GaussianNexus: Room-Scale Real-Time AR/VR Telepresence with Gaussian Splatting

Xincheng Huang*
University of British Columbia
Vancouver, BC, Canada
xincheng.huang@ubc.ca

Dieter Frehlich*
University of British Columbia
Vancouver, BC, Canada
frehlid@student.ubc.ca

Ziyi Xia
University of British Columbia
Vancouver, BC, Canada
zxia0101@cs.ubc.ca

Peyman Gholami
University of British Columbia
Vancouver, BC, Canada
peymang@cs.ubc.ca

Robert Xiao
University of British Columbia
Vancouver, BC, Canada
brx@cs.ubc.ca

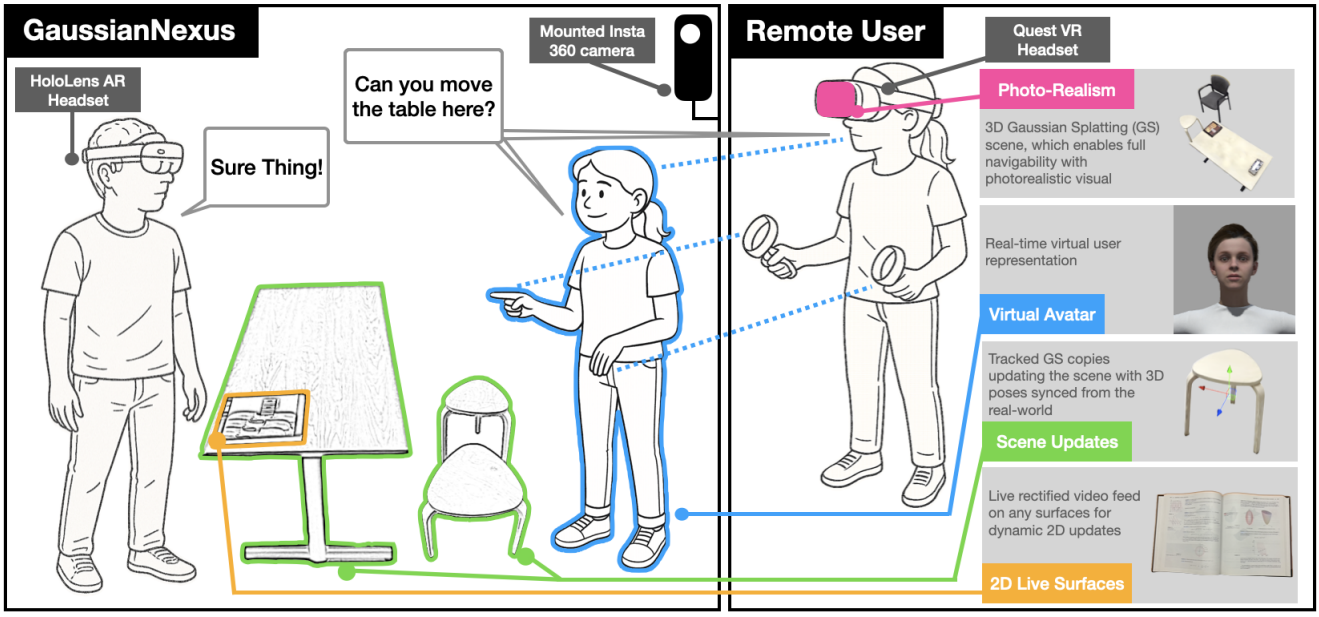


Figure 1: GaussianNexus – A Mixed-Reality Telepresence System: A HoloLens AR user and a Quest VR user share a spatial workspace through Gaussian Splatting-based 3D scene rendering and real-time object synchronization. The system enables: (1) view-dependent 3D scene exploration, (2) a virtual avatar with live head and hand tracking, (3) interactive movable objects that update the Gaussian Splatting scene in near real time, and (4) embedded, rectified 2D live video on physical surfaces.

Abstract

Telepresence systems with AR/VR immerse a remote user in a local physical environment, enabling virtual travel, remote guidance, and collaborative design. Contemporary systems typically rely on 360° video or RGB-D reconstruction—each with trade-offs between visual fidelity and spatial perception. Emerging rendering techniques like Gaussian Splatting unify these strengths, offering photo-realistic scene representations with spatial interactivity. However, due to the long training times required, updating such scenes in real-time is still largely infeasible. We present *GaussianNexus*, a system that applies Gaussian Splatting to room-scale telepresence. Our system uses Gaussian Splatting as the primary scene representation medium, and a 360° camera to stream and track 2D and 3D dynamic changes. For live 2D interaction, the system overlays rectified video onto user-selected surfaces. For live 3D interaction, users identify dynamic objects in the environment, which are then segmented, tracked and synchronized as real-time updates to the Gaussian Splatting environment, enabling smooth, low-latency telepresence

*Both authors contributed equally to this research.

UIST '25, Busan, Republic of Korea

ACM ISBN 979-8-4007-2037-6/2025/09
https://doi.org/10.1145/3746059.3747693

without retraining. We demonstrate the utility of *GaussianNexus* through 2 example applications and evaluate it in a usability test.

CCS Concepts

• **Human-centered computing → Collaborative interaction**; **Mixed / augmented reality**.

Keywords

Telepresence, Remote AR/VR Collaboration, Computer Mediated Communication, Gaussian Splatting

ACM Reference Format:

Xincheng Huang, Dieter Frehlich, Ziyi Xia, Peyman Gholami, and Robert Xiao. 2025. GaussianNexus: Room-Scale Real-Time AR/VR Telepresence with Gaussian Splatting. In *The 38th Annual ACM Symposium on User Interface Software and Technology (UIST '25), September 28-October 1, 2025, Busan, Republic of Korea.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3746059.3747693

1 Introduction

Telepresence systems with AR/VR bring people together in a shared space, enabling natural collaboration and immersive presence. Typically in such systems, a "local user" wears an AR headset and is physically present in their real-world environment, while a "remote user" virtually joins them through VR. These systems hold great potential for realistic shared experiences such as virtual travel [7, 32], remote guidance [14, 18, 68, 78], and collaborative design [23, 50].

However, contemporary telepresence systems still fall short of delivering fully immersive and realistic experiences. Common scene representation approaches—such as RGB-D reconstruction and 360° video—struggle to achieve spatial interactivity and visual fidelity simultaneously [66]. While 360° video offers high-resolution visuals, it lacks depth perception and restricts user interaction due to the inability to navigate the scene [21, 56]. In contrast, RGB-D reconstruction enables spatial interaction and free scene navigation, but at the cost of lower visual quality [66, 68].

Emerging neural rendering techniques, such as Neural Radiance Fields (NeRF) [5, 49] and Gaussian Splatting [34], offer photo-realistic scene representations while allowing spatial interactivity and free-viewpoint navigation—combining the strengths of 360° video and RGB-D. However, these techniques are primarily designed for static scenes and struggle to support dynamic updates in real time, making their application to telepresence challenging.

Recently, SharedNeRF [58] has attempted to bridge this gap by using optical flow to detect scene changes and retraining Instant-NGP [49] on the fly. While effective in small-scale settings, it relies on users' head movements to collect training views—limiting it to tabletop scenarios—and discretely updates the NeRF scene approximately every 5 seconds, combining it with a lower-quality point cloud for real-time visuals. Therefore, real-time AR/VR telepresence with neural rendering at room scale remains an open challenge.

We present *GaussianNexus*, a room-scale real-time AR/VR telepresence system utilizing Gaussian Splatting, combined with live video, to provide high visual quality and full navigability—the remote VR user can freely stand or walk anywhere within the reconstructed local environment while retaining photorealistic scene quality. Our system combines Gaussian Splatting, 3D segmentation, and real-time object tracking to eliminate the need for retraining the scene on the fly. *GaussianNexus* handles 2D and 3D content updates separately. For 2D content (e.g., screens and whiteboards), it overlays live, rectified video—captured by the 360° camera—onto surfaces selected by the remote user. For 3D updates, *GaussianNexus* introduces a scene preparation phase, where the local user is assisted in capturing and training Gaussian splats, identifying and segmenting interactable objects, and creating movable Gaussian Splatting cutouts. This preparation only needs to be performed once per environment; objects will be tracked across telepresence sessions even if they are moved in the interim. During telepresence, the system tracks these objects in real time as the local user interacts with them, and synchronizes their poses within the Gaussian Splatting scene for the remote user. Our technique computes motion updates continuously using a novel splat-rendering optimizer, which tracks object poses at about 2.33 fps. Combined with smoothing-induced delay and hardware (e.g., camera/network) processing delay, we achieve an end-to-end latency of on average 0.95 second with commodity hardware. We demonstrate *GaussianNexus* across 2 example application scenarios: tabletop remote instruction with planar objects, and collaborative room layout planning. We further validate the system's usability and potential in a user study with 9 participants. To our knowledge, *GaussianNexus* is the first system to combine real-time scene updates, room-scale navigability, and high visual fidelity from any viewpoint, marking a significant step toward hyper-realistic telepresence.

2 Related Work

The prior work most closely related to *GaussianNexus* involves research on immersive telepresence and remote collaboration using AR/VR [21, 23, 51, 66, 68]. Such systems typically reconstruct the physical environment of a local user and virtually "teleport" a remote user into this shared space. The most common approaches for reconstructing and representing physical environments in telepresence include 360° video [21, 56, 66] and RGB-D-based reconstructions, such as point clouds [24–26, 68] or textured spatial meshes [29, 65, 78]. More recently, emerging neural rendering techniques—such as NeRF [5, 46, 49] and Gaussian Splatting [8, 34]—have enabled photo-realistic 3D scene representations, demonstrating great potential as media for immersive experiences and remote collaboration. Here, we review prior telepresence systems leveraging these media to contextualize our contributions.

2.1 360° Video Telepresence

Rendering 360° video in virtual reality allows a remote user to omnidirectionally view a space from a first-person perspective. With commodity 360° cameras capable of streaming at 6–8K resolution, telepresence systems using 360° video achieve relatively high visual realism [21, 56]. Previous studies have applied 360° video telepresence in virtual tourism [32], remote guidance of physical tasks [53, 54], and collaborative prototyping [23]. However, while 360° video creates the illusion of immersive 3D space, it inherently remains a 2D texture, limiting richer spatial interactions such as free navigation and scene augmentation.

In traditional 360° telepresence, the remote user's viewpoint is fixed at the camera position. Systems with stationary camera setups [21, 56] constrain the remote user's viewpoint and mobility. To mitigate this, previous research mounted 360° cameras directly onto local users [65–67]. For instance, JackInHead [32] placed a camera on the local user's head, enabling remote users to share the local user's perspective. Teo et al. [65, 67] further allowed users to visualize each other's real-time viewpoints, enhancing collaborative awareness[6, 16, 64]. Some systems have explored hand-held 360° cameras [54], allowing the local user to guide a remote collaborator for close inspection and instruction. However, mounting cameras onto the local user's body introduces inconsistencies between physical and perceived motion for the remote viewer, potentially causing simulator sickness [17, 19, 33, 70], while limiting the remote user's independent exploration. Alternatively, mounting 360° cameras onto robotic platforms [30] offers mobility but is expensive and cumbersome for general use.

Prior research emphasizes maintaining mutual awareness between collaborators through a shared "reference" space [6, 64]. Consequently, 360° telepresence systems have implemented synchronized interaction techniques, including hand-rays and annotations [23, 56, 67]. However, the lack of binocular depth perception in monocular 360° video restricts richer interactions, such as manipulating shared virtual assets. VirtualNexus [23] renders virtual objects monocularly along with the 360° video, albeit at the expense of accurate depth perception during interaction.

2.2 RGB-D Reconstructed Telepresence

Compared to 360° video, RGB-D reconstructions—such as point clouds [26, 51, 77] and textured spatial meshes [29, 65, 78]—are inherently 3D, allowing remote users to freely explore environments with accurate depth perception. KinectFusion [26] first demonstrated RGB-D reconstruction using consumer-grade hardware. The asymmetrical collaboration system Volumetric Mixed Reality Telepresence [25] streams real-time point cloud reconstructions of local workspaces, supporting remote guidance for physical tasks. Loki [68] integrates similar remote guidance systems with additional interaction methods like annotations and recorded playback. Extending this approach, Irlitti et al. [24] proposed symmetrical remote collaboration by merging real-time point clouds from two physically separate but identical environments.

Despite superior spatial perception and richer interaction capabilities, RGB-D reconstructions typically suffer from lower visual quality [66], including reduced fields-of-view, occlusion issues, noisy depth capture, and inaccurate lighting. Holoportation [51] addressed these limitations by capturing local environments using multiple RGB-D cameras from different angles, significantly improving visual fidelity. However, such setups require specialized, high-end hardware and substantial bandwidth, making them less accessible to general users.

2.2.1 Combining 360° Video and RGB-D Reconstruction. Prior research has also explored combining RGB-D reconstructions with 360° video [66] to leverage their complementary strengths: 360° video prioritizes visual quality at the expense of spatial interactivity, whereas RGB-D reconstruction prioritizes spatial interaction over visual fidelity. Research by Teo et al. [65–67] has explored combining these two media for remote collaboration, by integrating recorded 360° panoramas and live 360° video streams into static spatial meshes [65]. Teo et al. [66] further proposed switching between live 360° video and RGB-D reconstruction. However, the differing interactive modalities required by these media result in frequent context switching, which could increase the user's mental load.

Huang et al. [23] proposed representing scenes primarily with 360° video, embedding a registered spatial mesh to facilitate realistic physical interactions with virtual objects. They additionally enabled users to copy and extract synchronized "cutouts" from 360° videos as textured spatial meshes, offering an alternative to physical scene navigation. Despite these improvements, systems combining 360-degree video and RGB-D reconstruction still struggle to provide fully natural and realistic telepresence experiences due to mode-switching and inconsistent visual quality.

2.3 Neural Rendering and AR/VR Collaboration

Neural rendering is a class of emerging techniques that leverage neural networks or differentiable representations to synthesize novel views from images [5, 34, 46, 49]. These methods generate photo-realistic renderings of 3D scenes and are capable of capturing subtle visual details such as complex materials and lighting effects.

The first widely recognized neural rendering technique, Neural Radiance Fields (NeRF) [46], models a scene by representing the color and density along spatial rays using a fully connected feed-forward neural network. Mip-NeRF 360 [5] extends this framework to handle unbounded scenes by introducing hierarchical sampling and mipmapping strategies. While early versions of NeRF required tens of hours to train, Instant-NGP [49] introduced a multiresolution hash encoding that drastically reduced training times to minutes or even seconds.

More recently, Gaussian Splatting [34] has emerged as an alternative to NeRF. Instead of modeling scenes implicitly through ray-based neural fields, Gaussian Splatting represents scenes explicitly using a set of 3D Gaussians with differentiable parameters, trained via gradient descent. This explicit representation enables efficient rendering and allows for more intuitive scene editing and manipulation compared to NeRF-based approaches. Strictly speaking, Gaussian Splatting does not involve neural networks, but is often loosely categorized as "neural rendering" due to its differentiable formulation.

Unlike video or point clouds, both NeRF and Gaussian Splatting are static scene representation techniques. Although creating live photorealistic human avatars with neural rendering is already possible [28, 38, 39, 45, 57], rendering general dynamic scenes in real-time (e.g., 30 fps) remains largely infeasible due to the significant computational demands of modeling and retraining to arbitrary dynamic content. Recent advances in 4D reconstruction with neural rendering [13, 41, 55, 63, 73] targets volumetric video reconstruction. However, these methods require access to the entire pre-recorded video, as they perform global optimization across all frames. As a result, adapting them for live use is non-trivial. Moreover, state-of-the-art 4D reconstruction approaches such as GaussianFlow [41] and 4DGS [73] still require approximately 10 seconds of training per frame, making them far from suitable for

real-time streaming. Quark [12] achieves 30 fps Gaussian Splatting reconstruction and rendering, but requires a large and dense array of cameras (e.g., 8 cameras spaced 30 cm apart), and movements beyond the camera's baseline can significantly degrade visual quality or cause reconstruction failure. These limitations preclude its use in telepresence scenarios, where free navigation and continuous scene consistency are essential.

Due to the above challenges, few prior works have applied neural rendering to real-time, synchronous remote collaboration. To the best of our knowledge, the only existing system that precedes ours is SharedNeRF [58], which detects changes in a dynamic scene using optical flow and retrains the scene representation using Instant-NGP [49]. However, SharedNeRF relies on a user's natural head movement to collect new training data, limiting its application to tabletop-scale collaboration. At room scale, such a system would require users to continuously walk around the scene to gather updated images, significantly disrupting the collaborative experience. Furthermore, SharedNeRF discretely updates its scene representation approximately 5 seconds after detecting a dynamic change and bridges this gap using real-time point cloud rendering—resulting in a noticeable compromise in visual quality. In contrast, with GaussianNexus, users pre-identify and segment dynamic components before collaboration begins, eliminating the need for in-session scene retraining. By tracking the 3D pose of these dynamic elements in real time, *GaussianNexus* continuously updates the Gaussian Splatting scene with reduced delay, maintaining high visual fidelity throughout the collaborative session.

3 System Design Principles

Distilling from the design choices and gaps left by the related work, here we identify the system design principles and technical challenges for our system, justifying the features our system implements, which will be detailed in Section 4.

3.1 Photo-Realism with Spatial Interactivity

Prior studies have shown that higher visual quality and resolution generally lead to better immersion and presence [62]. Besides neural rendering, scene representation based on 360° video [21, 56] offers similarly high visual fidelity, despite its limited spatial interactivity. While combining 360° video with RGB-D reconstruction improves interactivity, switching between modalities with inconsistent visual quality disrupts continuity and breaks the sense of realism.

Inspired by prior efforts to combine telepresence media, *GaussianNexus* integrates Gaussian Splatting, 360° video, and RGB-D spatial reconstruction into a unified system. To maintain consistent photo-realism, our system adopts Gaussian Splatting as the primary scene representation medium. While projected, rectified 360° video is used for updating dynamic surfaces (e.g., screens, whiteboards), we avoid visually presenting point clouds or spatial meshes, which would compromise visual consistency. Instead, *GaussianNexus* maintains a hidden spatial mesh registered to the Gaussian Splatting scene. This mesh enables physical effects—such as collisions and rigid-body interactions—enhancing the user's sense of physical presence and spatial realism. The system also offers common collaborative modalities such as hand-rays, annotations, virtual avatars, and shared virtual primitives.

3.2 Scalability to Room-Scale Environments

Telepresence systems may support remote activities across different spatial scales, ranging from tabletop interactions [31, 58], to room-scale environments [52, 68], and even unconstrained spaces [32]. Existing telepresence systems that apply neural rendering to the task space (i.e., beyond avatar reconstruction), such as SharedNeRF [58], are limited to tabletop collaboration. Scaling neural rendering–based telepresence to support broader applications—such as shared immersive experiences [52] or collaborative indoor layout planning [80]—remains an open challenge.

We identify the core bottleneck in scaling such systems to be the need for on-the-fly scene retraining. This process requires continuous capture and updating of the scene dataset. While natural head movement from a wearable camera may suffice for tabletop interactions, the same approach at room scale would require the user to traverse the entire space continuously, severely disrupting collaborative flow.

GaussianNexus eliminates the need for retraining during the telepresence session by introducing a scene preparation phase. Prior to the session, the system assists users in identifying and segmenting objects they expect to interact with into 3D Gaussian Splatting copies that can be individually tracked and moved. This converts the problem of retraining into one of 3D object tracking. Instead of retraining the whole scene, *GaussianNexus* takes a pretrained GS scene and updates it live to match reality. This approach is essential for enabling real-time interaction and room-scale navigability, while consistently maintaining photorealistic fidelity. The preparation process should be reusable across sessions to reduce overhead. The object tracking is then performed with a 360° camera which omnidirectionally monitors the scene in real time. Although it is designed for room-scale telepresence, the system remains compatible with smaller-scale collaboration (e.g. tabletop interaction).

3.3 Low-Latency Dynamic Content Integration

Scene retraining is also a key bottleneck when integrating dynamic content into neural rendering telepresence. To address this challenge, *GaussianNexus* introduces a dual-mode dynamic update strategy. For dynamic 2D content (e.g., screens or whiteboards), we directly re-project and overlay 360° video onto user-selected surfaces at runtime. For dynamic 3D content, as previously described, we leverage the object segmentation performed during the scene preparation phase, effectively transforming dynamic scene updates into a 3D object tracking problem.

However, general-purpose 3D object tracking from monocular RGB video remains an open problem in computer vision. Existing methods, such as YOLO3D [48], SAM3D [79], SMOKE [44], and MediaPipe Objectron [1, 2], typically rely on object categories (e.g., cars, pedestrians). Adapting such methods to each new telepresence scene would require extensive dataset collection and retraining, introducing significant overhead.

Consequently, a critical challenge for *GaussianNexus* emerges: achieving general-purpose, category-agnostic 3D object tracking within a Gaussian Splatting-based scene representation. To address this, our system utilizes the existing 360° video feed to preserve hardware simplicity and omnidirectional coverage. An alternative choice is to use the HoloLens' embedded depth camera. However,

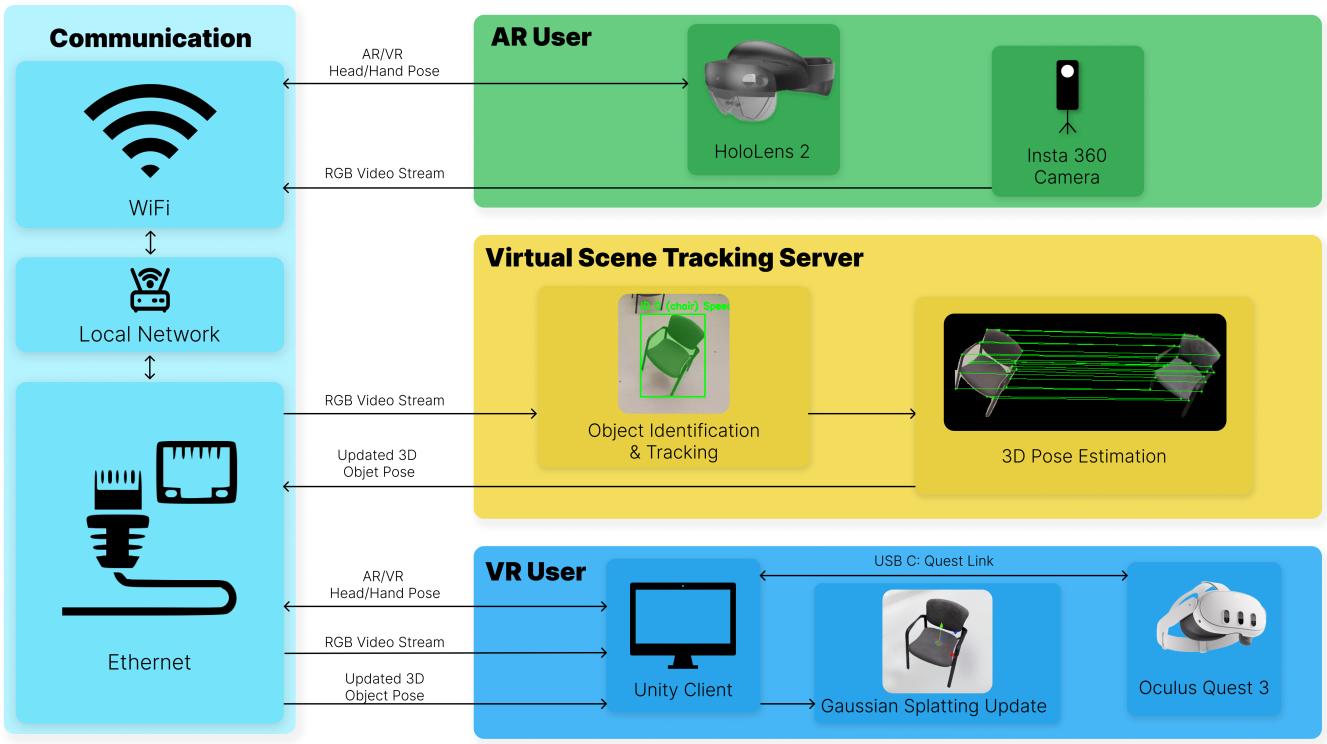


Figure 2: System architecture of GaussianNexus: The AR user side is equipped with a HoloLens 2 and an Insta360 camera, which streams RGB video to a local server over Wi-Fi. The Virtual Scene Tracking Server performs object identification, 3D pose estimation, and Gaussian splat updates based on the captured scene. The processed data — including dynamic object poses and user tracking—is synchronized with the VR user via Ethernet. The VR user, connected through a Quest Link to a desktop PC, receives real-time updates of the shared scene.

this requires the user to look at the objects they are manipulating, precluding eyes-free manipulation. It is also possible to use a dedicated depth camera such as Microsoft Kinect Azure[1], while further complicating the system with extra spatial registration and synchronization.

Therefore, we developed our own object tracking technique that employs the Gaussian splats of pre-segmented objects as tracking priors. This tracking solution should operate close to real time, ensuring low-latency updates without disrupting collaboration.

4 The GaussianNexus System

4.1 System Architecture

We implemented *GaussianNexus* using Unity 2021.3.20f1. The system can be configured and deployed in either AR or VR mode. In a typical session, the AR user is physically present in the local environment and wears a Microsoft HoloLens 2 (HL2)[2]. The VR user joins the environment remotely via a Meta Quest 3 headset[3]. An Insta360 X3 360° camera (with a maximum streaming resolution of 5760 × 2880) at the AR side captures and streams 360° video of the local environment to the remote user[4]. Meanwhile, the HoloLens maintains a spatial mesh of the physical world, which is also streamed to the VR side in real-time. We illustrate the system architecture of *GaussianNexus* in Figure 2.

To spatially register the Gaussian Splatting scene with the physical world (see Section 4.2.1), we attach two QR codes to fixed positions on the wall. An additional QR code is mounted on the 360° camera to track its pose. On the VR side, the Gaussian Splatting scene is rendered in Unity. Concurrently, the same machine runs a 3D object tracking algorithm (see Section 4.4), which takes the 360° video stream and YOLOv11 [35] object detection results as input. This algorithm synchronizes physical object movements from the local AR environment with their counterparts in the remote VR scene. The VR system runs on a desktop equipped with an Intel Core i9-12900KF 3.2GHz CPU, 64GB of RAM, and an NVIDIA GeForce RTX 4090 GPU. Both AR and VR users see each other as virtual avatars, which are open-source rigged models from Mixamo[5] with their full-body pose inferred using FinalIK[6] from HL2/Quest 3 head and controller data. The users are equipped with ray pointers, shared virtual objects, and synchronized annotations.

[1]https://azure.microsoft.com/en-us/products/kinect-dk
[2]https://learn.microsoft.com/en-us/hololens/
[3]https://www.meta.com/ca/quest/quest-3/
[4]https://unity.com/
[5]https://www.mixamo.com/
[6]https://assetstore.unity.com/packages/tools/animation/final-ik-14290

4.2 Scene Capture and Preparation

GaussianNexus presents the local environment to the remote user primarily through Gaussian Splatting, which we train using the original 3D Gaussian Splatting pipeline [34]. This process requires the local user to record a video of the physical scene and extract frames as individual images. We then use COLMAP [59, 60] to estimate camera poses for these images. With the extracted frames and corresponding poses, the user can run the training script provided in the 3D Gaussian Splatting repository[7] to reconstruct the environment in 3D Gaussian splats. We then use an open-source Gaussian Splatting for Unity tool[8] to convert the reconstructed scene to Unity assets and render them.

The time required for training scales with the number of input images. Through our testing, we found that using approximately 300–400 images offers a good balance between visual quality and efficiency, resulting in a total processing time of around 30 minutes (including both camera pose estimation and training). In Figure 3a and 3b, we show a side-by-side comparison of the real-world lab space and the reconstructed Gaussian Splatting environment.

4.2.1 Aligning the Virtual and Physical Worlds. As previously described, *GaussianNexus* combines 3D Gaussian Splatting, 360° video, and spatial meshes. To achieve this, it is essential to spatially align the virtual and physical environments. More specifically, the key technical challenge is to establish a common reference space shared between the Gaussian Splatting scene and the AR world.

Before capturing the environment, the local user needs to instrument the physical space by attaching two QR codes to orthogonal walls in the room. Figure 3c shows an example setup. If the codes are placed on non-perpendicular walls, users will need to manually specify the angle between the QR code planes.

To use these QR codes as spatial anchors, our system must determine their poses within the coordinate system of the Gaussian Splatting scene. A naive solution would be to detect the QR codes directly from the Gaussian Splatting reconstruction, but this assumes the QR patterns are clearly preserved. Instead, our system traverses the set of original captured images. For any image containing a QR code, we detect its transform relative to the scene origin using Python libraries (pyzbar[9] and pycolmap[10]), and then uses RANSAC to fit a least squares model of where the corners should be. Since COLMAP [59, 60] provides all camera poses in the Gaussian Splatting coordinate system, we can then derive the QR code's global pose relative to the scene origin. In theory, two images per QR code are sufficient to recover its pose. In practice, within the 300–400 image sets typically used for training, each QR code appears in approximately 10–40 frames. We average the recovered poses across all such images to improve robustness and accuracy.

With the above computation, we align the Gaussian Splatting scene with the physical environment. We observed that COLMAP and the 3D Gaussian Splatting pipeline do not guarantee alignment with the real-world scale. In our captured lab environment, the reconstructed scene was nearly twice as large as the actual space, with slight differences in scale along the X, Y, and Z axes. To address this, we apply a scaling transformation to the Gaussian Splatting scene, by calculating the scale ratio for each axis by comparing the physical and virtual dimensions of the QR code anchors. These axis-specific ratios are then used to scale the scene accordingly, aligning it more accurately with the physical environment.

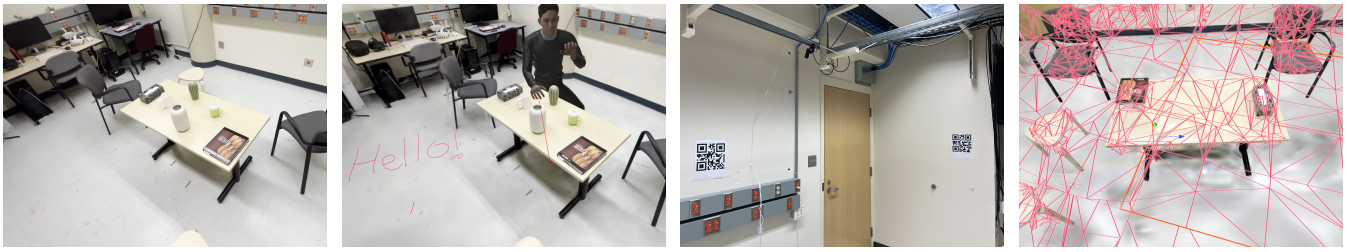
We then establish a common reference space between the Gaussian Splatting scene (i.e., the VR world) and the AR environment. Given the computed transforms between the virtual QR codes and the origin of the Gaussian Splatting scene, we replicate this spatial relationship in the AR application. Specifically, we place a *shared origin* (i.e., an empty Unity game object) in the AR world that maintains the same relative transform from the physical QR codes as the VR origin does from the virtual QR codes. This *shared origin* serves as a proxy for the VR coordinate system within the AR application. With the *shared origin* in place, any virtual asset shared between the AR and VR environments can be spatially aligned by transforming its pose relative to it. This ensures consistent placement and behavior of shared content across both local and remote users. In Figure 3d, we demonstrate the result of the scene alignment with a spatial mesh of the physical environment (captured by HoloLens 2) matched to the Gaussian Splatting scene. The spatial mesh is hidden during telepresence but supports interactions like virtual annotations and physics (e.g., a ball hitting a wall). Users raycast using pinch or controller input to make virtual annotations, which are drawn with Unity LineRenderers and synced over the network.

4.2.2 Object Pre-segmentation. To facilitate 3D updates during telepresence (see Section 4.4), our system assists the local user in pre-segmenting the objects they intend to interact with. We build on Segment-Any-3D-Gaussians (SAGA) by Cen et al. [8], a 3D segmentation method based on Segment Anything [37], which generates segmentation masks for 3D Gaussian Splatting reconstructions. To enable segmentation, SAGA's repository[11] provides a script to pre-process the same image set used for training the 3D Gaussian Splatting scene, which takes around 30 minutes on our machine with 300–400 images.

Since *GaussianNexus* incorporates YOLOv11 [35] as a core component for 3D tracking (see Section 4.4), the pre-segmentation process begins by automatically capturing an image of the scene using the 360° camera. We then apply YOLOv11 to detect bounding boxes for all recognizable objects in the scene (see Figure 4a). These bounding boxes are used as initial prompts for SAGA. The local user can then manually refine the segmentation masks for objects they plan to interact with, using the interactive interface provided by SAGA (see Figure 4b).

Once the masks are finalized, our system extracts individual Gaussian Splatting reconstructions for each segmented object and removes their original representations from the main scene. The segmented objects are subsequently overlaid back into the main scene. Because both the main scene and the individual objects originate from the same reconstruction, the objects retain their correct spatial positions without requiring additional alignment. Our system also uses the splat positions of each object to assign approximate 3D collision boxes, which can be manually refined by the user if needed.

[7] https://github.com/graphdeco-inria/gaussian-splatting
[8] https://github.com/aras-p/UnityGaussianSplatting
[9] https://pypi.org/project/pyzbar/
[10] https://colmap.github.io/pycolmap/index.html
[11] https://github.com/Jumpat/SegAnyGAussians



(a) (b) (c) (d)

Figure 3: (a) A training image used in the generation of the Gaussian splats. (b) The resulting Gaussian Splatting scene rendered in Unity from a similar viewpoint, featuring a virtual avatar and annotation. (c) Orthogonal QR codes employed for scene registration between the physical and virtual environments. (d) The spatial mesh captured by the HoloLens 2, overlaid on the Gaussian Splatting scene.

4.2.3 A Note on Effort and Reusability. For experienced users (e.g., the authors), the full scene capture and preparation phase takes approximately 1.5 hour for a setup with around 10 interactable objects. Specifically, the user first captures the scene over a 3-minute video recording. Training 3D Gaussian Splatting and SAGA takes approximately 1 hour, followed by object segmentation and importing into the Unity scene, which takes about half an hour for 10 objects. We acknowledge that this process introduces setup overhead for the local user. However, once prepared, the scene can be reused across multiple telepresence sessions. In Section 4.4, we describe how *GaussianNexus* supports scene reuse even if the interactable objects have been moved between sessions. We further discuss the reusability and limitations of this preparation process in Section 8.

4.3 2D Updates with 360° Video Projection

GaussianNexus uses 360° video to update 2D content surfaces in the scene, such as whiteboards, screens, or tabletop interaction with planar objects. During a telepresence session, the remote VR user can use their hand-ray to select the four corners of a surface they would like to make live. Our system then extracts the corresponding portion from the 360° video, rectifies it, and overlays it onto the selected surface within the Gaussian Splatting scene. Our system allows the user to create multiple live surfaces in the same session. We now describe the technical details behind this process.

To enable live surfaces through 360° video, it is essential to align the coordinate system of the 360° camera with that of the VR world. This alignment is straightforward, as the AR and VR environments have already been spatially synchronized as described earlier. We attach a QR code to the 360° camera, allowing the HoloLens 2 to track its pose in real time. This pose is then transformed and synchronized to the VR coordinate system.

The 360° camera streams raw fisheye video using an equidistant fisheye model, as described in Equation 1, where r is the radial distance from the image center, f is the focal length, and θ is the angle from the optical axis (i.e., the zenith angle). The optical axis is aligned with the camera's look direction, determined from its pose in the AR world.

$$r = f \cdot \theta \tag{1}$$

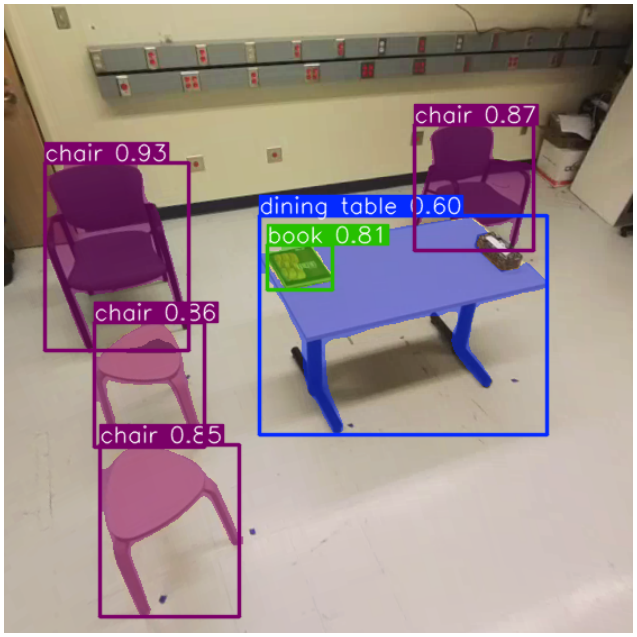
For each surface selected by the user, we create a quad based on the four specified corners. A custom shader is then used to extract the corresponding pixels from the 360° camera's texture and map them onto the quad using the camera's intrinsic parameters and the fisheye projection model, effectively turning the surface into a live video display. Notably, our system allows the local user to move the 360° camera freely during the telepresence session, allowing them to gain better resolution by moving the camera closer to a surface they actively work on.

4.4 3D Updates with Object Tracking

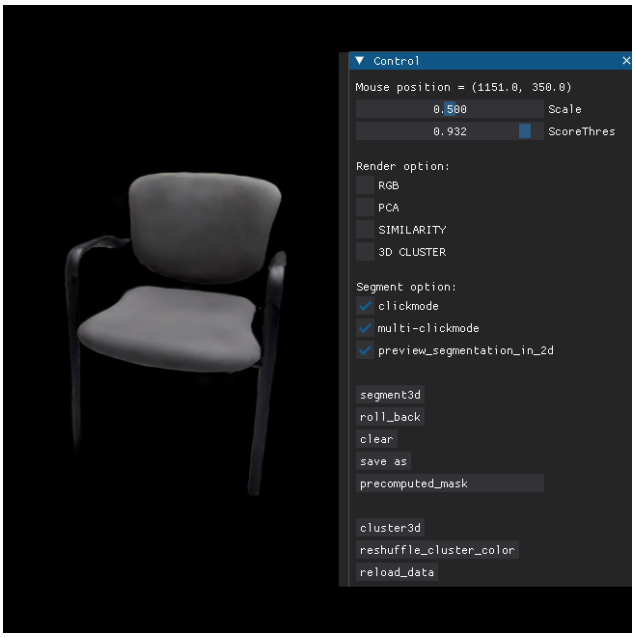
To support dynamic interactions with physical objects, *GaussianNexus* enables real-time 3D updates through object tracking. Building on the object pre-segmentation in the scene preparation phase, our system continuously tracks the segmented objects and synchronizes their poses within the Gaussian Splatting scene during telepresence.

4.4.1 2D Object Tracking with Identity. The first step of our 3D object tracking pipeline is to track object identities and their positions in the 2D video stream. For this purpose, we use YOLOv11 [35]. We begin by streaming the 360° video feed to YOLO. However, YOLOv11 requires a 640×640 rectified image as input, rather than a raw fisheye texture. To address this, we use the same projection approach described in Section 4.3 to project and rectify a central portion of the 360° camera's front video into a 640×640 video texture. The size of the central portion can be dynamically adjusted by the local user through the Unity interface, both before and during the telepresence session. This effectively changes the field of view (FoV) of the rectified video texture. Because we use a 360° camera, the FoV of the rectified view can approach 180°. Currently, our system only uses the front lens of the 360° camera for object tracking, although it is trivial to apply the same process to the back lens. Since our Unity application already receives the 360° video stream for 2D updates in VR mode, we implement the projection and rectification pipeline in Unity as well. The resulting rectified video feed is shared with YOLOv11, which we run with a Python Script, via a local network socket.

The original YOLO algorithm is designed for object detection rather than tracking. It assigns class labels to objects in each frame but does not maintain consistent object identities over time. As a



(a)



(b)

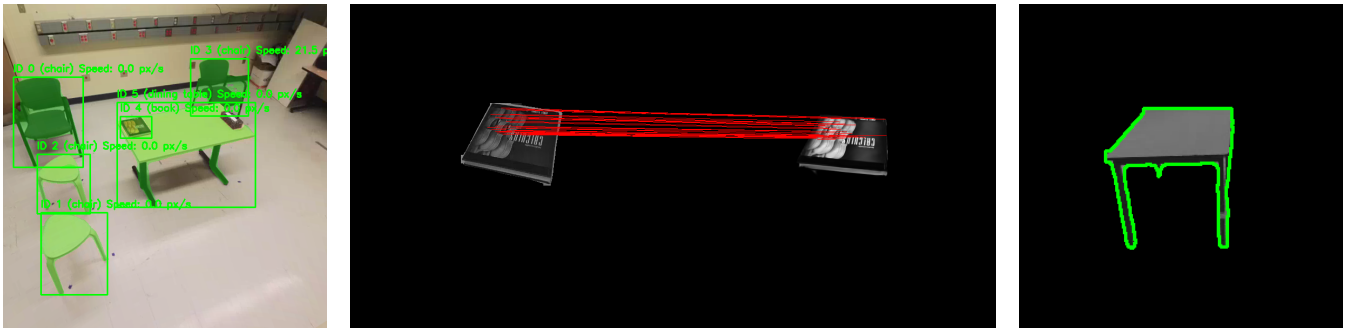
Figure 4: (a) Raw scene object detections produced by YOLOv11seg. (b) The SAGA user interface, used to segment a chair from a trained Gaussian splat representation.

result, naively applying YOLO for tracking can fail when an object's class is inconsistently predicted across frames or when multiple instances share the same class label. To address this, we developed an object tracking layer with identity management on top of YOLOv11. At the start of a telepresence session, the system displays the first frame processed by YOLO to the users. The users are then prompted to select which detected objects they would like to track, which should be a subset of the objects pre-segmented during the scene preparation phase. Each selected object is assigned a unique ID. For each subsequent frame, we apply a lightweight nearest-neighbor association algorithm to match current YOLO detections with the latest known positions (i.e., the center points of the bounding boxes) of tracked objects. A threshold of 100 pixels is applied to avoid associating selected objects with unrelated detections from the first frame. We further classify each object's visibility state based on changes in its bounding box: if the size changes exceed a threshold, the object is marked as *partially occluded*; if no match is found within the threshold, it is marked as *fully occluded*. This threshold is user customizable, while we set it to 20% by default. In addition to identity preservation, we estimate the motion of each object by measuring its frame-to-frame displacement, which will later be used to trigger 3D pose estimation for that object. While alternatives to our YOLO adaptation exist (e.g., ByteTrack [81]), our approach is more specifically designed for our needs. Figure 5a shows a frame with objects tracked by this adapted YOLO tracker.

4.4.2 3D Object Tracking with a Splat-Rendering Optimizer. We introduce a novel splat-rendering optimizer that enables 4-DOF 3D object tracking from a monocular RGB camera within a scene reconstructed using Gaussian Splatting. Intuitively, our optimizer estimates an object's 3D pose by aligning its virtual appearance—rendered by projecting its Gaussian Splatting copy onto the same rectified texture used by YOLO—with its physical appearance in the YOLO video feed. This tracking method is tailored specifically for the *GaussianNexus* telepresence experience. Although it is not intended as a standalone contribution to the computer vision pose tracking literature, we hope it can inspire future work on tracking within Gaussian Splatting frameworks.

Physical-Virtual Object Association: At the beginning of the very first telepresence session, our system automatically associates the physical objects selected by the user with their corresponding Gaussian Splatting copies. For each selected object, we back-project the center of its 2D bounding box into a 3D ray within the Unity scene. Since the Gaussian Splatting scene has already been aligned with the physical world, these rays intersect with the respective Gaussian Splatting copies of the objects, allowing the system to establish the association.

Optimizer Initialization: When the 2D object tracker detects that an object has moved, it triggers the optimizer to update the 3D pose of the corresponding Gaussian Splatting copy. A key insight here is that the optimizer estimates the object's position in camera space. Since the object's 2D location—provided by YOLO—already constrains its X and Y position relative to the camera, the optimization search space can be reduced primarily to the Z-axis (i.e., depth). To account for potential inaccuracies in the tracked pose of the 360° camera, our optimizer still estimates the X and Y components, but restricts them to within 5 cm of the camera's Z-axis. This bounds lateral drift while maintaining flexibility. Then, at the start of each 3D pose estimation, we initialize the optimizer's X and Y positions based on the current YOLO detection, while the Z position and rotation are initialized using their latest known values from previous updates. The position vector is transformed into camera space,



(a) (b) (c)

Figure 5: (a) Object IDs corresponding to YOLOv11 tracking results. (b) Keypoint correspondences between the physical book captured by the camera (left) and the Gaussian splat rendering of the book (right), used to achieve rotation alignment. (c) Contour extracted from a camera image using the YOLOv11 segmentation mask.

while the rotation vector, represented in Euler angles, is preserved in world space.

Optimization with Geometry Similarity: With the proposed pose at each optimization step, the optimizer compares the 2D appearance of the physical object and its Gaussian Splatting copy from the perspective of the camera.

The 2D physical appearance is obtained from YOLO. We used the YOLOv11m-seg model[12] for this purpose, which simultaneously returns an object's bounding box and its segmentation mask. The tracked object's 2D appearance is thus its segmented image from the YOLO video feed. The 2D virtual appearance of the tracked object is obtained by rendering the Gaussian Splatting copy of the object. Since Unity maintains the 360° camera's extrinsic and intrinsic parameters, it is able to project the tracked object's Gaussian Splatting copy to the camera plane in the same way as we obtained the YOLO video feed (see Section 4.4.1). Note that we also use the geometry similarity to filter false movement detection by the YOLO tracker. If the YOLO tracker initiates a pose estimation without significant difference in geometry (checked by a Chamfer distance [3] threshold, see below), our system terminates the pose estimation.

Geometry is measured using two complementary approaches: keypoint matching and contour alignment. For keypoint matching, we extract keypoints from the tracked object's physical and virtual appearances using SuperPoint [10], and perform feature matching with LightGlue [42]. For contour alignment, we use OpenCV[13] to extract contours from both appearances (Figure 5c shows an example contour), and then compute the Chamfer distance [3] between them. However, we found that directly optimizing with a combined loss based on keypoint matching and contour alignment does not perform well in practice, for two main reasons. First, we found that significant differences in object orientation led to a large number of keypoint mismatches, especially when dealing with objects with plain textures. Second, although contour alignment is generally effective for position tracking, it is relatively insensitive to object rotation. Objects with different orientations can produce similar Chamfer distances, which occasionally causes the optimizer to get stuck in local minima.

To overcome the above issues, we developed a staged optimization process. We observed that when object poses align, SuperPoint [10] and LightGlue [42] consistently return the most keypoints and matches. Therefore, we first coarsely search the rotation space (at 45° steps) to find the best initial rotation. Each candidate pose is evaluated using the following loss function, which penalizes both the low number of keypoint matches and the presence of outliers:

Let:

- Ry be the candidate rotation
- $\text{MatchCount}(Ry)$ be the number of keypoint matches for a given rotation Ry
- $\text{InlierRatio}(Ry)$ be the fraction of initial matches that remain after RANSAC with a re-projection threshold of 2px.

The **keypoint matching loss** is defined as:

$$L_{keypoint}(Ry) = -\text{MatchCount}(Ry) \cdot \text{InlierRatio}(Ry) \quad (2)$$

To compute the initial rotation Ry, the optimizer ultimately computes:

$$Ry^{(0)} = \arg\min R_{y \in \{0°,45°,\ldots,315°\}} L_{\text{keypoint}}(R_y) \quad (3)$$

This rotation sampling technique also guarantees that the optimizer can converge to an approximately correct rotation within 1 iteration, regardless of the orientation difference between the physical and virtual object. Figure 5b shows an example of an optimal keypoint match between the physical and virtual appearance.

We also find that this initialization process becomes costly if we traverse all three rotation axes. Given that most objects in typical telepresence scenarios rest on horizontal planes, we constrain our optimizer to estimate only the Y-axis rotation. While this limits our tracking to 4 degrees of freedom (4-DOF), it enables low-latency optimization and remains effective for common collaborative use cases. After determining the initial rotation, our optimizer proceeds to refine the pose using contour alignment. While combining Chamfer distance with keypoint matching might improve accuracy, we

[12]https://docs.ultralytics.com/tasks/segment/
[13]https://github.com/opencv/opencv

opt to use only the Chamfer distance in this refinement step to ensure the entire optimization remains within a 1-second budget. This choice provides a reasonable balance between accuracy and computational efficiency.

Let:

- ProjContour be the set of 2D points from the projected contour of the Gaussian Splatting copy.
- ObsContour be the set of 2D points from the observed contour (e.g., from segmentation).

The **contour alignment loss** is defined as:

$$L_{\text{contour}} = \frac{1}{|\text{ProjContour}|} \sum_{p \in \text{ProjContour}} \min_{q \in \text{ObsContour}} \|p - q\|^2 + \frac{1}{|\text{ObsContour}|} \sum_{q \in \text{ObsContour}} \min_{p \in \text{ProjContour}} \|p - q\|^2 \tag{4}$$

To optimize the **contour alignment loss**, we use SciPy's implementation of the Powell [69] method[14]. We restrict y rotation to be ±22.5 degrees around the rotation estimated by keypoint matching, and limit translation to ±5cm in x and y, with ±0.5m in z. Since we provide the optimizer with strong initial estimates, we find these bounds balance accuracy and performance, allowing the optimizer to converge to an accurate solution quickly.

Smoothing Results: We apply a Kalman filter [40, 71] to smooth the pose updates. Based on empirical tuning, we set the process and measurement standard deviation to 0.3 and 0.1 respectively, providing a balance between temporal smoothing and latency. The optimizer continues to refine the object's pose while it is in motion. Since the Gaussian Splatting copies can be freely transformed, we linearly interpolate both position and rotation between successive updates. This ensures that the remote VR user perceives smooth and continuous object motion, even if pose updates arrive with slight delays.

Tracked Object Recovery between Sessions: Between telepresence sessions, *GaussianNexus* can recover and re-track objects even if they have been moved. The system stores a list of objects tracked in the previous session. At the start of the next session, when the system prompts the user with the first YOLO detection frame (see the "Physical–Virtual Object Association" section), the local user only needs to select the same objects for *GaussianNexus* to resume tracking. These objects do not need to be in their original positions, as the optimizer initializes the X and Y positions based on YOLO detections. This allows the system to automatically discover the corresponding 3D ray intersecting with the physical object and move its Gaussian Splatting copy accordingly. This cross-session object recovery capability enables users to reuse the same scene preparation across multiple telepresence sessions, significantly reducing setup overhead over time. We further discuss the system's support for cross-session continuity in Section 8.

[14]https://docs.scipy.org/doc/scipy/reference/optimize.minimize-powell.html

5 System Evaluation

We evaluated our system's performance by measuring its quality of rendering (i.e., in splat count and rendering frame rate), object tracking accuracy, object update rate and latency.

5.1 Rendering Quality

On our machine with an NVIDIA GeForce RTX 4090, tethered to a Meta Quest 3 (2K resolution per eye), we render the full experimental scene (Figure 3a) containing 1.4 million Gaussian splats at approximately 75 fps. By contrast, state-of-the-art methods like 4DGS [73] achieve similar frame rates with only 50,000 splats scene-wide at 800×800 resolution (using an RTX 3090, which is slightly less powerful); this is comparable to the number of splats in one chair of our scene (60,000, Figure 4b). This comparison highlights the higher visual fidelity potential of our approach, which updates individual objects on top of a static scene reconstruction, rather than reconstructing the entire scene dynamically.

5.2 Object Tracking Accuracy

5.2.1 Setup and Ground Truth. The performance of the 3D tracking system was evaluated at two different spatial scales: a room-scale setup involving a piece of furniture (a chair), and a table-scale setup involving a book. Since we aligned the Gaussian Splatting scene with the physical world, we use the original position and rotation of a Gaussian Splatting copy as the reference for ground truth. To ensure that the alignment between the physical and virtual world is consistent, we check that the scanned physical objects used for evaluation are unmoved at the beginning of the evaluation. Then, by offsetting the targeted objects with carefully measured distances in both the virtual and physical world, we obtain 6 ground truth poses for both the room-scale and table-scale setup. The 6 positions altogether are parallel to the x-z plane of the virtual scene. Since the positional optimization is conducted in the camera space, moving objects through these positions covers 3DOF positional translation, and a separate rotation experiment covers rotation around the y-axis. For the room-scale evaluation, a chair was tracked at six distinct ground truth positions, spaced 1m apart. For the table-scale evaluation, the 6 positions are spaced 22cm apart on the z-axis and 26cm apart on the x-axis. At each position, rotational accuracy was measured at four orientations offsets: 0°, 90°, 180°, and 270°.

5.2.2 Results. The results are summarized in Tables 1 and 2. For the room-scale evaluation, we achieve an average world-space offset of 9.9 cm and a average rotation error of 5.7°, which should be adequate to support room-scale activities. In Figure 6, we further illustrate the positional tracking accuracy of our system through a scatter plot. In the supplemental material, we also include the same plots in 3D (including the Y-position).

The evaluation on the tabletop scale shows better performance, with an average world offset of 7.5 cm and rotation error of just 3.2°. In our evaluation, we found that the performance deteriorates with distance for both scenarios (P2 is the furthest from the camera for the room-scale scenario, and P3 is the furthest for the table-top scenario). We believe this is because, 1) the YOLO segmentation performance reduces with further distance, and 2) the features and contours become less representative when objects appear smaller on

Table 1: Average Scene-Scale Tracking Accuracy Across Six Ground Truth Positions

X (cm)	Y (cm)	Z (cm)	World Offset (cm)	Rotation (°)
5.6 ± 6.3	5.3 ± 5.1	2.7 ± 4.6	9.9 ± 4.4	5.7 ± 12.0

Table 2: Average Tabletop-Scale Tracking Accuracy Across Six Ground Truth Positions

X (cm)	Y (cm)	Z (cm)	World Offset (cm)	Rotation (°)
5.0 ± 1.7	2.2 ± 1.0	2.9 ± 1.9	7.5 ± 1.7	3.2 ± 4.6

Table 3: Average object tracking update rate and end-to-end latency

Object	Optimization (s)	Update Rate (fps)	End-to-End Latency (s)
Chair	0.436 ± 0.016	2.30	0.957 ± 0.067
Stool	0.388 ± 0.013	2.58	0.900 ± 0.047
Table	0.463 ± 0.015	2.159	1.026 ± 0.056
Overall	0.429 ± 0.034	2.33	0.951 ± 0.078

the camera. For the room scale evaluation, the standard deviation of tracking error is much higher. We hypothesize that this is because the chair used for the room scale scenario poses an additional challenge as it has a plainer texture (mostly gray). However, each pose estimate is independent, so errors do not accumulate and are quickly corrected in subsequent updates. Meanwhile, in the tabletop scale evaluation, the camera was positioned closer to the object, potentially contributing to improved tracking reliability as well.

5.3 Object Update Rate and Latency

5.3.1 Measurement Setup. To assess the real-time performance of *GaussianNexus*, we conducted update rate and end-to-end latency measurements with objects of different sizes, namely a chair, a stool, and a table. Note that the update rate determines how frequently the remote user will see the tracked object's position being updated. The end-to-end latency is the time between the movement of the physical object, and the subsequent movement of its virtual counterpart. To measure the object update rate, we instrumented our optimizer to capture precise timestamps at each processing stage across 100 iterations of optimization for each object. End-to-end latency was measured by recording the physical manipulation of the objects and the screens of the computers running the optimizer and Unity with a physical camera at 60 fps. For each object, we recorded 15 physical manipulations. Latency was then calculated by counting the frames between the object's motion and the respective updates in Unity.

5.3.2 Results. We summarize our measurements in Table 3. We found that the complete optimization pipeline on average takes 0.429 seconds, which translates to a update rate of 2.3 frames per second (see Table 3). End-to-end latency measurements show slightly more substantial delays, with total observed latency measurements ranging from 0.9 to 1.0 seconds. We attribute much of this latency to video processing and transmission delays from our commodity 360° camera (approximately 300 ms) and object movement detection thresholds.

Our analysis suggests that large objects consistently take longer to process. We hypothesize that this stems from two factors. First, large objects occupy more pixels in the 640x640 camera frame, generating increased numbers of keypoints and generally requiring more non-trivial computation during chamfer distance calculation. Second, due to increased mass, users generally accelerate larger objects slower, requiring additional frames to exceed the motion threshold to trigger object pose optimization.

6 Applications Scenarios

We demonstrate *GaussianNexus*'s practical utility by situating it in potential application scenarios for both 2D and 3D collaboration.

6.1 Immersive Telepresence with Live Surfaces

Physical surfaces are common collaborative spaces that naturally provide haptic feedback [74, 75], where people often exchange ideas on documents or screens [22, 31], assemble artifacts [20], and play board games [27]. Prior research has enabled surface sharing using projectors [31] and mixed reality headsets [22], allowing users to annotate directly on physical screens and whiteboards, or collaborate using planar objects such as cards or tangram puzzles. Extending from prior work, *GaussianNexus* extends surface-based collaboration by situating the remote user inside a fully walkable and photo-realistic scene reconstruction, while simultaneously supporting multiple live surfaces through our 2D dynamic update feature. The users can enjoy a movie together via shared projection, discuss shared content on a screen, and gather around a desk to collaborate on physical materials. Figure 8 illustrates a more concrete scenario in the education context: a local user plays with tangram puzzles on a live desk surface, while a remote instructor guides them in forming specific shapes (e.g., a fish).

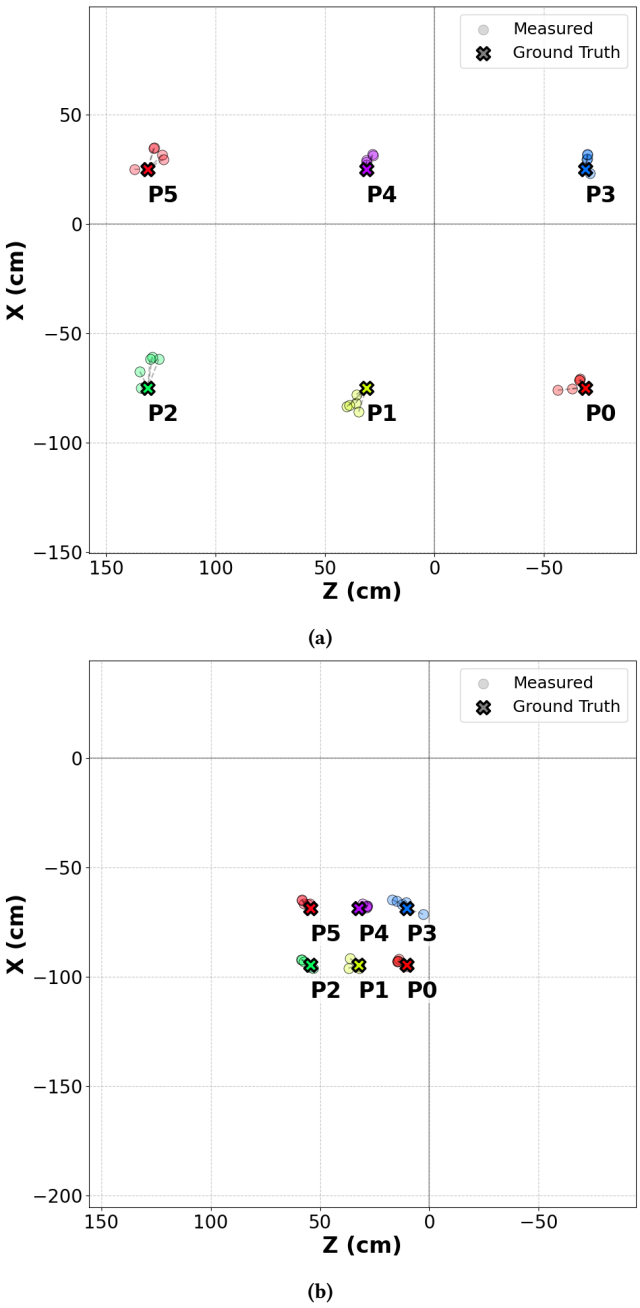


Figure 6: Positional tracking accuracy at two spatial scales. (a) Room-scale evaluation results with tracked chair positions relative to ground truth, with each point spaced 1 m apart. (b) Tabletop-scale evaluation results for a tracked book, with ground truth positions spaced 22 cm along the z-axis and 26 cm along the x-axis. Note: Both plots share the same scale, which makes the tabletop setup appear smaller.

Figure 7: The starting layout of the room for the user study.

6.2 From Tabletop to Room-Scale Collaboration

GaussianNexus supports multi-scale 3D collaboration with photo-realistic quality at multiple scales, ranging from tabletop activities (e.g., virtual/physical chess playing) to room-scale collaboration (e.g., layout planning [43, 80], storage management [9, 18], immersive gaming [82, 83], and remote design and prototyping [15, 50]). However, common scene representation media such as 360° video restrict user navigation, preventing free movement within the environment. Meanwhile, point cloud reconstruction lacks sufficient resolution and suffers from occlusions, limiting visual fidelity—especially when objects are viewed from arbitrary angles.

In Figure 9, we demonstrate a room layout planning scenario. The local user selects several pieces of furniture they intend to set up for a meeting, enabling *GaussianNexus* to track their movements in real time. A remote instructor joins via telepresence, appearing as an embodied virtual avatar, and guides the local user through the setup process.

The remote instructor is equipped with standard collaborative tools such as ray pointers and annotations. In addition, because they are free to move naturally within the reconstructed environment, they can use body language as a communicative modality. For example, rather than pointing or annotating, the instructor can simply walk to a location and ask the local user to move a piece of furniture there. As the local user moves the furniture, *GaussianNexus* continuously updates their poses within the Gaussian Splatting scene, allowing the remote instructor to perceive live changes.

7 User Study

We conducted a user study on *GaussianNexus*. With a room layout planning task (similar to the one described in Section 6.2), we evaluated our system's usability and presence, and gathered feedback.

7.1 Participants

We recruited 9 participants (4 identifying as male and 5 as female, average age 25.5) from the students at the local university. All participants had some prior experience with remote collaborative

(a) (b) (c)

Figure 8: This example demonstrates the use of 2D cutouts to enable a remote user to guide a local user through solving a tangram puzzle. (a) The initial configuration of the puzzle as seen from the AR user's perspective. (b) The remote VR user provides step-by-step instructions for placing the tangram pieces. (c) The completed puzzle, viewed from the remote user's perspective.

Figure 9: This demonstration showcases the system in a room layout planning scenario. Initially, the remote user employs 2D annotations (the light green line in the first and fourth image) to indicate the desired position for a piece of furniture. The local user then relocates the furniture accordingly, with GaussianNexus actively tracking the movement. Finally, the alignment between the physical and remote environments is illustrated.

tools (e.g. Google Docs) and video communication tools (e.g. Zoom). All participants had prior exposure to either AR or VR headsets.

Our study was reviewed and approved with the institutional ethical board. Each participant was compensated with $16.

7.2 Study Procedure

Upon arrival, the experimenter greeted the participant and asked them to review and sign a consent form. The participant then completed a brief tutorial to familiarize themselves with the Meta Quest 3 headset. After the tutorial, the experimenter and the participant moved into two separate rooms divided by a wall.

The experimenter was situated in a room designated as the "local physical space", which had been previously reconstructed using Gaussian Splatting and prepared through the scene preparation process (see Section 4.2). Figure 7 shows the layout of this space, which includes a desk, two chairs, and two stools—all configured to be tracked by our system. The participant assumed the role of the remote VR instructor, becoming telepresent in the Gaussian Splatting scene and instructing the experimenter to set up the room.

The participant was first encouraged to explore the scene freely. They were then given the following task prompt: "Suppose you are going to have a meeting tomorrow with two classmates and a professor. Please instruct your local collaborator to set up the space." We intentionally avoided providing a specific target layout to allow participants to express their own spatial preferences.

The experimenter followed the participant's instructions to adjust the furniture accordingly. Once the participant indicated that they were satisfied with the layout, the experimenter terminated the application and invited the participant to complete the System Usability Scale (SUS), the Slater-Usoh-Steed Presence Questionnaire [61], and a custom questionnaire (see appendix) designed to gather qualitative feedback on their experience.

7.3 Results

We summarize the participants' response to the SUS and the presence questionnaire in Figure 10 and Figure 11. We reversed the scores and wording of negatively worded items (using 5 minus the original score) in the SUS questionnaire to show better consistency, although participants filled out the original version of the questionnaire. Following the approach in [4], we report a SUS score of 85 for our system, which is considered "Excellent". Combining the scores and the qualitative feedback we gathered, we group our findings on the system's usability, and collaborative awareness below.

7.3.1 Ease of Use and Learnability. Participants found our system easy to use (SQ3: Mean = 4.44, Std = 0.53), and most reported that they think people were able to learn it quickly (SQ7: Mean = 4.67,

Std = 0.5). In general, the participants described the experience as "smooth" and "enjoyable":

> *P1: The system was pretty smooth and enjoyable.*
>
> *P4: Compared to 2D tools, it is more enjoyable since the interaction is more natural and similar with what we used to do in real world.*

7.3.2 Latency and Consistency. Despite some technical limitations in our current implementation—specifically, the latency caused by runtime optimization and objects occasionally not updating along smooth trajectories—the majority of participants (7 out of 9) found the system consistent (SQ6: Mean = 4.11, Std = 1.05). While latency and some inconsistencies due to imperfect tracking are noticeable, these issues were not significant enough to negatively impact task performance or the overall user experience:

> *P8: the lag was noticeable, but it didn't break my immersion too much… it felt very fun and real.*
>
> *P5: the objects ended up in the right position before I wanted to check with the AR partner.*

P2 viewed the interaction as a generative process and stated,

> *P2: I don't feel much delay or lag while performing the tasks… the reconstructed chair in the 3D space is generated soon.*

7.3.3 Presence and Realism. Participants felt a strong sense of "being there" in our GaussianNexus environment (PQ2: Mean = 4.78, Std = 0.44), with many expressing that the experience felt more like a place they visited physically rather than something they merely viewed on a screen (PQ3: Mean = 4.44, Std = 0.73). From the users' response, we attribute this to the improved visual fidelity, depth perception, and free navigability.

> *P7: The scene was coherent and real.*
>
> *P2: [The system] allows me to walk around the room and explore surroundings in a much more interactive and detailed way. [The system has] better depth and visual spatial relationship perception as well.*
>
> *P5: [The system] is much more efficient creates a better sense of being at that environment.*

7.3.4 Interactivity and Collaborative Awareness. Our system's enhanced presence and realism also improved interactivity and collaborative awareness. The participants generally reflected that their communication is efficient, natural, and enjoyable:

> *P8: [I can] get to the places in the virtual space where I wanted to go, and point exactly to where I wanted the AR user to look.*
>
> *P4: look and move around the environment without asking my AR partner to "move the camera"... This improves the efficiency and enjoyment.*
>
> *P7: I can envision how seeing someone close to me, like a loved one, in an embodied state would be more meaningful than just over Zoom.*
>
> *P7: Therefore, my pointing directly and moving directly, it is more clear and therefore more efficient to communicate. This added freedom aligns the communication dimension more with the spatial dimension.*

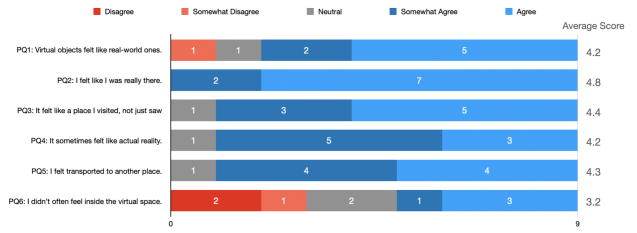


Figure 10: System Usability Scale Five-Point Likert Scale Results: The negative items are inverted (QS2, QS4, QS6, QS8, QS10) and each question is shortened for better visibility; the original questions and raw user responses can be viewed in the supplemental material.

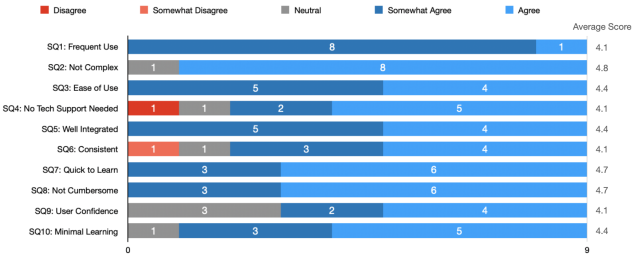


Figure 11: Presence Five-Point Likert Scale Results: Each question is shortened for better visibility; the original questions and raw user response can be viewed in the supplemental material.

8 Discussion and Future Work

Here we reflect on our system's technical contributions and discuss future improvements, drawing results and observations from the system and user evaluation.

8.1 Offloading Online Retraining to Scene Preparation

A key insight behind the design of *GaussianNexus* is the decision to offload the need for on-the-fly retraining of Gaussian Splatting to the scene preparation phase. By pre-segmenting the "live" components of the environment—effectively creating Gaussian Splatting copies of interactable physical objects—we enable those objects to be independently tracked and updated during telepresence. This design eliminates the need for runtime retraining or the fusion of neural rendering with live point cloud streams, which often results in degraded visual quality.

Compared to prior work in neural rendering-based telepresence, such as SharedNeRF [58], our approach scales to room-scale interactions, supports full navigability, preserves photorealistic visual quality throughout the session, and reduces perceived latency during collaboration. As revealed in our user study, our system was effective in assisting the participants to complete the task of setting up the room. Meanwhile, as revealed by the presence questionnaire, most participants felt strong presence and had the sense of being

in the real, physical environment. We believe that the navigability and photorealistic quality of our system is the key to effectiveness, presence, and realism. We observed that some users walked or teleported around the scene, inspecting it in the same way as what is naturally possible in the real world. We believe the ability to use natural communication methods also aid ease of use and learnability, which is highly rated in our SUS questionaire (SQ2, SQ3, SQ7, SQ9, and SQ10). In particular, our system reduces the reliance on common VR modalities such as hand pointers. In the context of the room setup task, a user could directly walk to a location and instruct the local user to move furniture there.

Our approach, however, introduces a trade-off: the local user must complete a scene preparation phase. As outlined earlier, this phase consists of (a) capturing the scene (3 minutes), (b) training Gaussian Splatting [34] and SAGA [8] (about 1 hour), and (c) semi-automatic 3D object segmentation (about 30 minutes for 10 interactable objects). Among these, the capturing and training stages are necessary for any telepresence application based on neural rendering. It is also possible to incorporate faster Gaussian Splatting training schemes, e.g. InstantSplat [11], to reduce this initial delay.

Therefore, the current bottleneck lies in the 3D object segmentation step. While SAGA provides a strong foundation, automatically applied segmentation masks often leave behind "floaters"—redundant splats that require manual cleanup. A useful future extension would be to fully automate the floater-cleaning step by detecting and removing sparse and unconnected blobs of splats.

8.2 Reusability and Continuity Across Telepresence Sessions

To alleviate the overhead of scene preparation, *GaussianNexus* is designed to support reuse of a trained and configured scene across multiple telepresence sessions. As described at the end of Section 4.4.2, the system can recover and track previously configured objects, even if they have been moved between sessions. Currently, the user must manually identify these objects, with assistance from a stored object list in Unity, so that they can be re-associated with the YOLO-based 2D object tracker. While this step only takes a few seconds, it introduces extra friction. In future work, we aim to automate this process by matching YOLO-detected objects with previously tracked Gaussian Splatting copies using shape similarity.

At present, *GaussianNexus* does not support tracking newly added objects that were not included in the original Gaussian Splatting reconstruction. Similar to the object scanning pipeline in VirtualNexus [23], future systems could embed a Gaussian Splatting scanner directly in the AR client, allowing the local user to scan and register a new object in-situ. Notably, reconstructing individual objects is simpler and more efficient than scanning an entire environment. For example, Yang et al. introduced *GaussianObject* [76], which enables high-quality reconstruction of Gaussian Splatting objects from as few as four input images.

8.3 One-shot 3D Object Tracking

While the latency of our system is significantly reduced and smoothed compared to prior research in telepresence based on nueral rendering [58], the participants in our user study all found it "noticeable". This is to be expected because a scene update rate of 1-second, even if smoothed and interpolated, is still far from the update rate of typical video streaming (e.g., 30 fps). The scene update latency of *GaussianNexus* is mainly due to the object-tracking optimizer. Here, we reflect on its implementation and propose potential improvements for future work.

The 3D object tracking module is the key enabler of Gaussian Splatting scene updates in *GaussianNexus*. This module was necessary due to the lack of zero-shot 3D pose estimation methods for arbitrary objects using only monocular RGB input, without category-specific fine-tuning or additional datasets.

Our approach adopts a one-shot tracking strategy, leveraging the information captured during the Gaussian Splatting scene reconstruction. As demonstrated in our application scenarios and user study, this module effectively supports users in completing collaborative tasks. We constrain each object pose estimation to within 1 second, enabling fluid updates and continuous interaction. Since users typically interact with at most two objects simultaneously, our system remains efficient even in multi-object scenarios—a trend consistent with our study observations.

While achieving robust zero-shot tracking remains out of scope, improving the efficiency and robustness of our pipeline is a promising direction. One existing bottleneck is the repeated 2D projection of an object's Gaussian Splatting copy at each optimization step, which takes approximately 4 ms per frame. Future work could precompute these projections across a range of possible poses during scene preparation. Although this would lengthen scene preparation time, it could significantly improve runtime performance and enable higher frame-rate tracking—further enhancing temporal accuracy. Another avenue for future optimization would be to explore differentiable contour alignment scoring (i.e., as opposed to Chamfer distance), which would enable the use of a more efficient optimizer such as Adam [36].

As with most vision systems, occlusion remains a key challenge. Our current solution simply pauses pose estimation for occluded objects, which can lead to tracking inconsistencies. During our user study, such inconsistencies often occurred when one object blocked another or when the local user walked in front of tracked items. Such occlusions are the main source of our tracking error. While we found our tracking accuracy sufficient for room-scale activities, future exploration is needed to support tasks with higher precision demands. A potential future workaround is to incorporate occlusion-aware rendering directly into the Gaussian Splatting scene. If tracking runs at a sufficiently high frame rate, occlusions in the physical scene could be mirrored by occlusions in the virtual view, allowing the tracking system to rely on consistent virtual visibility. Object tracking methods based on one-shot CAD scans are more mature [47, 72], and such approaches could be adapted for Gaussian Splatting in the future.

While this work primarily contributes a room-scale telepresence system with real-time neural rendering, further refining a general-purpose Gaussian Splatting-based 3D tracker remains an exciting direction for future research.

8.4 Broader Empirical Evaluations

In this work, we focused our user evaluation primarily on the usability of *GaussianNexus*. As *GaussianNexus* represents a major step

toward hyper-realistic telepresence, it is important to understand its impact on users' sense of presence, comfort, task efficiency, and collaborative awareness. In future work, we plan to conduct comparative studies between *GaussianNexus* and other representative systems (e.g., using 360° video or RGB-D reconstruction) across diverse task types (e.g., instruction, entertainment, prototyping), supported by additional surveys and conversational analysis. We also aim to scale the system to support group collaboration with more users (i.e., $N > 3$). It would also be interesting to extend our system to support larger spaces such as exhibition halls. As the 2D reprojected resolution and 3D tracking performance degrade in regions farther from the 360° camera, our system currently allows users to reposition the 360° camera mid-session to better support 2D and 3D updates in specific areas. In the future, incorporating multiple 360° cameras would be a practical approach for covering larger spaces.

9 Conclusion

We introduced GaussianNexus, a novel AR/VR telepresence system that enables real-time, room-scale interaction with photorealistic scenes through Gaussian Splatting and live video integration. By leveraging prior scene training and persistent 3D object tracking, *GaussianNexus* overcomes key limitations of current neural rendering based telepresence systems, eliminating the need for constant retraining while maintaining visual fidelity and spatial interactivity. Our system enables remote users to move freely through virtualized, photorealistic real-world environments while interacting with dynamic content. With on average 2.33 fps object update rate and 0.95 second end-to-end latency and promising early user study results, GaussianNexus represents a step forward in future hyper-realistic telepresence.

Acknowledgments

This work was supported in part by the Natural Science and Engineering Research Council of Canada (NSERC) under Discovery Grant RGPIN-2019-05624.

References

[1] Adel Ahmadyan, Tingbo Hou, Jianing Wei, Liangkai Zhang, Artsiom Ablavatski, and Matthias Grundmann. 2020. Instant 3D Object Tracking with Applications in Augmented Reality. arXiv:2006.13194 [cs.CV] https://arxiv.org/abs/2006.13194

[2] Adel Ahmadyan, Liangkai Zhang, Jianing Wei, Artsiom Ablavatski, and Matthias Grundmann. 2020. Objectron: A Large Scale Dataset of Object-Centric Videos in the Wild with Pose Annotations. arXiv:2012.09988 [cs.CV] https://arxiv.org/abs/2012.09988

[3] Ainesh Bakshi, Piotr Indyk, Rajesh Jayaram, Sandeep Silwal, and Erik Waingarten. 2023. A Near-Linear Time Algorithm for the Chamfer Distance. arXiv:2307.03043 [cs.DS] https://arxiv.org/abs/2307.03043

[4] Aaron Bangor, Philip Kortum, and James Miller. 2009. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. *J. Usability Studies* 4, 3 (may 2009), 114–123.

[5] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2021. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. *CoRR* abs/2111.12077 (2021). arXiv:2111.12077 https://arxiv.org/abs/2111.12077

[6] Bill Buxton. 2009. Mediaspace – Meaningspace – Meetingspace. In *Computer Supported Cooperative Work*. Springer London, London, 217–231. doi:10.1007/978-1-84882-483-6_13

[7] Minghao Cai, Soh Masuko, and Jiro Tanaka. 2018. Shopping Together: A Remote Co-shopping System Utilizing Spatial Gesture Interaction. In *Human-Computer Interaction. Interaction Technologies*. Springer International Publishing, 219–232.

[8] Jiazhong Cen, Jiemin Fang, Chen Yang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. 2023. Segment Any 3D Gaussians. *arXiv preprint arXiv:2312.00860* (2023).

[9] Daniela De Luca and Anna Osello. 2021. BIM and Mixed Reality for the New Management of Storage Area. In *From Building Information Modelling to Mixed Reality*, Cecilia Bolognesi and Daniele Villa (Eds.). Springer International Publishing, Cham, 123–141.

[10] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. 2018. SuperPoint: Self-Supervised Interest Point Detection and Description. arXiv:1712.07629 [cs.CV] https://arxiv.org/abs/1712.07629

[11] Zhiwen Fan, Kairun Wen, Wenyan Cong, Kevin Wang, Jian Zhang, Xinghao Ding, Danfei Xu, Boris Ivanovic, Marco Pavone, Georgios Pavlakos, Zhangyang Wang, and Yue Wang. 2024. InstantSplat: Sparse-view Gaussian Splatting in Seconds. arXiv:2403.20309 [cs.CV]

[12] John Flynn, Michael Broxton, Lukas Murmann, Lucy Chai, Matthew DuVall, Clément Godard, Kathryn Heal, Srinivas Kaza, Stephen Lombardi, Xuan Luo, Supreeth Achar, Kira Prabhu, Tiancheng Sun, Lynn Tsai, and Ryan Overbeck. 2024. Quark: Real-time, High-resolution, and General Neural View Synthesis. arXiv:2411.16680 [cs.CV] https://arxiv.org/abs/2411.16680

[13] Sara Fridovich-Keil, Giacomo Meanti, Frederik Warburg, Benjamin Recht, and Angjoo Kanazawa. 2023. K-Planes: Explicit Radiance Fields in Space, Time, and Appearance. arXiv:2301.10241 [cs.CV] https://arxiv.org/abs/2301.10241

[14] Danilo Gasques, Janet G Johnson, Tommy Sharkey, Yuanyuan Feng, Ru Wang, Zhuoqun Robin Xu, Enrique Zavala, Yifei Zhang, Wanze Xie, Xinming Zhang, Konrad Davis, Michael Yip, and Nadir Weibel. 2021. ARTEMIS: A Collaborative Mixed-Reality System for Immersive Surgical Telementoring. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21, Article 662)*. Association for Computing Machinery, New York, NY, USA, 1–14.

[15] 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 (Sept. 2019), 1–24.

[16] Carl Gutwin and Saul Greenberg. 2002. A Descriptive Framework of Workspace Awareness for Real-Time Groupware. *Comput. Support. Coop. Work* 11, 3 (Sept. 2002), 411–446. doi:10.1023/A:1021271517844

[17] Jukka Häkkinen, Monika Pölönen, Jari Takatalo, and Göte Nyman. 2006. Simulator sickness in virtual display gaming: a comparison of stereoscopic and non-stereoscopic situations. In *Proceedings of the 8th Conference on Human-Computer Interaction with Mobile Devices and Services* (Helsinki, Finland) *(MobileHCI '06)*. Association for Computing Machinery, New York, NY, USA, 227–230. doi:10.1145/1152215.1152263

[18] Steven Henderson and Steven Feiner. 2011. Exploring the benefits of augmented reality documentation for maintenance and repair. *IEEE Trans. Vis. Comput. Graph.* 17, 10 (Oct. 2011), 1355–1368.

[19] Teresa Hirzle, Maurice Cordts, Enrico Rukzio, Jan Gugenheimer, and Andreas Bulling. 2021. A Critical Assessment of the Use of SSQ as a Measure of General Discomfort in VR Head-Mounted Displays. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 530, 14 pages. doi:10.1145/3411764.3445361

[20] Xincheng Huang, Keylonnie L. Miller, Alanson P. Sample, and Nikola Banovic. 2023. StructureSense: Inferring Constructive Assembly Structures from User Behaviors. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 4, Article 204 (Jan. 2023), 25 pages. doi:10.1145/3570343

[21] Xincheng Huang, James Riddell, and Robert Xiao. 2023. Virtual Reality Telepresence: 360-Degree Video Streaming with Edge-Compute Assisted Static Foveated Compression. *IEEE Transactions on Visualization and Computer Graphics* 29, 11 (2023), 4525–4534. doi:10.1109/TVCG.2023.3320255

[22] Xincheng Huang and Robert Xiao. 2024. SurfShare: Lightweight Spatially Consistent Physical Surface and Virtual Replica Sharing with Head-mounted Mixed-Reality. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 4, Article 162 (Jan. 2024), 24 pages. doi:10.1145/3631418

[23] Xincheng Huang, Michael Yin, Ziyi Xia, and Robert Xiao. 2024. VirtualNexus: Enhancing 360-Degree Video AR/VR Collaboration with Environment Cutouts and Virtual Replicas. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) *(UIST '24)*. Association for Computing Machinery, New York, NY, USA, Article 55, 12 pages. doi:10.1145/3654777.3676377

[24] Andrew Irlitti, Mesut Latifoglu, Thuong Hoang, Brandon Victor Syiem, and Frank Vetere. 2024. Volumetric Hybrid Workspaces: Interactions with Objects in Remote and Co-located Telepresence. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 802, 16 pages. doi:10.1145/3613904.3642814

[25] Andrew Irlitti, Mesut Latifoglu, Qiushi Zhou, Martin N Reinoso, Thuong Hoang, Eduardo Velloso, and Frank Vetere. 2023. Volumetric Mixed Reality Telepresence for Real-time Cross Modality Collaboration. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 101, 14 pages. doi:10.1145/3544548.3581277

[26] 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. doi:10.1145/2047196.2047270

[27] Pascal Jansen, Fabian Fischbach, Jan Gugenheimer, Evgeny Stemasov, Julian Frommel, and Enrico Rukzio. 2020. ShARe: Enabling Co-Located Asymmetric Multi-User Interaction for Augmented Reality Head-Mounted Displays. 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, 459–471. doi:10.1145/3379337.3415843

[28] Tianjian Jiang, Xu Chen, Jie Song, and Otmar Hilliges. 2023. InstantAvatar: Learning Avatars From Monocular Video in 60 Seconds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 16922–16932.

[29] Michal Joachimczak, Juan Liu, and Hiroshi Ando. 2017. Real-time mixed-reality telepresence via 3D reconstruction with HoloLens and commodity depth sensors. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction* (Glasgow, UK) *(ICMI '17)*. Association for Computing Machinery, New York, NY, USA, 514–515.

[30] Brennan Jones, Yaying Zhang, Priscilla N. Y. Wong, and Sean Rintel. 2021. Belonging There: VROOM-ing into the Uncanny Valley of XR Telepresence. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW1, Article 59 (apr 2021), 31 pages. doi:10.1145/3449133

[31] Sasa Junuzovic, Kori Inkpen, Tom Blank, and Anoop Gupta. 2012. IllumiShare: sharing any surface. 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, 1919–1928.

[32] Shunichi Kasahara and Jun Rekimoto. 2015. JackIn head: immersive visual telepresence system with omnidirectional wearable camera for remote collaboration. In *Proceedings of the 21st ACM Symposium on Virtual Reality Software and Technology* (Beijing, China) *(VRST '15)*. Association for Computing Machinery, New York, NY, USA, 217–225.

[33] Robert S Kennedy, Norman E Lane, Kevin S Berbaum, and Michael G Lilienthal. 1993. Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. *Int. J. Aviat. Psychol.* 3, 3 (July 1993), 203–220. doi:10.1207/s15327108ijap0303_3

[34] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (July 2023). https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

[35] Rahima Khanam and Muhammad Hussain. 2024. YOLOv11: An Overview of the Key Architectural Enhancements. arXiv:2410.17725 [cs.CV] https://arxiv.org/abs/2410.17725

[36] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG] https://arxiv.org/abs/1412.6980

[37] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. 2023. Segment Anything. arXiv:2304.02643 [cs.CV]

[38] Nikos Kolotouros, Thiemo Alldieck, Enric Corona, Eduard Gabriel Bazavan, and Cristian Sminchisescu. 2025. Instant 3D Human Avatar Generation Using Image Diffusion Models. In *Computer Vision – ECCV 2024*, Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (Eds.). Springer Nature Switzerland, Cham, 177–195.

[39] Junxuan Li, Chen Cao, Gabriel Schwartz, Rawal Khirodkar, Christian Richardt, Tomas Simon, Yaser Sheikh, and Shunsuke Saito. 2024. URAvatar: Universal Relightable Gaussian Codec Avatars. In *SIGGRAPH Asia 2024 Conference Papers* (Tokyo, Japan) *(SA '24)*. Association for Computing Machinery, New York, NY, USA, Article 128, 11 pages. doi:10.1145/3680528.3687653

[40] Qiang Li, Ranyang Li, Kaifan Ji, and Wei Dai. 2015. Kalman Filter and Its Application. In *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*. 74–77. doi:10.1109/ICINIS.2015.35

[41] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. 2024. Gaussian-Flow: 4D Reconstruction with Dynamic 3D Gaussian Particle. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 21136–21145.

[42] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. 2023. LightGlue: Local Feature Matching at Light Speed. arXiv:2306.13643 [cs.CV] https://arxiv.org/abs/2306.13643

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

[44] Zechen Liu, Zizhang Wu, and Roland Tóth. 2020. SMOKE: Single-Stage Monocular 3D Object Detection via Keypoint Estimation. *arXiv preprint arXiv:2002.10111* (2020).

[45] Julieta Martinez, Emily Kim, Javier Romero, Timur Bagautdinov, Shunsuke Saito, Shoou-I Yu, Stuart Anderson, Michael Zollhöfer, Te-Li Wang, Shaojie Bai, Chenghui Li, Shih-En Wei, Rohan Joshi, Wyatt Borsos, Tomas Simon, Jason Saragih, Paul Theodosis, Alexander Greene, Anjani Josyula, Silvio Mano Maeta, Andrew I. Jewett, Simon Venshtain, Christopher Heilman, Yueh-Tung Chen, Sidi Fu, Mohamed Ezzeldin A. Elshaer, Tingfang Du, Longhua Wu, Shen-Chi Chen, Kai Kang, Michael Wu, Youssef Emad, Steven Longay, Ashley Brewer, Hitesh Shah, James Booth, Taylor Koska, Kayla Haidle, Matt Andromalos, Joanna Hsu, Thomas Dauer, Peter Selednik, Tim Godisart, Scott Ardisson, Matthew Cipperly, Ben Humberston, Lon Farr, Bob Hansen, Peihong Guo, Dave Braun, Steven Krenn, He Wen, Lucas Evans, Natalia Fadeeva, Matthew Stewart, Gabriel Schwartz, Divam Gupta, Gyeongsik Moon, Kaiwen Guo, Yuan Dong, Yichen Xu, Takaaki Shiratori, Fabian Prada, Bernardo R. Pires, Bo Peng, Julia Buffalini, Autumn Trimble, Kevyn McPhail, Melissa Schoeller, and Yaser Sheikh. 2024. Codec Avatar Studio: Paired Human Captures for Complete, Driveable, and Generalizable Avatars. In *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (Eds.), Vol. 37. Curran Associates, Inc., 83008–83023. https://proceedings.neurips.cc/paper_files/paper/2024/file/9712b78386cebdc3db7f1a48c2d20edb-Paper-Datasets_and_Benchmarks_Track.pdf

[46] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2021. NeRF: representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (dec 2021), 99–106. doi:10.1145/3503250

[47] Sungphill Moon, Hyeontae Son, Dongcheol Hur, and Sangwook Kim. 2024. GenFlow: Generalizable Recurrent Flow for 6D Pose Refinement of Novel Objects. arXiv:2403.11510 [cs.CV] https://arxiv.org/abs/2403.11510

[48] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 2017. 3D Bounding Box Estimation Using Deep Learning and Geometry. arXiv:1612.00496 [cs.CV]

[49] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Trans. Graph.* 41, 4, Article 102 (July 2022), 15 pages. doi:10.1145/3528223.3530127

[50] 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*. Association for Computing Machinery, New York, NY, USA, 1–12.

[51] 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. doi:10.1145/2984511.2984517

[52] 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.

[53] Thammathip Piumsomboon, Arindam Day, Barrett Ens, Youngho Lee, Gun Lee, and Mark Billinghurst. 2017. Exploring enhancements for remote mixed reality collaboration. In *SIGGRAPH Asia 2017 Mobile Graphics & Interactive Applications* (Bangkok, Thailand) *(SA '17)*. Association for Computing Machinery, New York, NY, USA, Article 16, 5 pages. doi:10.1145/3132787.3139200

[54] Thammathip Piumsomboon, Gun A Lee, Andrew Irlitti, Barrett Ens, Bruce H Thomas, and Mark Billinghurst. 2019. On the Shoulder of the Giant: A Multi-Scale Mixed Reality Collaboration with 360 Video Sharing and Tangible Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–17.

[55] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2020. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[56] Taehyun Rhee, Stephen Thompson, Daniel Medeiros, Rafael Dos Anjos, and Andrew Chalmers. 2020. Augmented Virtual Teleportation for High-Fidelity Telecollaboration. *IEEE Trans. Vis. Comput. Graph.* 26, 5 (May 2020), 1923–1933. doi:10.1109/TVCG.2020.2973065

[57] Shunsuke Saito, Gabriel Schwartz, Tomas Simon, Junxuan Li, and Giljoo Nam. 2024. Relightable Gaussian Codec Avatars. arXiv:2312.03704 [cs.GR] https://arxiv.org/abs/2312.03704

[58] Mose Sakashita, Balasaravanan Thoravi Kumaravel, Nicolai Marquardt, and Andrew David Wilson. 2024. SharedNeRF: Leveraging Photorealistic and View-dependent Rendering for Real-time and Remote Collaboration. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 675, 14 pages. doi:10.1145/3613904.3642945

[59] Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[60] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. 2016. Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision (ECCV)*.

[61] Mel Slater, Amela Sadagic, Martin Usoh, and Ralph Schroeder. 2000. Small-Group Behavior in a Virtual and Real Environment: A Comparative Study. *Presence: Teleoper. Virtual Environ.* 9, 1 (Feb. 2000), 37–51. doi:10.1162/105474600566600

[62] Mel Slater and Sylvia Wilbur. 1997. A framework for immersive virtual environments (FIVE): Speculations on the role of presence in virtual environments. *Presence: Teleoperators & Virtual Environments* 6, 6 (1997), 603–616. https://direct.mit.edu/pvar/article-abstract/6/6/603/18157

[63] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. 2023. NeRFPlayer: A Streamable Dynamic Scene Representation with Decomposed Neural Radiance Fields. arXiv:2210.15947 [cs.CV] https://arxiv.org/abs/2210.15947

[64] Anthony Tang, Michel Pahud, Kori Inkpen, Hrvoje Benko, John C Tang, and Bill Buxton. 2010. Three's company: understanding communication channels in three-way distributed collaboration. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*. ACM, New York, NY, USA. doi:10.1145/1718918.1718969

[65] Theophilus Teo, Ashkan F. Hayati, Gun A. Lee, Mark Billinghurst, and Matt Adcock. 2019. A Technique for Mixed Reality Remote Collaboration Using 360 Panoramas in 3D Reconstructed Scenes. In *Proceedings of the 25th ACM Symposium on Virtual Reality Software and Technology (VRST '19)*. Association for Computing Machinery, New York, NY, USA, 1–11. doi:10.1145/3359996.3364238

[66] Theophilus Teo, Louise Lawrence, Gun A Lee, Mark Billinghurst, and Matt Adcock. 2019. Mixed Reality Remote Collaboration Combining 360 Video and 3D Reconstruction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–14.

[67] Theophilus Teo, Mitchell Norman, Gun A. Lee, Mark Billinghurst, and Matt Adcock. 2020. Exploring Interaction Techniques for 360 Panoramas inside a 3D Reconstructed Scene for Mixed Reality Remote Collaboration. *Journal on Multimodal User Interfaces* 14, 4 (Dec. 2020), 373–385. doi:10.1007/s12193-020-00343-x

[68] 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.

[69] Vassilios S. Vassiliadis and Raúl Conejeros. 2001. *Powell methodPowell Method*. Springer US, Boston, MA, 2001–2003. doi:10.1007/0-306-48332-7_393

[70] Alla Vovk, Fridolin Wild, Will Guest, and Timo Kuula. 2018. Simulator Sickness in Augmented Reality Training Using the Microsoft HoloLens. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–9. doi:10.1145/3173574.3173783

[71] Greg Welch and Gary Bishop. 1995. *An Introduction to the Kalman Filter*. Technical Report. USA.

[72] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. 2024. FoundationPose: Unified 6D Pose Estimation and Tracking of Novel Objects. arXiv:2312.08344 [cs.CV] https://arxiv.org/abs/2312.08344

[73] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 2024. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 20310–20320.

[74] Ziyi Xia, Xincheng Huang, Sidney S Fels, and Robert Xiao. 2025. HaloTouch: Using IR Multi-Path Interference to Support Touch Interactions with General Surfaces. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 548, 17 pages. doi:10.1145/3706598.3714179

[75] Robert Xiao, Julia Schwarz, Nick Throm, Andrew D. Wilson, and Hrvoje Benko. 2018. MRTouch: Adding Touch Input to Head-Mounted Mixed Reality. *IEEE Transactions on Visualization and Computer Graphics* 24, 4 (2018), 1653–1660. doi:10.1109/TVCG.2018.2794222

[76] Chen Yang, Sikuang Li, Jiemin Fang, Ruofan Liang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. 2024. GaussianObject: High-Quality 3D Object Reconstruction from Four Views with Gaussian Splatting. *ACM Transactions on Graphics* (2024).

[77] Jacob Young, Tobias Langlotz, Steven Mills, and Holger Regenbrecht. 2020. Mobileportation: Nomadic Telepresence for Mobile Devices. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 2, Article 65 (jun 2020), 16 pages. doi:10.1145/3397331

[78] Kevin Yu, Alexander Winkler, Frieder Pankratz, Marc Lazarovici, Dirk Wilhelm, Ulrich Eck, Daniel Roth, and Nassir Navab. 2021. Magnoramas: Magnifying Dioramas for Precise Annotations in Asymmetric 3D Teleconsultation. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*. 392–401.

[79] Dingyuan Zhang, Dingkang Liang, Hongcheng Yang, Zhikang Zou, Xiaoqing Ye, Zhe Liu, and Xiang Bai. 2024. SAM3D: zero-shot 3D object detection via the segment anything model. *Science China Information Sciences* 67, 4 (March 2024). doi:10.1007/s11432-023-3943-6

[80] Lei Zhang, Jin Pan, Jacob Gettig, Steve Oney, and Anhong Guo. 2024. VRCopilot: Authoring 3D Layouts with Generative AI Models in VR. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) *(UIST '24)*. Association for Computing Machinery, New York, NY, USA, Article 96, 13 pages. doi:10.1145/3654777.3676451

[81] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. 2022. ByteTrack: Multi-Object Tracking by Associating Every Detection Box. (2022).

[82] Hongyu Zhou, Treshan Ayesh, Chenyu Fan, Zhanna Sarsenbayeva, and Anusha Withana. 2024. CoplayingVR: Understanding User Experience in Shared Control in Virtual Reality. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 8, 3, Article 148 (Sept. 2024), 25 pages. doi:10.1145/3678508

[83] Hongyu Zhou, Pamuditha Somarathne, Treshan Ayesh Peirispulle, Chenyu Fan, Zhanna Sarsenbayeva, and Anusha Withana. 2024. PairPlayVR: Shared Hand Control for Virtual Games. In *Proceedings of the Augmented Humans International Conference 2024* (Melbourne, VIC, Australia) *(AHs '24)*. Association for Computing Machinery, New York, NY, USA, 311–314. doi:10.1145/3652920.3653057