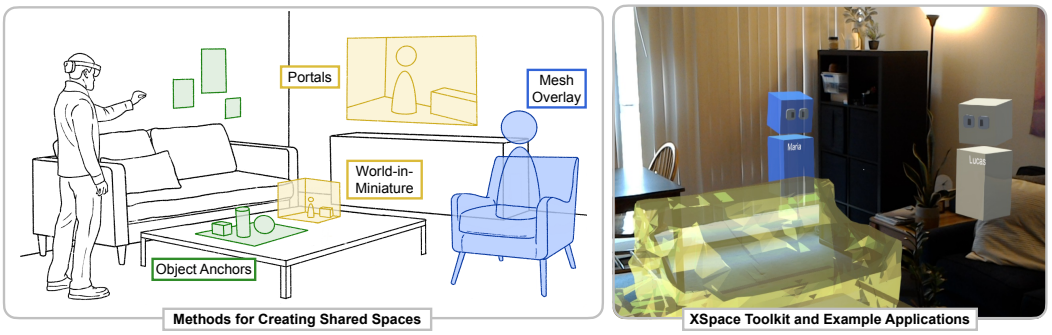

Fig. 1. XSpace is a toolkit for creating spatially-aware AR applications for distributed collaboration. This figure highlights some of the key components of our work. (Left) Through a review of prior applications, we identify three key methods for creating shared spaces in distributed AR applications. *Object-centric* methods, expressed in XSpace using object anchors to place content; *perspective-driven* methods, expressed in XSpace using both portals and a world-in-miniature view; and *mesh-based* methods, expressed in XSpace using a mesh overlay. (Right) XSpace provides a Unity toolkit allowing developers to add support to existing applications for distributed users.

While prior work has envisioned a number of innovative ways for users to share perspectives [25, 62], environmental information [22, 46, 62], or even to merge two distinct environments [20, 59] in both AR and virtual reality (VR), distributed multi-user AR applications remain difficult to create. A variety of tools have been developed for quickly prototyping AR applications [17, 29, 33, 41, 42], but toolkits that raise the ceiling of what developers can create, in particular, with a view towards collaborative AR applications are limited [15, 56]. Recently, Microsoft introduced Mesh [37] to provide a platform for distributed multi-user AR applications; however, their focus appears to be on the prerequisite problem of sharing expressive avatars. In general, there is a lack of tools to help developers explore and implement the rich and contextual variety of sharing scenarios that we see in experimental systems.

In this paper, we contribute the design and development of XSpace, a toolkit for creating spatially-aware AR applications for distributed collaboration, enabling a technical exploration of how different types of collaborative AR applications can be constructed. We designed XSpace based on an analysis of prior work and developer tools, which we distilled into three primary methods for constructing a shared virtual space: *object-centric*, *perspective-driven*, and *mesh-based*. Each of these methods centers the collaboration around a different spatial aspect used for coordination and alignment of environments, and we demonstrate how they can be used in conjunction to enable a variety of application scenarios.

XSpace implements these three key methods as a toolkit to simplify the use of different operations for sharing information across space, provides utilities for environmental scanning and avatar management, and exposes its functionality in Unity for easy integration into existing applications. XSpace also provides complementary visual tools that allow designers to explore various configurations of shared spaces for their application without coding, allowing rapid prototyping of shared AR experiences via direct manipulation.

We evaluate how XSpace can be used to support a variety of applications by implementing promising application scenarios that are difficult to produce currently, as well as analyzing the development effort required. Using XSpace, we create three multi-user distributed AR applications—a co-working environment, a collaborative furniture layout design application, and a multiplayer game—and demonstrate how the methods implemented in XSpace can be used and combined to create a variety of collaboration modes. Per Ledo et al. [26], this is a Type 1 validation by

demonstration, a popular and accepted technique in 68 published toolkits papers. We also highlight the developer effort required to create multi-user distributed AR applications using XSpace by comparing the lines of code needed to create single- and multi-user versions of the same application.

We show that XSpace can cover a range of application types and use cases, demonstrating flexibility and expressive power [45] for developers to explore new types of applications that were not previously feasible. By combining known techniques like object anchors, portals, world-in-miniature, and environmental mesh sharing, XSpace enables a variety of applications to be created and customized to specific needs, something that was not possible before without developer effort to construct each case. We also demonstrate that XSpace can do this with only a small amount of code added by developers.

Through XSpace, we identified the components needed to create a variety of distributed AR applications and how a developer toolkit could be constructed. As a result, we also generate implications for future distributed collaborative application design, user space customization, and mesh privacy that have not yet been considered. Overall, XSpace takes an important step toward supporting richer distributed AR collaboration by leveraging users' local environments to create a shared spatial context, and raising the ceiling of AR tool support to inspire design of future collaborative applications.

2 RELATED WORK

XSpace draws from prior research in the following areas: *(i)* mutual awareness in collaborative work, *(ii)* collaboration in Extended Reality (XR), and *(iii)* XR prototyping and development tools. In this section, we highlight key contributions from these and draw comparisons to our own work. Later in Section 3, we further draw from prior work, applications, and tools to present a set of three primary methods for creating shared AR spaces, which we use to design XSpace's features.

2.1 Mutual Awareness

Building mutual awareness has long been recognised as a critical requirement of collaborative virtual environments [12, 19, 52]. Gutwin and Greenberg defined this concept, which they termed "workspace awareness," as "the up-to-the minute knowledge a person holds about another's interaction with the workspace" [19]. This understanding consists of four aspects: *(1)* who is involved, *(2)* where they are working, *(3)* what they are doing, and *(4)* what their intended future actions are.

Prior research has explored a variety of techniques to support mutual awareness. The approach of direct relevance to XSpace is creating shared spaces [3, 4, 44]. Commonly demonstrated benefits of enabling multiple participating users, particularly in remote scenarios, to share spaces include: *(i)* creating a persistent context for on-going activity [4], *(ii)* enabling peripheral as well as focused attention of the activities of others [21], and *(iii)* facilitating serendipitous interactions [50].

These factors are common aims of collaborative XR applications; AR particularly has high potential towards this end as users can interact with their physical environments to ground the experience. However, designing and developing distributed AR applications that can maintain a shared sense of space across varying user environments is difficult. Our intent with XSpace is to address this difficulty and explore the potential of XR technologies in enabling the construction of richer shared environments.

2.2 Collaboration in XR

The prospect of using XR to support collaborative tasks has been discussed for two decades [4, 5, 53]. However, only recently has XR technology become sufficiently mature to support the complex collaborative scenarios envisioned in the past [8, 32]. For example, Room2Room [47] enabled the recreation of face-to-face conversations by projecting a life-size spatial capture of a remote user

Table 1. We draw on prior research to identify a set of methods for creating distributed AR/VR spaces for collaboration, which we then support with XSpace.

Object Centric	**Perspective Driven**		**Mesh Based**
Shared Anchors	**Portals**	**World-in-Miniature**	**Crop and Overlay**
Spatial [55], Mesh [37]	Photoportals [25]	Remixed Reality [31]	Remixed Reality [31]
Room2Room [47]	Slice of Light [65]	Photoportals [25]	MirageTable [6]
Müller et al. [39]	MirageTable [6]	Loki [62]	Slice of Light [65]
Congdon et al. [11]	Loki [62]		Loki [62]
	Physical Telepresence [27]		Holoportation [46]
			Physical Telepresence [27]

into a local user's space. XRDirector [40] enabled multiple designers to collaborate in AR/VR to prototype 3D movie scenes and games. Blocks [18] leveraged modern mobile AR technology to enable synchronous colocated creation of persistent block structures. Loki [62] facilitated remote instruction with video, audio, and spatial capture, as well as MR presentation methods which allow users to explore both the local and remote environments.

XR has proven to be particularly applicable in remote scenarios, where the bandwidth of communication is otherwise lowered between participating users [25]. Prior research has generally adopted one of two approaches. One portion of prior research has proposed placing remote users in fully immersive VR worlds. Sra et al. [58, 59], for instance, presented several techniques for generating shared social virtual spaces procedurally. Alternatively, other systems have aimed to support remote collaboration in AR applications. Prior work has achieved this by sharing expressive avatars that can be placed in the physical space [46, 47], or by sharing portions of a user's environment via depth or video cameras, typically for the purposes of remote instruction [60, 62]. We focus on enabling developers to share aspects of a user's physical environment to construct shared contexts for collaboration, and aim to combine techniques from both prior AR and VR systems to do so. As previously noted, how users interact with their environment provides rich contextual information about their needs and actions, thus, we focus on enabling portions of the environment to be shared.

2.3 XR Prototyping and Development Tools

There is now a vast landscape of available tools for XR prototyping and development, each with its own intended function in the development pipeline, target users, technological basis, objectives, and considerations [2]. A common objective of XR prototyping and development tools is to lower the technical barrier to entry for creating XR experiences. Research tools like DART [33], ProtoAR [42], 360proto [41], and Pronto [29], for instance, facilitate the creation of low-fidelity XR prototypes without the need for programming. Other commercial tools, such as Unity [61], Unreal Engine [16], and A-Frame [1], and enabling technologies in research, like WorldKit [66] and the RoomAlive toolkit [24], abstract away low-level technical details to make the development of higher-fidelity applications easier. XSpace aims to fit into this second category.

Prior work has also developed tools to increase context-awareness and environmental sensing in a variety of application areas. Projects such as KinectFusion [23] and DepthLab [13] have significantly lowered the barrier to gathering and accessing environment geometry data for application developers in recent years. The Proximity Toolkit [34], for example, supplies developers with fine-grained proxemic information between people, devices, and objects in the environment. Likewise, Sousa et al.'s toolkit [54] eases prototyping with multiple commodity depth cameras which capture

user joint information. XSpace is similar to these tools in that it attempts to make environmental information easier for developers to practically use. XSpace aims to enable developers to create applications such as Sra et al.'s work [58, 59] which uses the environment geometry as a canvas for generating shared virtual worlds, or like Holoportation [46] and Loki [62] which present AR collaboration systems that leverage spatial capture to provide remote users with additional context for their interactions.

Aditionally, other tools have focused on supporting the creation of multi-user, multi-device experiences. Speicher et al.'s XD-AR [57] development framework, for instance, was designed to unify input and output across a diverse set of AR displays. XRDirector [40] uses a role-based approach to simulating XR scenes with multiple users. XRDirector also highlights important issues around spatial coordination that can occur between AR and VR users, further motivating the need for shared spatial context in collaboration.

XSpace bridges and extends the aforementioned streams of XR prototyping and development research, a combination that to our knowledge is currently under-explored. XSpace is the first toolkit that focuses on allowing AR developers to more easily leverage spatial capture data as contextual information in their applications to facilitate remote collaboration between users.

3 DESIGNING COLLABORATIVE SPACES

In this section, we present a set of three primary methods for creating shared AR spaces, distilled from a review of prior work, commercial applications, and developer tools. For each method, we describe relevant literature, motivating scenarios, and the interactions that each enable. We use this review to motivate the design of XSpace. An overview is available in Table 1.

3.1 Object-Centric Methods

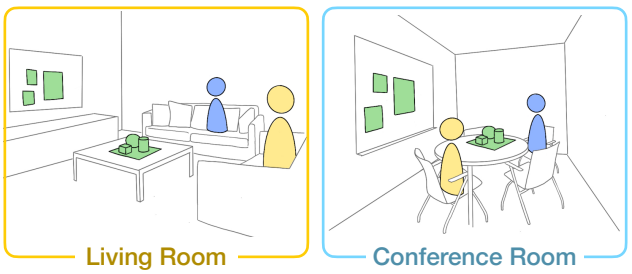

Fig. 2. Shared anchors scenario showing linked physical objects between a living room and a conference room. One user works from their living room with a coffee table, couch, and television, and another user works from a conference room with a table, chairs, and whiteboard. In AR, physical objects with similar functions could then serve as anchors for the same virtual content: avatars sit on chairs/ couches, a shared 3D model is placed on each table, and meeting notes are posted on the television/ whiteboard.

One method for creating a shared AR space is to center physical objects as 'anchors' or reference points for virtual content. Despite differences in geometry, room layout, and type of furniture, the physical objects in both environments have functional similarities that could be used to place and group virtual objects together. This scenario is shown in Figure 2.

This approach is used in two commercial AR applications Spatial [55] and Mesh [37], which enable distributed collaboration around a single shared, usually fixed entity that is manually placed by the user. For example, in Spatial, the user places a virtual 'wall' over a wall in their physical environment, and virtual content and avatars are then placed relative to this anchor. Room2Room

[47] also presents a similar idea, where seating affordances in each space are pre-specified, and the angle of a user's gaze is then redirected to account for slight differences in room layout. Prior research has also used physical landmarks as a way to re-map virtual reality spaces slightly to be flexible to new physical environments. For example, Congdon et al. present a technique for mapping two physical environments to each other based on key physical anchor points in order to create a shared VR space [11].

We support this concept in XSpace by allowing multiple physical landmarks in each space to serve as anchors for virtual content. For example, in Figure 2, the chairs, table, and wall are all acting as linked reference points for content. These anchor points create a mapping between the physical spaces. As in prior work, virtual objects could thus be placed relative to one or more anchor points in the space. If a virtual object is placed between the table and wall in one space, it should be placed between the table and wall in the other space, regardless of layout or distance. This introduces further challenges relating to redirecting user's gaze or movements continuously when the layout of the two spaces differs, which we discuss further and address in Section 4.2.1.

3.2 Perspective-Driven Methods

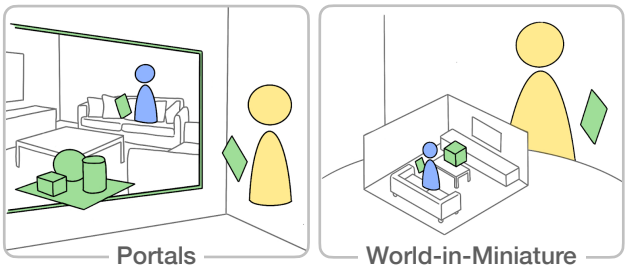

Fig. 3. Portals and world-in-miniature scenarios. Portals: Users can see the other space and share content through the portal, while keeping some content in their personal spaces. World-in-Miniature: Users can see and manipulate content in the miniature display of the remote space.

Creating a shared AR space can also be done by giving users the ability to have a direct perspective into another space. Portals are a popular way to do this. A variety of prior systems have used portal implementations to share context. For example, Photoportals implemented a variety of 2D and 3D portals to serve as representations of users, objects, and places [25]. Though, compared to VR, AR users cannot convincingly walk through a portal to be transported into another space, portals can still be used in AR to share virtual content and view how another user interacts with their environment, as shown in Figure 3. Portals also provide an opportunity for some AR content to remain private to each user.

Another method draws on prior work which uses a birds-eye view of a remote space for guidance. For example, Loki used a miniature live depth capture to display user context [62], and Stafford et al. used a miniature tabletop projection to guide users through a navigation task. [60]. Placing a miniaturized version of one space inside another allows users to share virtual content by placing it inside the miniaturized space, as shown in Figure 3. With only one space miniaturized, this is an asymmetric form of collaboration, which may be better suited to some tasks. However, it could also be made symmetric by providing each user with a miniaturized version of the other user's space. We used XSpace to explore these sharing modes individually and in combination.

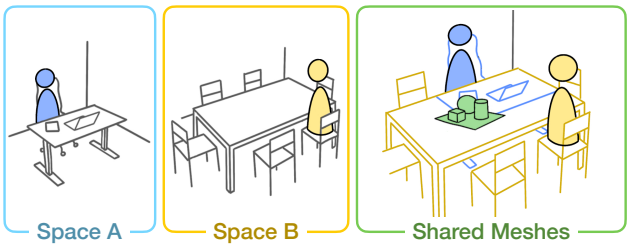

Fig. 4. Mesh crop and overlay scenario showing the geometry of a desk and dining table being merged together to create a shared tabletop extending the physical desk.

3.3 Mesh-Based Methods

Finally, a shared AR space could be created by directly sharing scanned environmental meshes between users. Scanned environmental meshes could be cropped to remove irrelevant context and then overlaid onto another user's space. This provides not only a shared coordinate plane for AR content, but also a visible spatial context for the other user's behavior. For example, one user's table could be overlaid onto another's desk to create a shared workspace, allowing each user to have context for the other's actions, as shown in Figure 4. Holoportation [46] achieved this effect by capturing one user's environment with multiple depth cameras, and displaying relevant objects to another user in AR.

Inspired by constructive solid geometry (CSG) operations used in many popular 3D modeling tools, we imagine that a shared space created in this way could be visualized as the intersection of two overlapped meshes. Additional operations could also be performed on the meshes before they are overlaid, for example, meshes could be scaled up or down to match the scale of another space if needed, as demonstrated by Sra et al. [59]. We experimented with these mesh-based operations for creating shared environments in XSpace.

4 XSPACE

To support the methods for creating shared AR spaces that we identified, we developed XSpace, a toolkit for creating spatially-aware AR applications for distributed collaboration. XSpace provides an infrastructure to turn single-user AR applications into ones that support distributed collaboration. It supports various compositions of the space alignment methods we presented earlier, allowing designers or developers to test these configurations using a set of complementary visual tools. XSpace is open-source and available at `https://github.com/HumanAILab/XSpace`. We implemented XSpace with the primary design goals of allowing for easy exploration of multiple space configuration options, and minimizing developer effort when designing and developing distributed AR applications. XSpace consists of three main components (Figure 5):

(1) **XSpace's developer toolkit** allows developers to add shared space configurations to existing AR applications.
(2) **XSpace's server** provides an architecture for multiple AR devices to share environmental information.
(3) **Visual design tools** allow developers or designers to simulate space configurations on real or synthetic environmental meshes.

In this section, we first give a walkthrough of creating an XSpace-enabled application, then we describe XSpace's system architecture.

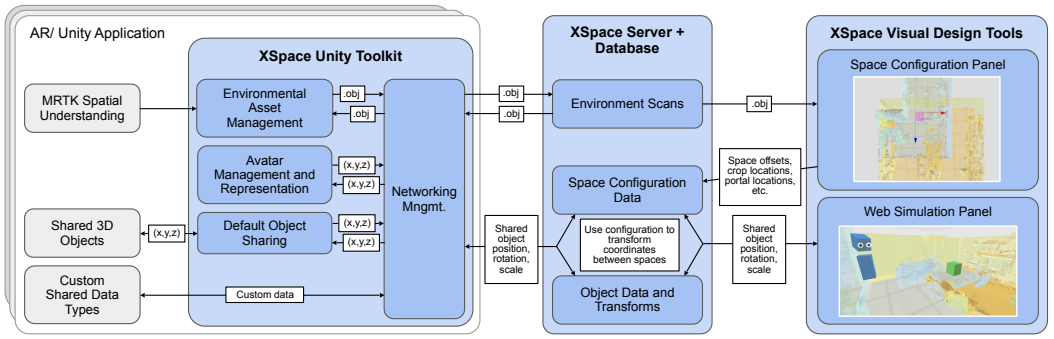

Fig. 5. XSpace system overview. XSpace has three primary components: *(1)* a Unity toolkit which allows AR applications to connect to this system as clients and provides common utilities; *(2)* a backend component which stores the space configuration and manages object's transformations. When a coordinate is accessed from the database, the server uses the saved configuration data to translate the coordinate into a local form suitable for each users physical environment; and *(3)* a web interface for configuring and simulating shared AR spaces using the four operations identified in our design space.

4.1 System Walkthrough

We base our system walkthrough in this section on the scenario of a group of developers who want to create a distributed co-working environment in AR. The end goal is to create an environment with a designated workspace for each user, and a designated 'break room' area for casual discussion. In this way, a user can naturally walk up to a colleagues desk to ask a question, or look into the 'break room' to see if anyone is available to grab a coffee and chat.

Create AR application. XSpace provides an infrastructure for adding distributed multi-user functionality to existing AR applications. In the case of this scenario, this may be a document viewing and sharing application that can be enhanced with XSpace to allow for distributed users. Developers first create a single-user version of the application as usual. There is no need to add external networking or avatar components. Developers should also make note of what AR content should be shared between users later on.

Preview space configurations. Next, in order to determine what configuration types they would like their application to support, developers can use XSpace's visual tools to preview a variety of configurations. XSpace provides example environmental meshes, or users can upload their own models. As described in Section 4.4, developers can directly configure a shared space by manipulating and cropping meshes, placing portals and miniaturized spaces, and creating shared anchors. In this scenario, the group plans to use two alignment methods. First, they will use the 'align objects' method to create a shared workspace for users. For example, the group may align one user's desk with another user's dining room table, so that their avatar appears to be sitting and working there, as in Figure 4. This can be achieved by drawing bounding boxes over the corresponding mesh areas. Next, the group will use the 'crop and overlay' method to create a shared 'break room' area for casual discussion. This can be achieved by overlapping the meshes on the shared area, and drawing its boundary.

Integrate XSpace with the application. We aim to make integrating XSpace components as simple as possible. First, Unity developers will add XSpace's main prefabs (the networking manager, environmental asset manager, and avatar manager) to the root of their scene. Any space configuration features (i.e., portals, cropped meshes) are managed by these central components. Next, developers will add the sharing script component to any hologram that needs to be shared between users. This script calls one of XSpace's functions to update the other users if the hologram's

transform changes. Developers can add their own parameters to send additional data between users. Finally, developers can deploy XSpace's server as provided for their application to connect to.

Fine-tune with visual authoring tools. On launch, the application will connect to the deployed server, then begin scanning the user's environment using MRTK's Spatial Understanding functionality [36]. Because this library aims to create a higher-fidelity and higher-quality scan than the default HoloLens scan, users need to deliberately walk around their environment and gaze at areas that they wish to include in the scan. Developers can then return to XSpace's visual tools to re-configure the shared space with the real-world scan data. Once saved, data about the chosen alignment (local origin offsets, shared space boundaries, portal locations, etc.) is sent to each device and used to mediate shared object transformations later. In this way, developers can quickly test various space configurations directly as their application runs.

4.2 XSpace Toolkit

XSpace's Unity Toolkit provides an interface for developers to create multi-user applications that make use of our mechanisms for creating shared spaces. The primary toolkit components are (1) networking management, (2) environmental asset management, and (3) avatar representation and management. Each of these components is implemented as a Unity prefab that can be used with minimal configuration. The networking management component initializes a connection to the shared database using the Unity3D-DDP-Client [7]. Next, the environmental asset management component initializes the scanning process using Microsoft's Mixed Reality Toolkit (MRTK) [36], formats scanned meshes into the .OBJ file format, and sends these files to the server. When a remote user joins the session, this component also imports that user's space from the database, reconstructs it into a mesh, and crops and displays the mesh according to the space configuration. Finally, the avatar representation and management component handles the display of remote user's avatars. Though these components can largely be used-as is, parts can be swapped out as needed. For example, a developer could choose to use a different avatar representation, which would only require slight modifications to XSpace's components.

Additionally, XSpace provides a component for sharing arbitrary virtual objects among users. This is provided as a script that can be attached to an existing GameObject or prefab in Unity. By default, this script shares the object's current transformation (position, rotation, and scale), along with a string to represent the model or prefab name. When the script is instantiated on one device, other devices can instantiate the same object and update its transformation accordingly. Developers can also add additional properties or data types to the shared object as needed. For example, a developer might want to share the current color of an object. We provide example functions for doing so that can be easily modified. While we specifically target the HoloLens with our toolkit, XSpace could be extended to support other devices with scanning capabilities, like mobile phones.

XSpace supports the shared space configuration methods that we identified in our earlier review. We describe the implementation of each below.

4.2.1 Shared Anchors. To calculate the position of virtual content relative to the shared anchors, we define the following mapping, motivated by the idea that users' relative position and orientation to the anchor in their local environment should be preserved in the remote environment. For one anchor pair, we use the following affine transformation to obtain remote positions and rotations of avatars and objects: $M = M_C^{-1} M_R$, where M_C is the world matrix of the anchor in the current space, and M_R is the world matrix of the anchor in the remote space.

For two anchor pairs, we also aim to preserve the relative movement between the anchors. For example, if a person walks from one anchor to another in their own space, they should also appear to do so in the remote space. To achieve this, we let the vector between the anchors act as the

world forward vector of the coordinate space, and reorient all positions accordingly. This vector's magnitude is also used to define a scaling factor between the two spaces.

When three or more anchor pairs are defined, we only consider the three anchors closest to the object we would like to position. We can then consider the three anchor points as creating a barycentric coordinate system. To translate an object's orientation, we first calculate directional vectors between the center of the anchor formed triangle and each of the anchors. Given the user's gaze vector, we calculate which directional vector is closest. Finally, we apply a rotational matrix representing a quaternion required to rotate the closest directional vector to the corresponding directional vector in the remote world to determine the remote gaze vector. More information about these calculations is provided in Appendix B.

4.2.2 Portals. To implement each portal's view, we place a virtual camera in the scene that mimics the viewpoint of the local user. The view from this camera is saved to a buffer, and then used to projectively texture the plane representing the portal from the user's point of view using custom vertex and fragment shaders. We can calculate the position and the rotation of the remote camera with the following affine transformation: $M_{P_C P_R} = M_{P_C}^{-1} R_{-180} M_{P_R}$. We determine when virtual objects should be passed between remote spaces by performing a portal-line intersection check with a line defined between the virtual object's current position and position in the previous frame. The same mapping defined for calculating the position and orientation of the remote camera can be used to determine the new position of virtual objects passed to a remote world.

4.2.3 World-In-Miniature. This effect is achieved using the object hierarchy system that scenes in Unity are organized by. Each avatar is added as a child object of their respective environmental mesh, while the miniature space is added as a child object of the larger space. Each avatar will thus be positioned and scaled relative to their space.

4.2.4 Mesh Crop and Overlay. The placement of the virtual avatars and objects between the two spaces are defined with an affine transformation calculated using the relative translation, rotation, and scaling between the two remote rooms in the configuration interface. Specifically, the mapping from room A to room B is defined as $M_{AB} = R_B^{-1} S_B^{-1} T_B^{-1} T_A S_A R_A$, where R_A, S_A, T_A and R_B, S_B, T_B define the rotation, translation, and scale of rooms A and B respectively. As an example, to obtain the position of a virtual object in room B in room A's local coordinates, the following calculation is performed: $P_A = M_{AB} P_B$.

Additionally, meshes can be cropped so that only relevant portions are included, or be sliced to produce two separate shared areas. Separate intersection or union operations can then be performed on individual slices. If this method is used, we calculate the position of remote avatars by first determining which slice the user is on. We then use a similar method as described above, substituting the transformation of the relevant mesh portion.

4.3 XSpace Server and Database API

In order for multiple users to connect to a collaborative session, XSpace includes a networking infrastructure that mediates object positions across various coordinate spaces. This is implemented as a MongoDB database, which is accessed using an API following Meteor's Distributed Data Protocol. For example, when one AR user moves to a new location in their environment, their local position is sent to the server. When the server receives this change, it uses information about how the shared space has been configured to translate this into a global coordinate. This change is then pushed to all other users in the global form, which each user translates to a location in their own environment. The server calculates the coordinate translation between environments according to the implementation in Section 4.2.

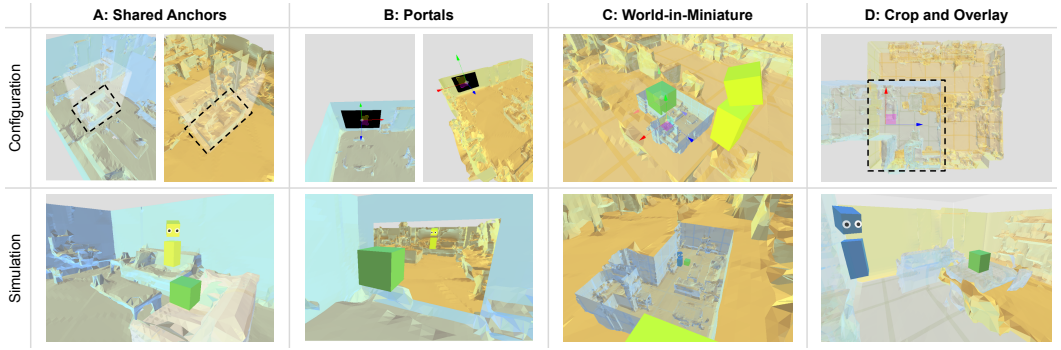

Fig. 6. An overview of XSpace's companion visual design tools. For each operation, we show how it is created in the configuration panel, and the result in the simulation panel. *(A)* Shared Anchors allows developers to designate physical objects in each space as an anchor for virtual content by drawing a bounding box around the object. *(B)* Portals can be placed in each space using transform controllers, and then act as a window into the other space. *(C)* World-in-Miniature allows users to place a miniature version of one space within another, and share virtual content by dropping it into the miniature space. *(D)* Mesh Crop and Overlay allow users to share portions of their environmental meshes to create a unified environment.

4.4 Visual Design Tools

XSpace provides a set of visual design tools in the form of a 3D web interface. This interface consists of a *configuration panel*, where alignment methods can be applied to scanned or synthetic environmental meshes by directly editing a 3D scene, and a *simulation panel*, where a first-person AR view of the resulting space is simulated in the browser. An overview is shown in Figure 6. Scanned environmental meshes are initially loaded into a scene in the configuration panel. Developers or designers can then modify the configuration by manipulating the meshes. This is done via traditional transform controls (to align meshes, place portals, and place miniature versions of the meshes), as well as simple click-and-drag controls (for slicing meshes and defining bounding boxes). The simulation panel is then used to visualize the result of a configuration from a first-person view. This has multiple uses. For example, a developer may use it to test their application with multiple synthetic meshes that can be imported into the scene. It can also allow web or VR users to join a collaborative session with AR users to test various configurations. This interface is implemented using Three.js [63] and Meteor [35].

4.4.1 Shared Anchors. To configure a space using shared anchors, users click and drag within the configuration panel to create bounding boxes around two corresponding objects within each mesh. A visualization of the bounding boxes is shown in Figure 6. After selecting these corresponding objects to act as a shared anchor for virtual content, an additional panel will then appear allowing users to set a local 'forward' vector for each object. This is included because some objects have an obvious 'front', i.e., a chair or a desk, so specifying this allows content to be placed relative to a specific part of the object. Upon confirming the orientation alignment, the objects will be highlighted as feedback indicating an object-pair has been defined.

We also implemented a second mechanism for creating a shared anchor which allows users to copy a physical object from one space to another in the case where there is not a relevant object in each space. The interface is similar to the above, except the user only specifies a bounding box for one object. That portion of the mesh is then copied into the second mesh, and the user can use transform controls to adjust its position. The object and its copy then act as a shared anchor for

virtual content, and the recipient of the copy will see the selected portion of the mesh overlaid onto their physical environment.

4.4.2 Portals. Users can add pairs of portals to the configuration scene via a toolbar. Portals can then be positioned, oriented, and scaled in each space using a transform controller. This is shown in Figure 6. When portal placements are confirmed, they will appear in the simulation interface and display a view of the remote space.

4.4.3 World-In-Miniature. To configure a world-in-miniature view, users can add a scaled-down copy of a mesh into the configuration scene. This will initially be placed into the center of the other mesh in the scene, and can be edited with a transform controller. From the larger space's view, the miniature space will appear with a small avatar inside. Virtual objects can be passed between the two spaces by intersecting them with the 'ceiling' of the miniature room, and will be scaled automatically to match the space they are contained within.

4.4.4 Mesh Crop and Overlay. Users can also crop and overlay meshes onto each other using CSG-inspired operations. In the configuration panel, environmental meshes can be positioned, oriented, and scaled using a transform controller. When the spaces are overlapped to produce an intersecting region, the user can then define the overlapping region as an intersection or union using a bounding box. If designated an intersection, the scanned room geometry of both rooms will be merged and shared within the bounded overlapping space. If designated a union, the the entire merged meshes will be shared between users, even past the bounding box.

5 EXAMPLE APPLICATIONS

Table 2. Example application scenario descriptions. (*indicates the application was considered conceptually but not implemented)

Scenario	Method	Description
Multiplayer game (Entertainment)	Crop and Overlay	A multiplayer snowball throwing game to highlight the use of shared environments to enrich collaborative game-play.
Collaborative furniture layout (Personal)	World-in-Miniature	Collaboratively design furniture layout from both a birds-eye and first person view of a space.
Shared work space (Professional)	Shared Anchors	An AR-enhanced co-working space, replicates an office space in the home
Remote user study platform* (conceptual)	World-in-Miniature, Portals (proposed)	An experiment platform for study administrators to observe remote users interacting with their environment.
Configurable remote classroom* (conceptual)	Crop and Overlay (proposed)	A teaching environment that enables instructors to create breakout rooms from scanned spaces.

In order to evaluate XSpace's coverage and flexibility, we select three common single-user XR applications to extend with XSpace to support multiple distributed users (see Table 2). Demonstration by example is a standard toolkit evaluation technique as discussed by Ledo et al. [26]. We consider this the most important type of evaluation for XSpace. A study with developers could be performed in the future to understand how XSpace would be utilized in the application development

process but to us this is a secondary concern. With the scenarios in Table 2, we aim to illustrate the utility and versatility of XSpace. Specifically, we hope to highlight the diverse use cases of our space configuration techniques. Each application implements a different method identified from our review. In the following, for each of our example applications, we describe our current implementation and highlight potential future additions. Finally, we also highlight the developer effort needed to construct one application using XSpace by comparing the lines of code needed to create single- and multi-user versions of the same application.

5.1 *Snowball Throw* (Multiplayer Game)

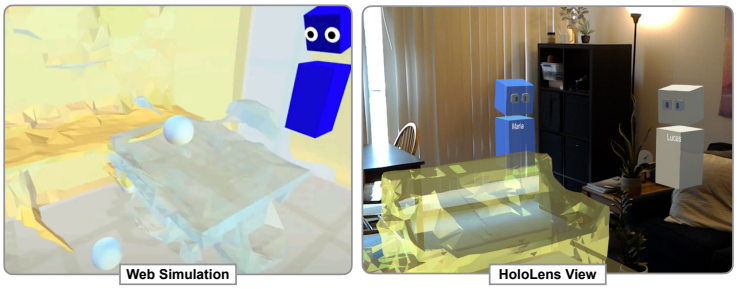

Fig. 7. *Snowball Throw's* AR application using the intersection operation. Left: *Snowball Throw*'s web application simulating AR by visualizing the user's spatial mesh in a web interface. Using the intersection operation, each user can see the other's space overlaid onto their own. Right: *Snowball Throw's* HoloLens application developed with XSpace's Unity toolkit. The yellow mesh shows the remote user's couch which has been merged with the local space. Remote users can then move behind the couch to dodge.

```
ThrowSnowball()
  XSpace.SendSnowball(position, direction)
  GameObject snowball = Instantiate(…)
  snowball.AddForce(…)
```

```
ReceiveSnowball()
  position = XSpace.Local(position)
  direction = XSpace.Local(direction)
  GameObject snowball = Instantiate(…)
  snowball.AddForce(…)
```

Fig. 8. A snippet of code highlighting how XSpace's functionality was integrated into *SnowballThrow*. On the left, when a snowball is thrown by the local user, an additional function is called to send its position and direction to XSpace's server. On the right, when a snowball is thrown by a remote user, XSpace's utilities are used to convert its position and direction to vectors in the local coordinate space, based on the current space configuration. Full sample code is provided in Appendix A.

In *Snowball Throw*, we explore how shared spatial contexts can change and enhance embodied game-play experiences. This example was modeled after typical multiplayer games: players can throw snowballs at other player's avatars which break apart on impact. We implemented this XSpace application using the mesh crop and overlay method, meaning parts of a remote player's mesh are overlaid on top of another user's physical environment. We designed the snowball projectiles to interact with both the player's local environment and the virtually overlaid remote environment. After the local and remote spaces are configured using XSpace's visual authoring tools, the player's environment becomes a tactile game environment. They can dodge their enemy's snowballs by hiding behind the physical furniture and walls in their room. They can additionally leverage the shared virtual furniture and walls as coverage.

As XSpace enables the shared environment to be reconfigured as needed, the same physical environment can be adapted into a range of game-worlds through intersections with different

remote spaces. Even with just two spatial meshes, by applying transformations (e.g., scaling one space) of different sorts, we can generate a vast array of different gameplay experiences.

We implemented *Snowball Throw* both on the HoloLens using XSpace's Unity Toolkit, and as a 3D web application. The general development process outlined in Section 4.1 was followed to implement the HoloLens application. For this application, we added one custom shared data type into the system to represent the snowball throwing functionality. When a snowball is thrown, we share its initial position and throwing direction. Its trajectory is then calculated using the physics engine on each device (see Figure 8 for an overview). Adding this functionality required modifying around 20 lines of code, compared to the 125 lines required to develop the single-user version of *Snowball Throw*. More information about this implementation is provided in Appendix A.

Though we implemented *Snowball Throw* using the mesh crop and overlay method, other designs could also lead to different and creative game-play. For example, using multiple portals to link spaces and throw snowballs through would create a much more challenging game.

5.2 *Room Design* (Collaborative Furniture Layout)

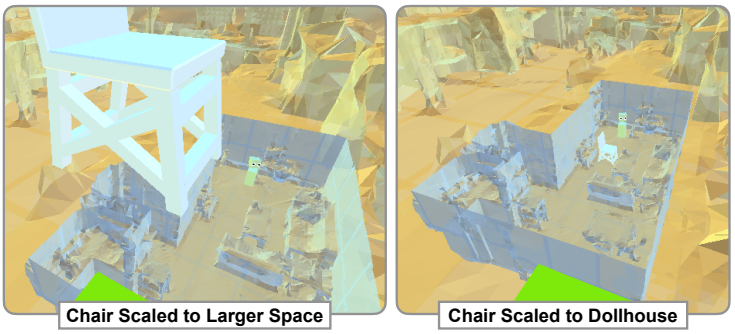

Fig. 9. *Room Design's* web implementation simulating AR by visualizing the user's spatial mesh, using the world-in-miniature operation. Left: a user moving a chair into the miniature space. Right: the scale of the chair is automatically adjusted when it enters the miniature space.

In *Room Design*, we explore the use of XSpace to support multi-scale and multi-perspective interactions, specifically in the context of furniture layout design. In the application we implemented, users can each place various furniture items in their space, and edit their positions by clicking and dragging. We enable users to interact via XSpace's world-in-miniature operation. We place the environment of one user as a miniature model on top of a table in the room of the other, allowing for a top-down view of the furniture and room layout. When the user passes an object into the miniature space, the scale of the object is immediately adjusted so that it appears small. However, for the user inside of the miniature space, the object appears at room scale. This enables collaborating users to collectively have both a local- and global-level perspective of their design, which Ibayashi et al. [22] has highlighted previously as highly beneficial. *Room Design* was implemented as a 3D web simulation of AR using a three.js [63] version of XSpace's toolkit. This was done as a separate webpage that interfaces with XSpace's API in the same way a Unity application would. Previously scanned environmental meshes from a HoloLens were saved and used as example meshes for this demonstration. This application can also pair well with the portal operation. With both of these techniques, users could have both a top-down view and first-person window view into the other persons space, which has benefits to this task.

5.3 *Office Space* (Shared Work Space)

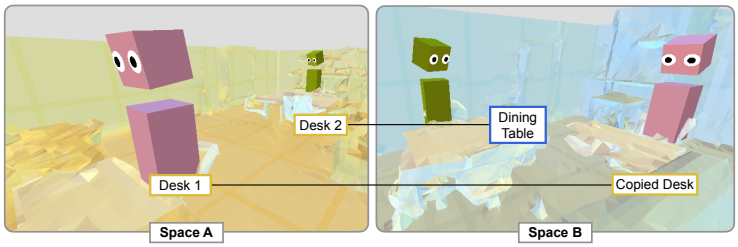

Fig. 10. *Office Space's* web implementation simulating AR by visualizing the user's spatial mesh, using the shared anchors operations. Here, there are two pairs of shared anchors: One desk linked to a dining table, and another desk copied to a remote space. The resulting avatar locations in each space is shown here. Each user is at their own physical desk, but their avatar is thus placed contextually in the other's space.

In *Office Space*, we demonstrate how XSpace can be used to share spatial contexts in order to facilitate serendipitous interactions that can arise out of being co-present with others in a physical environment. We explore this idea in the context of remote work. One aspect of being in a physical office space that is difficult to replicate with current video conferencing tools is the ability to glance at a co-worker to see if they are busy, and walk up to chat with them when an opportunity arises. We enable this experience in our *Office Space* application.

Office Space leverages XSpace's shared anchors operation in our current implementation. In the example we present in Figure 10, we paired one user's desk with another user's dining table. As a result, both users appear working contextually within each other's environments. Within the application, users can also share virtual objects as 'status indicators' of their work progress. For example, users can hold a coffee mug to signify that they are taking a break and open to conversation. *Office Space* was implemented as a 3D web simulation of AR using a three.js [63] version of XSpace's toolkit, using the same method as our *Room Design* application.

Though we implemented Office Space with the shared anchors method, this can also pair well with the mesh crop and overlay method. With this technique, portions of each user's space could be separated and designated as a 'break room' or 'conference room' while still maintaining the general office space. This opens a future opportunity to consider the materials of shared space boundaries. While we implement boundaries as a translucent border around a space, this could be varied in the future by privacy needs. For example, the 'conference room' space could have a completely opaque, sound-insulated barrier, while the 'break room' space could have a semi-opaque barrier.

5.4 Further Examples

In addition to the implemented applications above and their potential extensions, we also present two additional conceptual use cases that could benefit from shared space creation.

First, XSpace can be used to create applications for performing remote user studies that require observing a user interacting with the environment. The world-in-miniature operation could be used for a top-down perspective, while multiple portals act as windows into various locations in the user's space. In this application however, the study administrator may not want to be visible to the participant, necessitating the use of a one-way portal. Though we did not implement this, XSpace could easily be extended to support various types of sharing. To use the metaphor of a physical window, the boundary between spaces could act as a completely transparent window, a one-way mirror, frosted or translucent glass, or as an opaque wall. All of these options may be

useful in different settings, and in the future allowing users to customize this on the fly would add an important additional layer of control over collaborative spaces.

XSpace could also be used to create remote teaching applications. For example, the mesh crop and overlay method, which allows one space to be cropped into multiple shared areas, could be used to create multiple breakout rooms for students to work on a task together. These rooms could be reconfigured as needed to shuffle student groups. An instructor would then be able to walk around between 'rooms' in their space as normal to monitor students. This could be joined with similar techniques presented in Loki [62] and Slice of Light [65] to create powerful and contextual teaching environments.

6 DISCUSSION

6.1 Technical Reflection

Developing distributed, multi-user AR applications is a complex task requiring a significant amount of developer effort and many iterations. We aim to begin to address this with XSpace, demonstrating that multiple methods for creating distributed, shared AR spaces can be integrated into one toolkit. XSpace allows developers to add shared space configurations to existing AR applications, and simulate configurations through a set of visual design tools. Here, we first thematically analyze XSpace using Olsen's framework [45], then we discuss limitations and future work.

6.1.1 Problem Not Previously Solved. Prior work has envisioned a number of innovative ways to create distributed shared spaces [46, 58, 59, 62]. XSpace brings these methods together into a single toolkit, enabling a technical exploration of how different types of applications could be constructed. Additionally, while a number of AR prototyping tools and development toolkits exist [13, 23, 29, 33, 40–42, 54, 56], tool support for creating collaborative AR applications has previously been limited to specific scenarios and collaborative settings like meeting rooms, which is only one of the possible configurations covered by XSpace's design space, as discussed in Section 3.

6.1.2 Generality. We presented five example applications to illustrate the versatility and generality of XSpace, and its methods for creating shared spaces. Additionally, XSpace has been designed based off of a wide range of existing AR applications, which it aims to support. One limitation of XSpace is that it centers around specifying shared spaces based on real or synthetic environmental meshes. Future work should investigate how shared spaces can be specified based on parameterized environments to support more dynamic collaborative settings. This could be similar to how Unity MARS works for single-user AR applications: developers can prototype an application by placing elements relative to a synthetic environmental mesh, which is then parameterized by its surfaces, walls, and other objects so that the AR experience can be generalized to other spaces [64]. Developing such adaptive AR systems that can extend to arbitrary combinations of diverse spaces has been the subject of ongoing research [9, 30] and future research could investigate how to best incorporate such techniques into a collaborative system like XSpace.

6.1.3 Reduce Solution Viscocity. Because XSpace as a toolkit was specifically designed to support customization of AR applications to enable different modes of collaboration, it is likely less effort than the current developer solution of implementing each space configuration manually. XSpace's current strength in this area is in its *flexibility* [45], meaning the ability to make rapid design changes, as showcased in its visual design tools for quickly configuring collaborative spaces. However, as XSpace's visual design tools primarily use a single rendered 3D view with a click-and-drag interface to configure a shared space, this could be refined in the future, perhaps through additional views and snapping tools such as those demonstrated in SnapToReality [43], which would increase XSpace's

expressive match, the expression of design choices in a toolkit per Olsen [45]. XSpace provides a useful foundation for future research in this direction.

6.1.4 Empowering New Design Participants. We aimed to develop XSpace's visual controls so that they would be familiar to those already working in 3D design. XSpace uses a mix of three-axis transform controls (as found in common 3D modeling software), as well as some custom drag-and-drop controls for creating bounding boxes or slices in environmental meshes. However, the user experience could further be improved so that it is more efficient and approachable to a wider range of backgrounds. Although XSpace lowers the threshold for developing complex distributed AR applications which typically require a high degree of skill to create, XSpace still requires its users to have a knowledge of Unity development, which is specialized. XSpace's visual design tools could be integrated with aspects of other AR prototyping and design tools to allow non-technical designers to create distributed AR applications. For example, Pronto [29] or ProtoAR's [42] rapid prototyping approaches could be leveraged for quickly adding AR content into XSpace without code. Code-free methods for specifying the behavior of objects in shared spaces has been the focus of systems like XRDirector [40] and Rapido [28], and future work could investigate how to best enable visual authoring of dynamic objects to form an integral part of the collaboration where Blocks [18] provides a starting point.

6.1.5 Power in Combination. XSpace demonstrates that multiple methods for creating distributed, shared AR spaces can be integrated into one toolkit. We first distill a set of key methods from prior work. Then, in our example applications, we present how these methods can support common AR collaboration scenarios.

The methods we implement in XSpace can be used independently which is sufficient to recreate some prior collaborative applications, but are more powerful in combination and able to support more complex, new scenarios. For example, while in *Room Design* we currently use one world-in-miniature model to allow someone to have a top-down view into another environment, this could be combined with other methods. A mesh overlay could be added in part of the space so that users could walk around the newly-designed space first hand, tweaking furniture placement on a more precise scale.

Given that XSpace allows users to modify the construction of a virtual environment, it is possible for spaces to be created that are not entirely functional. Some of these spaces may be unnatural, hard to understand, or redundant, but ultimately still usable. For example, if a portal is placed inside an area where two meshes have been overlaid with each other, the portal will essentially look in on itself. This does not cause technical issues, but it potentially does not function in the manner intended by the user. Another example of this can occur when cropping a mesh into multiple pieces and creating multiple overlays with another space. If the user moves from one slice to another, to the remote user, they may appear to teleport across the room. Furthermore, if spaces are slightly misaligned, then a person's avatar could appear to be walking through furniture or walls in another user's space, potentially causing disorientation and breaking the illusion of presence.

Other combinations may yield non-functional spaces. For example, a mesh could be cropped into two parts, then overlapped with itself, causing a person to essentially be at two coordinates at once. Another example of this type of issue can occur when remapping some coordinate spaces using the shared anchors operation. If three objects in a triangle in one space are mapped to three objects in a line in another space, this results in a conflict. XSpace enabled these explorations of creating shared virtual space, and adding checks on the meshes and extending our visual tools would allow us to detect potential issues and guide users to mitigate them.

6.1.6 Can It Scale Up? While methods like the portals, world-in-miniature, or shared anchors operations scale well to groups of users (we successfully tested with up to 5), mesh-based operations may become increasingly difficult to interpret if more than two users' environmental meshes are layered on top of each other. One direction for future work would be to study user's perceptions of such spaces. It is possible that differences in spatial reasoning or understanding between users may cause coordination issues or confusion in these cases. Additional collaboration tools may mitigate such issues. For example, gaze indicators or shared annotations [56, 62] have previously been used in social VR situations to assist with coordination and perception

6.2 Limitations

XSpace aims to integrate a variety of methods for creating shared spaces into one toolkit. Aside from the opportunities for future work expanding XSpace's functionality previously mentioned in the following section, we recognize that there are limitations to our work. While we evaluate XSpace's expressivity and coverage of previously built applications in this work, and highlighted how XSpace could reduce developer effort through analyzing lines of code, we have not yet evaluated how XSpace may be used efficiently in design and development practice. Namely, XSpace's current visual authoring tools for designing and simulating various space configurations could be refined further, as mentioned previously in this section. Additionally, our approach in general of using a web-based set of visual design tools to configure and preview spaces could be expanded in the future to support other methods of doing so, for example, allowing designers to configure shared 3D spaces directly in AR or VR.

7 FUTURE WORK

In this section, we discuss considerations for using XSpace in practice relating to the development workflow, end user experience, and user privacy, as well as ways of extending XSpace in the future.

7.1 Automation vs. End-User Customization

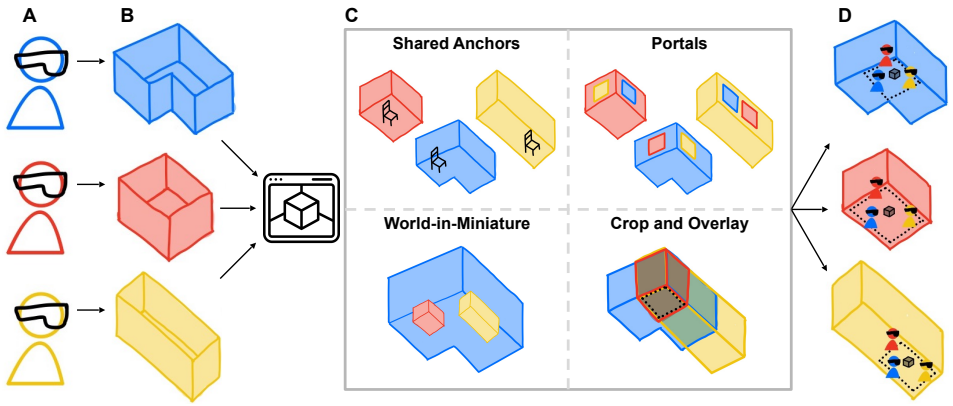

Fig. 11. Future XSpace end-user workflow concept. *(A)* A group of remote users, each with their own device, opens an AR application that was made with XSpace. *(B)* Each user's device scans the environment. After processing on device, the scans are then sent to a shared server. *(C)* XSpace's web interface imports scans from the server once they are available. Users can visit the web configuration UI to choose how various alignment primitives will be applied to their spaces. Once finalized, relevant alignment data (i.e., local origin offsets, space boundaries, portal locations, etc.) is sent to the server. *(D)* Finally, users can start using the intended AR application.

One important direction for future work is in exploring how the shared space configuration process in XSpace can be made available to end users. Currently, XSpace allows developers to customize shared spaces for their needs. We developed a manual creation method to fully explore possible configurations and potentially invalid alignments. One potential method would be to repurpose XSpace's visual design tools directly as an interface for end-user customization. This has the benefit of allowing users a fine-grained level of control over their collaborative spaces and what they want to share, with still benefiting from developer recommendations and having the ability to adjust those to their exact scenarios and needs. This future workflow is shown in Figure 11.

From a user experience perspective, a completely automated system or semi-automated system which takes user's scanned meshes as input and provides multiple potential spatial configurations as output could be beneficial. Prior work has explored re-mapping VR spaces to fit various physical environments [30], and similar techniques could potentially be applied to partially automate XSpace's shared anchor and intersection techniques. However, we suspect that the optimal configuration for a shared space will vary widely depending on the application and task at hand, so providing users or developers some level of control is crucial.

A related topic is exploring how shared spaces can be reconfigured on-the-fly. One potential method could leverage the semantics of physical objects in the space for direct manipulation of the shared space affordances (e.g., doors, lights, curtains, furniture arrangements). For instance, whether or not a room is shared could be linked to the state of the room's door. An open door could indicate the user is open to peers accessing their space remotely, while a closed door indicates the opposite. We could leverage similar environment state detection mechanisms as in SpaceState [14] to achieve this. With XSpace, we can furthermore base decisions regarding how the shared space should be constructed on the collective state of linked environments. For instance, instead of basing whether a space should be shared based on the state of one user's door, we could require both users to have their doors open before any sharing occurs.

7.2 Privacy and Safety

XSpace shares a scanned mesh of a users space with their collaborators. Though this mesh is untextured in our implementation, this is still revealing and can present a concern for users. Certain spaces in the home are considered private (e.g., a bedroom), and even for those that are more public like a living room, people may not wish to share this information [49]. At a coarse level, XSpace's Unity toolkit does not use previously scanned meshes of a space, and instead creates a new scan each time, allowing users to manually keep areas of their home out of the scan by not pointing the camera to those places. This could be expanded in the future to allow users to edit the scanned mesh on device before sharing it, for example, through cropping operations or allowing the user to 'blur' certain areas to make them lower fidelity.

In many cases, the extent to which various aspects of a collaboration or multi-user environment need to be private may differ. For instance, in a teaching setting, the instructor may at times need to have one-on-one conversation with a student. Similarly, while a user may generally be open to having their peers enter their living-room environment, they may have select documents in the room they would prefer to be confidential. Ruth et al.'s threat model for secure multi-user AR applications [51] can provide a starting point for future work to explore methods of providing users with more fine-grained control over the mesh sharing procedure. Similar to Ruth et al.'s ghosting mechanism where it was applied to hide sensitive documents from remote viewers, one approach we envision is to enable developers or users to control the materiality of their shared mesh (e.g., setting parts of the scanned space as fully transparent, semi-transparent, shared one way, or fully opaque). For instance, a user may prefer to set more private regions of their space as fully transparent so that it would not be visible, or block sharing of that area altogether.

7.3 XSpace as a Future Research Platform

Finally, we believe that XSpace can be a useful platform for researching distributed AR collaboration. There is a large amount of CSCW work that can be re-evaluated in this setting. For example, how is workspace awareness best facilitated? How do the various operations presented in this work affect collaboration and communication? How can we integrate existing theories from collaborative work into XSpace to make it more effective? Moreover, XSpace currently focuses on expanding the notion of 'space' in synchronous distributed collaboration, where there may be a higher motivation for increasing co-presence between users. However, asynchronous collaboration is also an interesting area for future work. MAVRC [10] explored the challenges and design considerations for multimodal asynchronous collaboration in VR. Blocks [18] presented techniques for asynchronous AR collaboration. XRDirector [40] studied collaborative design tasks with some users in VR and others in AR. These techniques could be adapted and combined with our techniques in XSpace for creating shared virtual spaces, perhaps in the form of a visualization in a shared virtual space to show the temporal dimension of recent changes. Similar to Loki [62], XSpace could be extended with functionality to support users in pre-recording interactions with their environment to be played remotely. We can envision pre-recordings acting as a contextually situated reminder for the local user. They could also potentially serve as space-specific instructions. We believe the flexibility of our approach facilitates rapid construction of shared environments for study purposes.

8 CONCLUSION

In this paper, we have presented XSpace, a toolkit for creating spatially-aware AR applications for distributed collaboration. XSpace supports a variety of methods for creating shared spaces, which we developed through a review of prior work. We also presented a set of example applications to illustrate how XSpace can enhance promising AR application scenarios. Overall, we demonstrate the potential of further involving environmental context in distributed AR collaboration. We identify important challenges for future work, including improving privacy techniques, automating shared space creation, and studying collaborative behaviors in this space. Overall, XSpace is an important step towards creating more immersive and effective distributed collaborative AR experiences by making it easier to leverage users' local environments.

ACKNOWLEDGMENTS

We thank our reviewers for their time and feedback. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1841052. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not necessarily reflect the views of the National Science Foundation.

A CODE SAMPLE

Here we provide more information about how XSpace was used to implement *Snowball Throw*, the demo application described in Section 5.1. Two primary components are modified: one for when a snowball is thrown locally and is sent to XSpace's server, and one for when a snowball is thrown by a remote user. Comments have been added to indicate which code has been written by the developer, and which code is provided by XSpace.

First, when a snowball is thrown locally, an additional function is called to send it's position and direction to XSpace's server. The developer only needs to add this function to their existing code (see Figure 12).

```
private void ThrowSnowball() {
    Transform cameraTransform = CameraCache.Main.transform;

    // Developer: Make a new object that is 1m away in direction of gaze
    var direction = cameraTransform.forward;
    var origin = cameraTransform.position;
    var position = origin + direction * 1.0f;

    // XSpace: Send snowball to server
    XSpace.SendSnowball(position, direction);

    // Developer: Create local snowball instance
    GameObject snowballInstance = Instantiate(snowballPrefab, position,
        Quaternion.identity);
    Vector3 throwDirection = new Vector3(direction.x, direction.y + 0.3f,
        direction.z);
    snowballInstance.GetComponent<Rigidbody>().AddForce(throwDirection * 400);

    // Developer: Destroy snowball instance
    Destroy(snowballInstance, 20);
}
```

Fig. 12. When a snowball is thrown locally, an additional function is called to send it's position and direction to XSpace's server.

This 'SendSnowball' function first formats the data, then pushes it into a database called 'projectiles'. This code was provided by XSpace, and modified by the developer to include the 'directionArr' property (see Figure 13).

```
    public void SendSnowball(Vector3 pos, Vector3 dir) {
        // XSpace: Format data
        JSONObject positionArr = JSONObject.Create(JSONObject.Type.ARRAY);
        positionArr.Add(-pos.x);
        positionArr.Add(pos.y);
        positionArr.Add(pos.z);

        // Developer: Add custom property
        JSONObject directionArr = JSONObject.Create(JSONObject.Type.ARRAY);
        directionArr.Add(-dir.x);
        directionArr.Add(dir.y);
        directionArr.Add(dir.z);

        // XSpace: Add snowball to database
        MethodCall methodCall = ddpConnection.Call("projectiles.insert",
            JSONObject.CreateStringObject(sysStateMulti.ourSpace),
            positionArr,
            directionArr);
    }
```

Fig. 13. The 'SendSnowball' function first formats the data, then pushes it into a database called 'projectiles'. This demonstrates how developers can share custom data types with XSpace.

For remote users, when new data is added to the 'projectiles' database, the following function is called. Only the final two lines were modified to instantiate the snowball object. The remaining calculations are provided by XSpace's Unity toolkit. The developer only needs to add the final two lines of code (see Figure 14).

```
public void ThrowSnowball(JSONObject position, JSONObject direction) {

    // XSpace: Format data
    if (position == null) return;
    Vector3 originalPos = new Vector3(...);
    if (direction == null) return;
    Vector3 originalDir = new Vector3(...);


    // XSpace: Transform from world position to local position (provided by toolkit)
    Vector3 newPos;
    if (sysStateMulti.ourSpace == "A")
    {
        newPos = originalPos + (this.gameObject.GetComponent<RoomHandler>().RoomBPos -
            this.gameObject.GetComponent<RoomHandler>().RoomAPos);
    }
    else
    {
        newPos = originalPos + (this.gameObject.GetComponent<RoomHandler>().RoomAPos -
            this.gameObject.GetComponent<RoomHandler>().RoomBPos);
    }

    // Developer: Code modified to create a snowball
    Vector3 newDir = originalDir;
    GameObject snowball = Instantiate(snowballPrefab, newPos, Quaternion.identity);
    snowball.GetComponent<Rigidbody>().AddForce(newDir * 400);
}
```

Fig. 14. When remote users receive a Snowball, XSpace transforms the given coordinates into its local position.

B IMPLEMENTATION DETAILS

Here we provide more information about our implementation of the shared anchors operation, specifically how coordinates are translated from one space to another.

For two anchor pairs, We let the vector between the anchors act as the world forward vector of the coordinate space, and reorient all positions accordingly. This vector's magnitude is also used to define a scaling factor between the two spaces.

Let T_{A_a}, T_{A_b} define the positions of two anchor objects in the local space, T_{B_a}, T_{B_b} define the corresponding anchor objects in the remote space, and $T_{B_{user}}$ define the position of the remote user. To determine the position of the remote user in the local environment, denoted $T'_{B_{user}}$, we perform the following computation:

$$T'_{B_{user}} = \|T_{A_b} - T_{A_a}\| R_{BA} \frac{T_{B_{user}} - T_{B_a}}{\|T_{B_b} - T_{B_a}\|} + T_{A_a}$$

Where R_{BA} defines a transformation matrix that rotates from $T_{B_b} - T_{B_a}$ to $T_{A_b} - T_{A_a}$. R_{BA} also defines the mapping of the remote avatar's rotation to the local space.

When three or more defined pairs are defined, we only consider the three anchors closest to the object we would like to position. We can then consider the three anchor points as creating a barycentric coordinate system. To translate an object's orientation, we first calculate directional vectors between the center of the anchor formed triangle and each of the anchors.

Let T_{A_a}, T_{A_b}, T_{A_c} define the positions of two anchor objects in the local space, T_{B_a}, T_{B_b}, T_{B_c} define the corresponding anchor objects in the remote space, and $T_{B_{user}}$ define the position of the remote user. To determine the position of the remote user in the local environment, denoted $T'_{B_{user}}$, we perform the following computation:

$$T'_{B_{user}} = wT_{A_a} + uT_{A_b} + vT_{A_c}$$

where $w = \frac{\|(T_{B_b} - T_{B_{user}}) \times (T_{B_c} - T_{B_{user}})\|}{\|(T_{B_b} - T_{B_a}) \times (T_{B_c} - T_{B_a})\|}, u = \frac{\|(T_{B_c} - T_{B_{user}}) \times (T_{B_a} - T_{B_{user}})\|}{\|(T_{B_b} - T_{B_a}) \times (T_{B_c} - T_{B_a})\|}, v = \frac{\|(T_{B_a} - T_{B_{user}}) \times (T_{B_b} - T_{B_{user}})\|}{\|(T_{B_b} - T_{B_a}) \times (T_{B_c} - T_{B_a})\|}$.

Given the user's gaze vector, we calculate which directional vector is closest. Finally, we apply a rotational matrix representing a quaternion required to rotate the closest directional vector to the corresponding directional vector in the remote world to determine the remote gaze vector.

Let $G_{B_{user}}$ define the direction of the remote user's gaze. Let $T_{A_{ctr}} = \frac{1}{3}(T_{A_a} + T_{A_b} + T_{A_c})$ and $T_{B_{ctr}} = \frac{1}{3}(T_{B_a} + T_{B_b} + T_{B_c})$. We compute the remote user's gaze in the local environment as follows:

$$G'_{B_{user}} = R_{AB_{max}} G'_{B_{user}}$$

Where $R_{AB_{max}}$ is a transformation matrix defined as follows:

$$R_{AB_{max}} = \begin{cases} \text{rotation from } (T_{B_a} - T_{B_{ctr}}) \text{ to } (T_{A_a} - T_{A_{ctr}}) & \text{if } G_{B_{user}} \cdot (T_{B_a} - T_{B_{ctr}}) \text{ is greatest} \\ \text{rotation from } (T_{B_b} - T_{B_{ctr}}) \text{ to } (T_{A_b} - T_{A_{ctr}}) & \text{if } G_{B_{user}} \cdot (T_{B_a} - T_{B_{ctr}}) \text{ is greatest} \\ \text{rotation from } (T_{B_c} - T_{B_{ctr}}) \text{ to } (T_{A_c} - T_{A_{ctr}}) & \text{if } G_{B_{user}} \cdot (T_{B_c} - T_{B_{ctr}}) \text{ is greatest} \end{cases}$$

REFERENCES

[1] A-Frame. 2021. A-Frame. https://aframe.io/

[2] Narges Ashtari, Andrea Bunt, Joanna McGrenere, Michael Nebeling, and Parmit K. Chilana. 2020. Creating Augmented and Virtual Reality Applications: Current Practices, Challenges, and Opportunities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376722

[3] Steve Benford, Chris Brown, Gail Reynard, and Chris Greenhalgh. 1996. Shared Spaces: Transportation, Artificiality, and Spatiality. In *Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work* (Boston, Massachusetts, USA) *(CSCW '96)*. Association for Computing Machinery, New York, NY, USA, 77–86. https://doi.org/10.1145/240080.240196

[4] Steve Benford, Chris Greenhalgh, Gail Reynard, Chris Brown, and Boriana Koleva. 1998. Understanding and Constructing Shared Spaces with Mixed-Reality Boundaries. *ACM Trans. Comput.-Hum. Interact.* 5, 3 (Sept. 1998), 185–223. https://doi.org/10.1145/292834.292836

[5] Steve Benford, Chris Greenhalgh, Tom Rodden, and James Pycock. 2001. Collaborative Virtual Environments. *Commun. ACM* 44, 7 (July 2001), 79–85. https://doi.org/10.1145/379300.379322

[6] Hrvoje Benko, Ricardo Jota, and Andrew Wilson. 2012. MirageTable: Freehand Interaction on a Projected Augmented Reality Tabletop. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI '12)*. Association for Computing Machinery, New York, NY, USA, 199–208. https://doi.org/10.1145/2207676.2207704

[7] Vincent Cantin. 2016. Unity3D-DDP-Client. https://github.com/green-coder/unity3d-ddp-client.

[8] 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 (2011), 341–377. https://doi.org/10.1007/s11042-010-0660-6

[9] Yifei Cheng, Yukang Yan, Xin Yi, Yuanchun Shi, and David Lindlbauer. 2021. SemanticAdapt: Optimization-based Adaptation of Mixed Reality Layouts Leveraging Virtual-Physical Semantic Connections. In *UIST '21: The 34th Annual ACM Symposium on User Interface Software and Technology, Virtual Event, USA, October 10-14, 2021*, Jeffrey Nichols, Ranjitha Kumar, and Michael Nebeling (Eds.). ACM, 282–297. https://doi.org/10.1145/3472749.3474750

[10] Kevin Chow, Caitlin Coyiuto, Cuong Nguyen, and Dongwook Yoon. 2019. Challenges and Design Considerations for Multimodal Asynchronous Collaboration in VR. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (2019), 1–24. https://doi.org/10.1145/3359142

[11] Ben J Congdon, Tuanfeng Wang, and Anthony Steed. 2018. Merging Environments for Shared Spaces in Mixed Reality. In *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*. 1–8. https://doi.org/10.1145/3281505.3281544

[12] Paul Dourish and Victoria Bellotti. 1992. Awareness and Coordination in Shared Workspaces. In *Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work* (Toronto, Ontario, Canada) *(CSCW '92)*. Association for Computing Machinery, New York, NY, USA, 107–114. https://doi.org/10.1145/143457.143468

[13] Ruofei Du, Eric Turner, Maksym Dzitsiuk, Luca Prasso, Ivo Duarte, Jason Dourgarian, Joao Afonso, Jose Pascoal, Josh Gladstone, Nuno Cruces, Shahram Izadi, Adarsh Kowdle, Konstantine Tsotsos, and David Kim. 2020. DepthLab: Real-Time 3D Interaction with Depth Maps for Mobile Augmented Reality. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 829–843. https://doi.org/10.1145/3379337.3415881

[14] Andreas Fender and Jörg Müller. 2019. SpaceState: Ad-Hoc Definition and Recognition of Hierarchical Room States for Smart Environments. In *Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces* (Daejeon, Republic of Korea) *(ISS '19)*. Association for Computing Machinery, New York, NY, USA, 303–314. https://doi.org/10.1145/3343055.3359715

[15] Andreas Rene Fender, Hrvoje Benko, and Andy Wilson. 2017. MeetAlive: Room-scale Omni-Directional Display System for Multi-User Content and Control Sharing. In *Proceedings of the 2017 ACM international conference on interactive surfaces and spaces*. 106–115. https://doi.org/10.1145/3132272.3134117

[16] Epic Games. 2021. Unreal Engine. https://www.unrealengine.com/

[17] Terrell Glenn, Ananya Ipsita, Caleb Carithers, Kylie Peppler, and Karthik Ramani. 2020. StoryMakAR: Bringing Stories to Life with an Augmented Reality & Physical Prototyping Toolkit for Youth. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–14. https://doi.org/10.1145/3313831.3376790

[18] Anhong Guo, Ilter Canberk, Hannah Murphy, Andrés Monroy-Hernández, and Rajan Vaish. 2019. Blocks: Collaborative and Persistent Augmented Reality Experiences. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 3, Article 83 (Sept. 2019), 24 pages. https://doi.org/10.1145/3351241

[19] Carl Gutwin and Saul Greenberg. 1996. Workspace Awareness for Groupware. In *Conference Companion on Human Factors in Computing Systems* (Vancouver, British Columbia, Canada) *(CHI '96)*. Association for Computing Machinery, New York, NY, USA, 208–209. https://doi.org/10.1145/257089.257284

[20] Jeremy Hartmann, Christian Holz, Eyal Ofek, and Andrew D. Wilson. 2019. RealityCheck: Blending Virtual Environments with Situated Physical Reality. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300577

[21] Christian Heath and Paul Luff. 1991. Collaborative Activity and Technological Design: Task Coordination in London Underground Control Rooms. In *Proceedings of the Second European Conference on Computer-Supported Cooperative Work ECSCW'91*. Springer, 65–80.

[22] Hikaru Ibayashi, Yuta Sugiura, Daisuke Sakamoto, Natsuki Miyata, Mitsunori Tada, Takashi Okuma, Takeshi Kurata, Masaaki Mochimaru, and Takeo Igarashi. 2015. Dollhouse VR: A Multi-View, Multi-User Collaborative Design Workspace with VR Technology. In *SIGGRAPH Asia 2015 Emerging Technologies* (Kobe, Japan) *(SA '15)*. Association for Computing Machinery, New York, NY, USA, Article 8, 2 pages. https://doi.org/10.1145/2818466.2818480

[23] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. 2011. KinectFusion: Real-Time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 559–568. https://doi.org/10.1145/2047196.2047270

[24] Brett Jones, Rajinder Sodhi, Michael Murdock, Ravish Mehra, Hrvoje Benko, Andrew Wilson, Eyal Ofek, Blair MacIntyre, Nikunj Raghuvanshi, and Lior Shapira. 2014. RoomAlive: Magical Experiences Enabled by Scalable, Adaptive Projector-Camera Units. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) *(UIST '14)*. Association for Computing Machinery, New York, NY, USA, 637–644. https://doi.org/10.1145/2642918.2647383

[25] André Kunert, Alexander Kulik, Stephan Beck, and Bernd Froehlich. 2014. Photoportals: Shared References in Space and Time. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing* (Baltimore, Maryland, USA) *(CSCW '14)*. Association for Computing Machinery, New York, NY, USA, 1388–1399. https://doi.org/10.1145/2531602.2531727

[26] David Ledo, Steven Houben, Jo Vermeulen, Nicolai Marquardt, Lora Oehlberg, and Saul Greenberg. 2018. Evaluation Strategies for HCI Toolkit Research. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–17. https://doi.org/10.1145/3173574.3173610

[27] Daniel Leithinger, Sean Follmer, Alex Olwal, and Hiroshi Ishii. 2014. Physical Telepresence: Shape Capture and Display for Embodied, Computer-Mediated Remote Collaboration. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) *(UIST '14)*. Association for Computing Machinery, New York, NY, USA, 461–470. https://doi.org/10.1145/2642918.2647377

[28] Germán Leiva, Jens Emil Grønbæk, Clemens Nylandsted Klokmose, Cuong Nguyen, Rubaiat Habib Kazi, and Paul Asente. 2021. Rapido: Prototyping Interactive AR Experiences through Programming by Demonstration. In *UIST '21: The 34th Annual ACM Symposium on User Interface Software and Technology, Virtual Event, USA, October 10-14, 2021*, Jeffrey Nichols, Ranjitha Kumar, and Michael Nebeling (Eds.). ACM, 626–637. https://doi.org/10.1145/3472749.3474774

[29] Germán Leiva, Cuong Nguyen, Rubaiat Habib Kazi, and Paul Asente. 2020. Pronto: Rapid Augmented Reality Video Prototyping Using Sketches and Enaction. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376160

[30] David Lindlbauer, Anna Maria Feit, and Otmar Hilliges. 2019. Context-Aware Online Adaptation of Mixed Reality Interfaces. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 147–160. https://doi.org/10.1145/3332165.3347945

[31] David Lindlbauer and Andy D. Wilson. 2018. *Remixed Reality: Manipulating Space and Time in Augmented Reality*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3173703

[32] Stephan Lukosch, Mark Billinghurst, Leila Alem, and Kiyoshi Kiyokawa. 2015. Collaboration in Augmented Reality. *Computer Supported Cooperative Work (CSCW)* 24, 6 (2015), 515–525.

[33] Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. 2004. DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology* (Santa Fe, NM, USA) *(UIST '04)*. Association for Computing Machinery, New York, NY, USA, 197–206. https://doi.org/10.1145/1029632.1029669

[34] Nicolai Marquardt, Robert Diaz-Marino, Sebastian Boring, and Saul Greenberg. 2011. The Proximity Toolkit: Prototyping Proxemic Interactions in Ubiquitous Computing Ecologies. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 315–326. https://doi.org/10.1145/2047196.2047238

[35] Meteor. 2021. Meteor. https://www.meteor.com/

[36] Microsoft. 2018. Mixed Reality Toolkit Unity. https://github.com/microsoft/MixedRealityToolkit-Unity.

[37] Microsoft. 2021. Microsoft Mesh. https://www.microsoft.com/en-us/mesh

[38] Jens Müller, Roman Rädle, and Harald Reiterer. 2017. *Remote Collaboration With Mixed Reality Displays: How Shared Virtual Landmarks Facilitate Spatial Referencing*. Association for Computing Machinery, New York, NY, USA, 6481–6486. https://doi.org/10.1145/3025453.3025717

[39] Jens Müller, Roman Rädle, and Harald Reiterer. 2017. Remote Collaboration With Mixed Reality Displays: How Shared Virtual Landmarks Facilitate Spatial Referencing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 6481–6486. https://doi.org/10.1145/3025453.3025717

[40] Michael Nebeling, Katy Lewis, Yu-Cheng Chang, Lihan Zhu, Michelle Chung, Piaoyang Wang, and Janet Nebeling. 2020. XRDirector: A Role-Based Collaborative Immersive Authoring System. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376637

[41] Michael Nebeling and Katy Madier. 2019. 360proto: Making Interactive Virtual Reality & Augmented Reality Prototypes from Paper. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300826

[42] Michael Nebeling, Janet Nebeling, Ao Yu, and Rob Rumble. 2018. *ProtoAR: Rapid Physical-Digital Prototyping of Mobile Augmented Reality Applications*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173927

[43] Benjamin Nuernberger, Eyal Ofek, Hrvoje Benko, and Andrew D Wilson. 2016. Snaptoreality: Aligning Augmented Reality to the Real World. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 1233–1244. https://doi.org/10.1145/2858036.2858250

[44] Kenton O'hara, Jesper Kjeldskov, and Jeni Paay. 2011. Blended Interaction Spaces for Distributed Team Collaboration. *ACM Trans. Comput.-Hum. Interact.* 18, 1, Article 3 (May 2011), 28 pages. https://doi.org/10.1145/1959022.1959025

[45] Dan R Olsen Jr. 2007. Evaluating User Interface Systems Research. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*. 251–258. https://doi.org/10.1145/1294211.1294256

[46] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L. Davidson, Sameh Khamis, Mingsong Dou, Vladimir Tankovich, Charles Loop, Qin Cai, Philip A. Chou, Sarah Mennicken, Julien Valentin, Vivek Pradeep, Shenlong Wang, Sing Bing Kang, Pushmeet Kohli, Yuliya Lutchyn, Cem Keskin, and Shahram Izadi. 2016. Holoportation: Virtual 3D Teleportation in Real-Time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 741–754. https://doi.org/10.1145/2984511.2984517

[47] Tomislav Pejsa, Julian Kantor, Hrvoje Benko, Eyal Ofek, and Andrew Wilson. 2016. Room2Room: Enabling Life-Size Telepresence in a Projected Augmented Reality Environment. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing* (San Francisco, California, USA) *(CSCW '16)*. Association for Computing Machinery, New York, NY, USA, 1716–1725. https://doi.org/10.1145/2818048.2819965

[48] Iulian Radu, Tugce Joy, Yiran Bowman, Ian Bott, and Bertrand Schneider. 2021. A Survey of Needs and Features for Augmented Reality Collaborations in Collocated Spaces. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW1, Article 169 (April 2021), 21 pages. https://doi.org/10.1145/3449243

[49] Franziska Roesner, David Molnar, Alexander Moshchuk, Tadayoshi Kohno, and Helen J Wang. 2014. World-Driven Access Control for Continuous Sensing. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 1169–1181. https://doi.org/10.1145/2660267.2660319

[50] Robert W. Root. 1988. Design of a Multi-Media Vehicle for Social Browsing. In *Proceedings of the 1988 ACM Conference on Computer-Supported Cooperative Work* (Portland, Oregon, USA) *(CSCW '88)*. Association for Computing Machinery, New York, NY, USA, 25–38. https://doi.org/10.1145/62266.62269

[51] Kimberly Ruth, Tadayoshi Kohno, and Franziska Roesner. 2019. Secure Multi-User Content Sharing for Augmented Reality Applications. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 141–158. https://www.usenix.org/conference/usenixsecurity19/presentation/ruth

[52] Kjeld Schmidt and Carla Simone. 1996. Coordination Mechanisms: Towards a Conceptual Foundation of CSCW Systems Design. *Comput. Support. Cooperative Work.* 5, 2/3 (1996), 155–200. https://doi.org/10.1007/BF00133655

[53] Gareth Smith. 1996. Cooperative Virtual Environments: Lessons from 2D Multi User Interfaces. In *Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work* (Boston, Massachusetts, USA) *(CSCW '96)*. Association for Computing Machinery, New York, NY, USA, 390–398. https://doi.org/10.1145/240080.240350

[54] Maurício Sousa, Daniel Mendes, Rafael Kuffner Dos Anjos, Daniel Medeiros, Alfredo Ferreira, Alberto Raposo, João Madeiras Pereira, and Joaquim Jorge. 2017. Creepy Tracker Toolkit for Context-Aware Interfaces. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces* (Brighton, United Kingdom) *(ISS '17)*. Association for Computing Machinery, New York, NY, USA, 191–200. https://doi.org/10.1145/3132272.3134113

[55] Spatial. 2021. Spatial. https://spatial.io/

[56] Maximilian Speicher, Jingchen Cao, Ao Yu, Haihua Zhang, and Michael Nebeling. 2018. 360Anywhere: Mobile Ad-Hoc Collaboration in Any Environment Using 360 Video and Augmented Reality. *Proc. ACM Hum.-Comput. Interact.* 2, EICS, Article 9 (June 2018), 20 pages. https://doi.org/10.1145/3229091

[57] Maximilian Speicher, Brian D. Hall, Ao Yu, Bowen Zhang, Haihua Zhang, Janet Nebeling, and Michael Nebeling. 2018. XD-AR: Challenges and Opportunities in Cross-Device Augmented Reality Application Development. *Proc. ACM Hum.-Comput. Interact.* 2, EICS, Article 7 (June 2018), 24 pages. https://doi.org/10.1145/3229089

[58] M. Sra, S. Garrido-Jurado, and P. Maes. 2018. Oasis: Procedurally Generated Social Virtual Spaces from 3D Scanned Real Spaces. *IEEE Transactions on Visualization and Computer Graphics* 24, 12 (2018), 3174–3187. https://doi.org/10.1109/TVCG.2017.2762691

[59] Misha Sra, Aske Mottelson, and Pattie Maes. 2018. Your Place and Mine: Designing a Shared VR Experience for Remotely Located Users. In *Proceedings of the 2018 Designing Interactive Systems Conference* (Hong Kong, China) *(DIS '18)*. Association for Computing Machinery, New York, NY, USA, 85–97. https://doi.org/10.1145/3196709.3196788

[60] Aaron Stafford, Wayne Piekarski, and Bruce H Thomas. 2006. Implementation of God-Like Interaction Techniques for Supporting Collaboration Between Outdoor AR and Indoor Tabletop Users. In *2006 IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE, 165–172. https://doi.org/10.1109/ISMAR.2006.297809

[61] Unity Technologies. 2021. Unity. https://unity.com/

[62] Balasaravanan Thoravi Kumaravel, Fraser Anderson, George Fitzmaurice, Bjoern Hartmann, and Tovi Grossman. 2019. Loki: Facilitating Remote Instruction of Physical Tasks Using Bi-Directional Mixed-Reality Telepresence. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 161–174. https://doi.org/10.1145/3332165.3347872

[63] Three.js. 2021. Three.js. https://threejs.org/

[64] Unity. 2021. Unity MARS. https://unity.com/products/unity-mars

[65] Chiu-Hsuan Wang, Chia-En Tsai, Seraphina Yong, and Liwei Chan. 2020. Slice of Light: Transparent and Integrative Transition Among Realities in a Multi-HMD-User Environment. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 805–817. https://doi.org/10.1145/3379337.3415868

[66] Robert Xiao, Chris Harrison, and Scott E. Hudson. 2013. WorldKit: Rapid and Easy Creation of Ad-Hoc Interactive Applications on Everyday Surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 879–888. https://doi.org/10.1145/2470654.2466113

Received 2022-02-08; accepted 2022-06-09