# RIFTCast: A Template-Free End-to-End Multi-View Live Telepresence Framework and Benchmark

Domenic Zingsheim
University of Bonn
Bonn, Germany
zingsheim@cs.uni-bonn.de

Markus Plack
University of Bonn
Bonn, Germany
mplack@cs.uni-bonn.de

Hannah Dröge
University of Bonn
Bonn, Germany
droege@cs.uni-bonn.de

Janelle Pfeifer
University of Bonn
Bonn, Germany
pfeifer@cs.uni-bonn.de

Patrick Stotko
University of Bonn
Bonn, Germany
stotko@cs.uni-bonn.de

Matthias B. Hullin
University of Bonn
Bonn, Germany
hullin@cs.uni-bonn.de

Reinhard Klein
University of Bonn
Bonn, Germany
rk@cs.uni-bonn.de

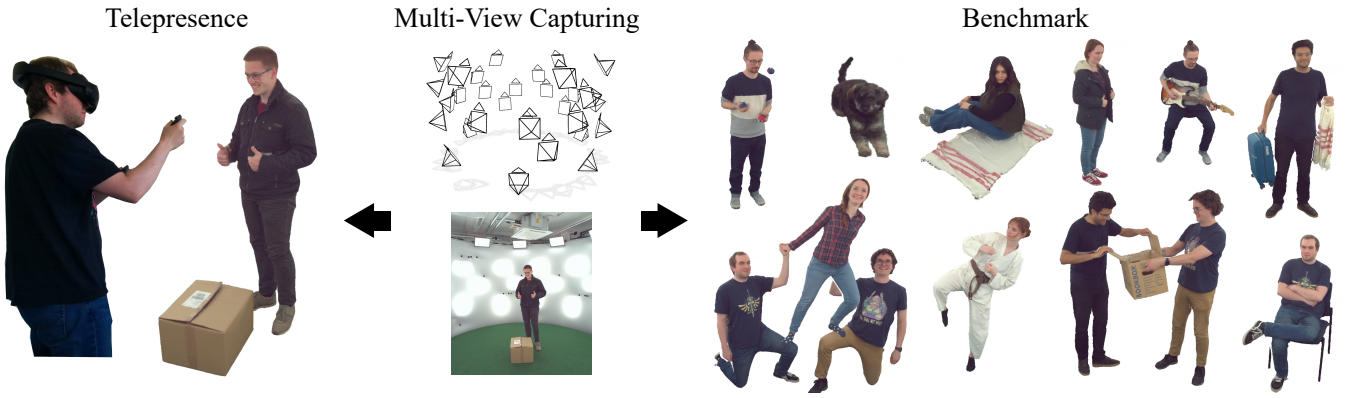


**Figure 1: RIFTCast allows sharing immersive 3D experiences (left) based on multi-view RGB image sequences that are captured from various capturing configurations (middle) and offers a new multi-actor benchmark with complex interactions (right).**

## Abstract

Immersive telepresence aims to authentically reproduce remote physical scenes, enabling the experience of real-world places, objects and people over large geographic distances. This requires the ability to generate realistic novel views of the scene with low latency. Existing methods either depend on depth data from specialized hardware setups or precomputed templates such as human models, which severely restrict their practicality and generalization to diverse scenes. To address these challenges, we introduce *RIFTCast*, a real-time template-free volumetric reconstruction framework that synthesizes high-fidelity dynamic scenes from a multi-view RGB-only capture setup. The framework is specifically targeted at the efficient reconstruction, transmission and visualization of complex scenes, including extensive human-human and human-object interactions. For this purpose, our method leverages a GPU-accelerated client-server pipeline that computes a visual hull representation to select a suitable subset of images for novel view synthesis, substantially reducing bandwidth and computation demands. This lightweight architecture enables deployment from small-scale configurations to sophisticated multi-camera capture stages, achieving low-latency telepresence even on resource-constrained devices. For evaluation, we provide a comprehensive high-quality multi-view video data benchmark as well as our reconstruction and rendering code, including tools for loading and processing a variety of data input formats, to facilitate future telepresence research.

## CCS Concepts

• **Computing methodologies** → **Image-based rendering**; **Reconstruction**; **Volumetric models**; **Mixed / augmented reality**; **Virtual reality**.

## Keywords

Telepresence, Streaming, Dynamic Reconstruction, Novel View Synthesis, Multi-View, Benchmark

**ACM Reference Format:**
Domenic Zingsheim, Markus Plack, Hannah Dröge, Janelle Pfeifer, Patrick Stotko, Matthias B. Hullin, and Reinhard Klein. 2025. RIFTCast: A Template-Free End-to-End Multi-View Live Telepresence Framework and Benchmark. In *Proceedings of the 33rd ACM International Conference on Multimedia (MM '25), October 27–31, 2025, Dublin, Ireland.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3746027.3754789


*MM '25, Dublin, Ireland*

ACM ISBN 979-8-4007-2035-2/2025/10
https://doi.org/10.1145/3746027.3754789

## 1 Introduction

The ability to perceive physical presence in remote environments has been a long-standing area of research, with numerous applications that open up new possibilities to collaborate and interact over long distances. Thanks to the advances and increasing availability of Augmented Reality (AR) and Virtual Reality (VR) technology, the exploration and sharing of immersive 3D experiences have gained significant attention. Crucial factors for enabling such authentic experiences are real-time visualization frame rates, a high responsiveness with a minimal latency, as well as free-form navigation through the virtual environment such that interactions feel natural, smooth, and immediate. However, remote environments are typically captured and shared from a fixed set of cameras and their viewpoints, which necessitates the synthesis of novel views from an on-the-fly estimated representation of the dynamic scene in order to meet these requirements – challenges that are addressed by telepresence methods.

Previous approaches tackled these problems by reconstructing textured 3D meshes of the scene from multi-view video data [36, 42, 52]. These setups rely on available depth data that is additionally provided by the respective camera setups to estimate the scene geometry. In the common use case of human performance capturing, dynamic reconstruction methods have been equipped with prerecorded template models as priors to ensure plausible motions [58, 63]. However, while enabling realistic results, template models inherently restrict the range of applicable scenarios, thereby excluding cases that involve non-human subjects such as animals or unconventional motions and objects. Orthogonal to these methods that aim for reconstructing an explicit 3D model which can be directly visualized with standard rendering tools, Neural Radiance Fields (NeRF) [45] and 3D Gaussian Splatting (3DGS) [34] have demonstrated to provide an astonishing degree of realism in novel view synthesis from color images only, but require extensive offline training times, limiting their applicability for real-time scenarios. This computational bottleneck highlights the need for lightweight approaches and optimized pipelines to enable telepresence within reasonable bandwidth and compute constraints.

In this paper, we present **RIFTCast**, a **R**eal-time, **I**mmersive, **F**ree-viewpoint **T**elepresence system that is particularly tailored for streaming and synthesizing high-fidelity dynamic scenes from various capture configurations including small-scale consumer-grade setups as well as large professional capture stages with controlled illumination. To this end, we designed a template-free, real-time volumetric reconstruction framework that leverages a visual hull-based representation computed from foreground mask images to efficiently select suitable RGB views for rendering. This enables our method to support arbitrary dynamic scene content exhibiting complex human-human and human-object interactions without the necessity of precise temporal tracking of correspondences. We achieve this by an efficient data representation and architectural separation of responsibilities across our hardware components. In order to combat the lack of available code for state-of-the-art telepresence systems, we will release our telepresence framework and toolbox for handling various different dataset input formats. Furthermore, we captured a multi-view video data benchmark featuring diverse scenes and actions, available at https://cg.cs.uni-bonn.de/riftcast.

In summary, our key contributions are as follows:

- A real-time GPU-accelerated reconstruction pipeline that synthesizes novel views via adaptively estimating dynamic geometry and textures directly from multi-view RGB image sequences without relying on depth information or prescanned template models.
- A lightweight, bandwidth-efficient client-server architecture that offloads intensive computations to the server, enabling simultaneous low-latency telepresence across multiple users and a wide range of devices including low-end hardware.
- A new benchmark dataset with complex multi-actor motions and interactions for evaluating real-time telepresence systems under diverse conditions.

The source code is available at https://github.com/vc-bonn/RIFTCast.

## 2 Related Work

*3D Reconstruction.* Due to the success of the seminal KinectFusion work [30, 49], reconstructing static 3D scenes with affordable RGB-D cameras in real time gained significant popularity. To also handle and reconstruct free-form, non-rigid scenes, DynamicFusion [48] represented the scene in a canonical space and estimated a volumetric warp field for transforming to the current live frame. Further extensions explored more robust deformation estimation by extracting color features [29], handling topological changes [59, 60], predicting occluded motion [40], or leveraging additional sensors such as IMU data [79] or high-speed, yet low-resolution image data [25]. In addition, different scene representations such as surfel-based models [22, 35] were considered.

While the aforementioned approaches aim for single-camera scenarios, multi-view RGB-only [24, 64] and RGB-D [14, 15] setups allow to handle more challenging motions during performance capturing. To estimate plausible motions, human priors such as prerecorded and pretrained 3D template models [26, 31, 58, 63] have been leveraged as additional deformation constraints. Furthermore, recent methods also considered explicitly modeling human-human and human-object interactions via separate scene representations [24, 31, 63]. However, these approaches either need long a-priori per-scene training [24], do not generalize to arbitrary scenes [58], or rely on available depth information [14, 15, 31, 63]. Close to our system, some works focused on high-quality texture rendering [14, 44, 64]. In particular, Motion2Fusion [14] uses projective texturing and photo-consistency to achieve sharper texturing results while LookinGood [44] fills holes via neural re-rendering and has been trained with a human head prior to enhance details. Despite these similarities, our system works with RGB-only data and does not impose any constraints on the captured scene content making it applicable to a wider range of scenarios.

*Novel View Synthesis.* Traditional novel view synthesis has been realized via warping of explicit representations such as point clouds based on multi-view stereo [6, 7, 19, 23]. Another line of research focused on silhouette-based approaches that leverage the visual hull as a computationally efficient proxy geometry [9, 12, 50, 71, 77].

Recent progress has been driven by neural scene representations and, particularly, by the seminal Neural Radiance Fields (NeRF) [45] which achieved remarkable visualization results. This led to a plethora of extensions to address its limitations including faster

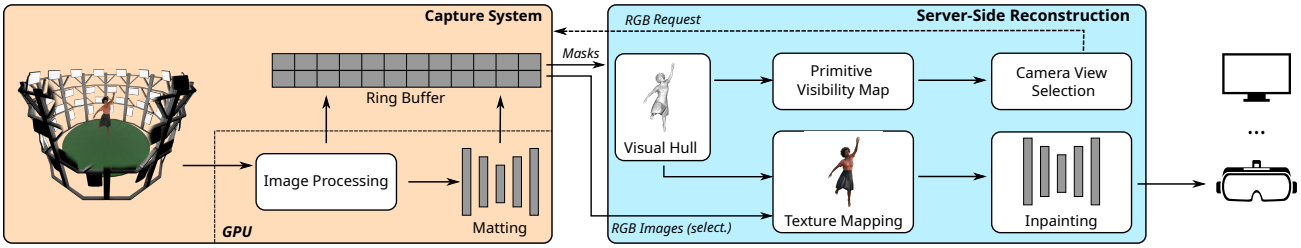


**Figure 2: Overview of the proposed end-to-end telepresence pipeline. The pipeline consists of a multi-camera capturing system, a server responsible for real-time geometry reconstruction and texture mapping, and client devices for visualization.**

training times [8, 20, 46, 47], more accurate geometry estimation via depth cues [1, 13, 56], or reducing aliasing artifacts [2–4]. Various methods also extended NeRF-based novel view synthesis to dynamic scenes, either by modeling deformations [53, 55], or by considering the temporal dimension [18, 21, 38, 67, 73]. More recently, 3D Gaussian Splatting (3DGS) marked another milestone, as it enabled real-time rendering performance by representing the scene as a set of Gaussian splats [34], and has since been further explored to also handle, e.g., dynamic scenes [28, 37, 72]. However, these approaches require offline training within minutes or hours and are thereby limited to replay use cases as opposed to the live telepresence scenario which is targeted by our method.

*Telepresence Systems.* Developing immersive telepresence system has been a challenging endeavor due to the high demands in terms of real-time visualization of an on-the-fly reconstructed scene model. 3D reconstruction methods working with cheap RGB-D sensors like KinectFusion [30, 49] enabled telepresence application at room scale [32, 42, 43]. In this context, live remote exploration of static places captured by a moving camera has been investigated [61, 75] and further extended to scenes with dynamically moving people [27], collaborative labeling scenarios [81], and robot teleoperation [62]. Other scenarios include teleconferencing system which primarily focused on high-quality reconstruction and immersive rendering of the participant's upper body even without requiring head-mounted displays [36, 69, 70, 78]. Concerning full human performance captures, more sophisticated multi-view capture setups have been proposed [5, 10, 17, 52, 54, 80]. In particular, Holoportation [52] is built upon a fast dynamic 3D reconstruction [15] pipeline to obtain and stream detailed textured 3D models. Although all the aforementioned systems pushed the limits towards immersive telepresence, they are usually specifically tailored to a certain capturing scenario and setup and rely on specially crafted pipelines without respective open-source implementations. In contrast, our telepresence framework builds upon a lightweight architecture that makes it suitable for various capturing configurations and that will be released to the community to foster future research.

## 3 Pipeline Overview

Our full end-to-end telepresence pipeline, from capture to client, is outlined in Fig. 2, aiming to provide real-time volumetric streaming of 3D scenes to VR headsets or standard displays using color images alone. Starting with the multi-view images recorded by the **capture system** using a set of pre-calibrated cameras, we perform initial processing, i.e., color correction and optical undistortion, followed by a real-time estimation of foreground masks used for geometry reconstruction. For an in-depth description of our capture system, please refer to Sec. 4. Note that our pipeline only requires the masks and images, and as such, other setups can easily be realized and integrated, which we show exemplary on a minimal setup (Sec. 4.1).

During capturing, the masks and a subset of the images are transmitted to the **reconstruction server** (Sec. 5) for geometry reconstruction and texturing. While the compressed binary masks are compact and efficient to transmit, the color images are more demanding in terms of bandwidth. To address this, we propose a camera view selection strategy based on a visual hull inferred from the masks and compute per-camera visibility maps. This guides the selection of an optimal subset of views, balancing quality and efficiency. The chosen color images are subsequently transmitted for texture mapping, followed by an inpainting algorithm to refine textures and correct artifacts.

The final renderings can be displayed on several **remote client** devices such as VR headsets or standard displays. Since expensive reconstruction and rendering are offloaded to the reconstruction server, the hardware requirements to the client are low, ensuring broad accessibility across a wide range of devices.

## 4 Capture System

As mentioned above, our capture system only has a few essential requirements to ensure compatibility with our reconstruction pipeline. We describe a full-scale setup using a capture stage for maximum quality, but also present a more lightweight alternative to suit different needs and demonstrate the versatility of our system. In essence, the system must be able to provide synchronized multi-view images and their corresponding foreground masks along with pre-computed calibration data that is used for the reconstruction.

### 4.1 Image Acquisition Hardware

Our capture stage consists of 34 RGB cameras with a resolution of 24.5 MP affixed to a cylindrical scaffolding with a diameter of 5.25 m. In addition, 46 mounted video lights provide a uniform illumination of the subjects, shining through a stretched fabric layer that contains cut-out holes for the camera lenses and ensures an uninstrusive background for matting. All cameras are connected to an array of capture servers via optical cables where each machine handles up to 4 cameras and features a NVIDIA RTX 4070 Ti Super for local processing, 128 GB RAM for buffering, and 8 SSDs with a total storage capacity of 16 TB.

*Minimal Capturing Setup.* Our system requires only standard RGB cameras, thereby enabling a minimal and cost-efficient capturing configuration. Depending on the desired output quality, such a setup can be realized with lower-cost cameras without compromising the core functionality of our method. To demonstrate the capabilities of our approach in resource-constrained environments, we conduct additional experiments using a minimal capture setup of 4 cameras arranged in a half circle in front of the recorded subject.

## 4.2 Local Processing and Buffering

The raw images from the cameras are transferred into VRAM and processed on the GPU. We apply demosaicing, white balance, color correction, undistortion, and gamma correction within a custom CUDA kernel and compress the resulting image via JPEG encoding using nvJPEG [51] for network transfer. The uncompressed images are also passed to a background matting network to generate foreground masks (see Sec. 4.3). Both the mask and compressed image are stored in a ring buffer in RAM to conserve VRAM. Using the frame index, which is an increasing counter, the images and masks can be arbitrarily retrieved from the buffer for network transfer.

## 4.3 Foreground Extraction

To extract accurate foregrounds from the captured images, we rely on the real-time matting pipeline introduced by Dröge et al. [16]. Because our setup features a fixed background during capture, we can reliably incorporate this static background as a prior into the matting framework to improve its accuracy. Specifically, we record a set of images of the empty capture stage before recording and then proceed with the actual capturing process with the present subjects. Building on this, we finetuned an adopted version of the matting model proposed by Lin et al. [39] on a small dataset of images 40 acquired in our setup to extract highly detailed foregrounds in real time. Finally, we binarize each of the foreground-background masks for further processing.

## 4.4 Communication Interface

Efficiently transferring mask data between the capture and reconstruction servers requires special attention as the resulting high volume of data demands significant bandwidth for transmission and leads to overhead when uploading the data from the network thread to the GPU. To address these challenges, we do not represent the binarized masks as ordinary images but rather compress them into a run-length encoded format. Since we only require binary information indicating whether a pixel belongs to foreground or background, we encode the masks by storing counts of consecutive pixels with identical values in a one-dimensional array and subsequently computing the cumulative sum over these run-length counts. In this representation, each bin $i$ denotes the first linear pixel index of a new segment, where the segment type (foreground or background) is determined by the parity of $i$. During decoding, the pixel value at any given index can be efficiently retrieved via binary search over these bins and checking the bin's parity. Additionally, we store a single flag indicating the segment type of the first pixel to correctly interpret the parity information.

# 5 Server-Side Reconstruction

In this section, we detail the server-side reconstruction and rendering processes of our pipeline. First, we describe our scene reconstruction method in Sec. 5.1 which consists of 1) real-time volumetric geometric reconstruction; 2) a sparse texture selection algorithm that works without explicit color data; 3) a fast projective texture mapping approach; and 4) inpainting-based post-processing to ensure temporally consistent result. This way, we can reduce data streaming to a small subset of the captured data and, thereby, significantly save bandwidth and runtime. Finally, we discuss the server-client communication in Sec. 5.2 and the final visualization at the remote clients in Sec. 5.3.

## 5.1 Scene Reconstruction

*5.1.1 Volumetric Geometry Reconstruction.* Our geometry reconstruction process relies solely on a set of foreground segmentation masks, which can be used to construct a visual hull for the geometry representation. These masks are processed and compressed on the capture servers, reducing bandwidth requirements, and are directly streamed and decoded into GPU memory on the reconstruction server with minimal overhead.

To construct the *visual hull* from the given masks, we build a sparse octree that contains the set of candidate voxels forming a small band around the surface geometry [57]. Afterwards, we extract the isosurface via Marching Cubes [41] and apply post-processing by removing duplicate vertices to obtain a clean watertight mesh of the scene geometry. Note that all these operations are executed in real time and independently for each frame which allows our method to avoid the inherent computational cost associated with multi-frame tracking and correspondence estimation. Furthermore, computing an explicit representation offers advantages beyond reconstruction, as it enables the extraction of auxiliary information to support various downstream processing tasks, such as image selection and occlusion handling.

One key application of this representation is the generation of *primitive visibility maps* for each camera view, which determine the visible portions of the reconstructed mesh from that view. This is achieved by rendering the mesh using the known camera parameters of the capturing system and storing unique triangle IDs. Here, the rasterization pipeline automatically handles occlusion of primitives as non-visible triangles are discarded during z-testing. To reduce computational overhead, the visibility maps are generated in a lower resolution where we specify the image width as 400 pixels and compute the height based on the known camera intrinsics to preserve the original aspect ratio.

*5.1.2 Camera View Selection.* Based on the primitive visibility information and the visual hull-based scene geometry, we developed a novel camera view selection method that efficiently determines the most relevant input views for texturing, without requiring additional access to color information. Choosing a subset is crucial to meet the bandwidth and runtime constraints of the telepresence scenario, so we leverage the per-input-view associated primitive maps and render a corresponding map for the novel target view $V$.

To compare visibility across views, we first convert the primitive visibility maps into binary vectors. For each view $i \in \mathcal{I}$ with $\mathcal{I} = \{1, \ldots, n_C\}$, we define a binary vector $\boldsymbol{b}_i \in \{0,1\}^{n_T}$ where $n_T$

**ALGORITHM 1:** Greedy Camera Selection

**Input:** Candidate camera set $\mathcal{I}$; target vector $\boldsymbol{b}_V \in \{0,1\}^{n_T}$; visibility vectors $\{\boldsymbol{b}_i\}_{i\in\mathcal{I}}$; size of subset $k$
**Output:** Selected camera indices $\mathcal{S}$
$\mathcal{S} \leftarrow \emptyset$
**while** $\boldsymbol{b}_V \neq \mathbf{0}$ ***and*** $|\mathcal{S}| < k$ **do**
  $i^* \leftarrow \arg\max_{i\in\mathcal{I}} \langle \boldsymbol{b}_i, \boldsymbol{b}_V \rangle$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{i^*\}$
  $\mathcal{I} \leftarrow \mathcal{I} \setminus \{i^*\}$
  $\boldsymbol{b}_V \leftarrow \boldsymbol{b}_V \odot (\mathbf{1} - \boldsymbol{b}_{i^*})$
**end**
**return** $\mathcal{S}$

is the total number of primitives. Each entry $b_{ij}$ is given by

$$b_{ij} = \begin{cases} 1, & \text{if triangle } j \text{ is visible form view } i, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

We compute this binary representation for each camera as well as for the virtual viewpoint $\boldsymbol{b}_V$.

Our objective is to select a subset $\mathcal{S} \subset \mathcal{I}$ of $k$ camera views (i.e. $|\mathcal{S}| = k$) that best captures the primitives observed in the reference view $\boldsymbol{b}_V$. Thus, we maximize the similarity between the aggregated visibility and the reference view, that is

$$\mathcal{S} = \underset{\mathcal{J},\,|\mathcal{J}|=k}{\arg\max} \left\langle \bigvee_{i\in\mathcal{J}} \boldsymbol{b}_i, \boldsymbol{b}_V \right\rangle \tag{2}$$

where $\bigvee_{i\in\mathcal{J}} \boldsymbol{b}_i$ denotes the element-wise disjunction (logical OR) of the binary vectors $\boldsymbol{b}_i$ and $\langle\cdot,\cdot\rangle$ the dot product.

We address this optimization problem using a greedy strategy, as shown in Alg. 1. In the first step, we select the camera view $i^*$ that observes the greatest number of triangles present in the target view. Once a camera is selected, we add it to the subset $\mathcal{S}$, remove it from the candidate set $\mathcal{I}$, and update the reference vector $\boldsymbol{b}_V$ by setting the entries to zero which correspond to triangles already covered by the selected view. This way, cameras are prioritized that are likely to capture previously unseen regions, thereby improving overall target view coverage. The repeat this process until either all triangles are covered or $k$ cameras have been successfully selected. It is worth to mention that, although some cameras might observe already covered triangles, this additional information is beneficial to ensure that critical primitives are well represented.

*5.1.3 Projective Texture Mapping.* After the subset $\mathcal{S}$ has been determined, the actual projective texturing is performed in a final render pass. For this, we request the RGB images of the selected views, decompress them, and copy them into OpenGL textures. Furthermore, we render depth maps $\boldsymbol{D}_i$ from the selected views $i \in \mathcal{S}$ to perform occlusion testing. To reduce ghosting artifacts that appear if the occlusion test near the boundary of objects fails, we use a similar technique to Holoportation [52]. However, instead of looking for discontinuities of the depth values in image space, we render out the depth maps with slightly inflated meshes such that boundary pixels are more likely to fail the occlusion test. Given a fragment's world position $\boldsymbol{v}$, we compute its projected UV-coordinates into the $i$-th camera view,

$$\mathbf{uv}_i(\boldsymbol{v}) = \frac{1}{2}\Big(\boldsymbol{\pi}(\boldsymbol{P}_i \cdot \boldsymbol{V}_i \cdot \hat{\boldsymbol{v}})_{xy}\Big) + \frac{1}{2} \in [0,1]^2, \tag{3}$$

where the $\hat{\boldsymbol{v}}$ represents the position $\boldsymbol{v}$ in homogeneous coordinates, $\boldsymbol{\pi}$ denotes perspective division, and $\boldsymbol{P}_i \cdot \boldsymbol{V}_i$ is the combined view-projection matrix for camera $i$. We additionally retain the $z$-coordinate of this projection, so we can use it for occlusion testing by comparing it to the corresponding pixel value in the previously generated depth map $\boldsymbol{D}_i$. All colors passing this depth test are collected and then blended together via additive blending to infer the final color of the fragment

$$\boldsymbol{c} = \frac{\sum_{i\in\mathcal{S}} w_i \cdot \boldsymbol{c}_i}{\sum_{i\in\mathcal{S}} w_i}, \qquad w_i = w_i^{\text{normal}} \cdot w_i^{\text{view}}. \tag{4}$$

Here, $w_i$ consists of a normal-based weight [52] and a view-based weight [17]. The first weight $w_i^{\text{normal}}$ uses normal information $\boldsymbol{n}$ from the visual hull to minimize distortions by down-weighting triangles that are significantly tilted with respect to the camera's per fragment viewing direction $\boldsymbol{d}_i$:

$$w_i^{\text{normal}} = \max\big(0, \langle \boldsymbol{n}, \boldsymbol{d}_i \rangle^{\alpha}\big). \tag{5}$$

The second weight $w_i^{\text{view}}$ is calculated based on the alignment between the virtual per fragment viewing direction $\boldsymbol{d}$ and $\boldsymbol{d}_i$:

$$w_i^{\text{view}} = \max\Big(0, \langle \boldsymbol{d}, \boldsymbol{d}_i \rangle^{\beta}\Big). \tag{6}$$

Thus, $w_i^{\text{view}}$ prioritizes cameras whose viewing angles closely match the virtual view, assigning higher weights to pixels observed under similar viewing directions. The hyperparameters $\alpha$ and $\beta$ control the influence of certain directions and we choose $\alpha = 2$ and $\beta = 16$ throughout all experiments.

*5.1.4 Video Inpaiting.* While our camera view selection and projective texturing approach enables efficient novel view synthesis, small parts of the mesh, e.g. close to occlusion boundaries discontinuities, may not be textured from the $k$ cameras views or exhibit artifacts. To address these issues, we apply a real-time inpaining algorithm as a final post-processing step for texture reconstruction. For this purpose, we additionally export a binary mask with regions that could not be reconstructed during the previously outlined texturing pass. Afterwards, we apply a video inpaining framework [66] in these regions which, by taking information from the previous inpainted frames into account, ensures temporal consistency.

*5.1.5 Thread and GPU Handling.* In order to achieve good performance, it is vital to reduce the amount of memory transfer between CPU and GPU. As some of the aforementioned operations are executed in CUDA kernels and most of the rendering-related tasks are performed in OpenGL shaders, we use the interoperability between the two to facilitate efficient synchronization directly in GPU memory. Furthermore, we use a multi-GPU architecture where one GPU is used to handle the geometry reconstruction (Sec. 5.1.1) and one is used for rendering and inpaining (Sec. 5.1.2 to 5.1.4). Both of these GPUs are accessed by a different set of threads to reduce synchronization overhead which is depicted in Fig. 3.

First, the *reconstruction thread* handles the *reconstruction GPU*. It receives the compressed foreground segmentation masks from the capture server and uploads them onto the GPU where the data is

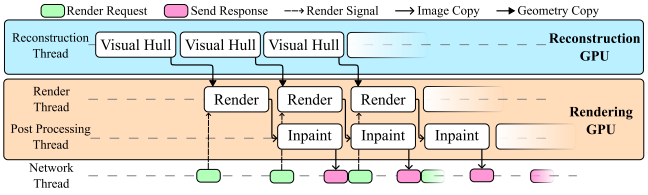


**Figure 3: Overview of our dual-GPU reconstruction server with multi-threaded processing.**

decoded into a pre-allocated image buffer and used for the geometry reconstruction. Second, the *rendering thread* is responsible for the texture reconstruction, which is performed on the designated *rendering GPU*. This thread performs our proposed camera view selection algorithm and requests the corresponding RGB images from the capture servers. As mentioned in Sec. 4.2, the capture servers compress the selected images into JPEG byte streams, which are then received and directly decoded into the device memory of the reconstruction server via hardware-accelerated decompression [51]. The decoded data is then copied into an OpenGL texture object, which eliminates the overhead of traditional CPU-to-GPU transfers, and the final texture mapping is performed in a fragment shader.

In order to scale the system to multiple remote users, each client spawns its own rendering thread when connecting to the server. Each thread spawned this way is also associated with its own client-exclusive memory to avoid race conditions and expensive synchronization. On top of that, as the rendering tasks are independent for each client, each thread creates and handles its own OpenGL context. Synchronization between the two GPUs is only necessary when copying the geometry representation into the clients' local memory, which is directly mapped to an OpenGL vertex buffer for texturing. Additionally, each client also spawns a *post-processing thread* that is responsible for running video inpainting on the rendered results to further improve visualization quality.

## 5.2 Server-Client Communication

Lastly, the server has to send the rendered result back to the individual clients which is performed in a dedicated *network thread* as shown in Fig. 3. We again use hardware accelerated JPEG compression to encode the final rendering on the GPU and send the resulting byte stream to the client. In addition, we also encode and send the depth buffer such that other tasks like reprojection for VR headsets can be performed on client side. As lossy JPEG compression would create artifacts in the depth map that are noticeable in the final rendering, the depth data is instead encoded via quantization by remapping the values to 16-bit integers and applying lossless compression via zstd [74].

## 5.3 Remote Clients

The last component of our telepresence system are the remote clients. Since our goal is to perform most of the computationally expensive tasks on the reconstruction server, the remote clients can run on comparatively cheap hardware as for example a smartphone or lower-end graphics cards. In the simplest scenario, they send the current viewing position and direction to the server and receive an image of the reconstructed scene from this location, which then only has to be decoded and displayed. However, this can lead to an unpleasant user experience (for example in VR) if the latency is too high or the reconstruction frame rate is too low. To mitigate this problem, the reconstruction server also sends depth data of the scene to the client as described in Sec. 5.2. This additional information can then be used to generate a point cloud that can be re-rendered on the client side. Using this technique, the actual rendering framerate is decoupled from the reconstruction framerate as necessary intermediate frames can be generated on the client side which leads to a significantly smoother experience. Note that this cheap approximation also directly allows to render a second view on the client side for VR headsets without additional overhead.

# 6 Benchmark

We introduce a new benchmark dataset that is particularly designed for multi-view reconstruction and novel view synthesis tasks. The dataset was recorded with the capturing system described in Sec. 4 and comprises video data across 32 unique scenes. Unlike prior datasets focused on single subjects [68, 76], ours contains scenes with multiple actors and a wide range of objects, from rigid to soft, small to large, and diffuse to reflective. It also features animals and topologically challenging interactions (e.g., objects moved out of backpacks, cloth folding), which are designed to test the limits of dynamic reconstruction systems. The video sequences cover a variety of actions lasting from 2 to 20 seconds and were recorded at a resolution of $2664 \times 2304$ pixels at 25 FPS. Thus, they offer significantly more details than the TotalCapture [68] or the CMU Panoptic datasets [33] and mostly match the quality of the DNA-Rendering [11] dataset but at higher framerates. Each recording includes high-resolution color images, per-frame foreground masks for both humans and objects, separately captured background images – a feature not available in the other datasets – as well as full camera calibration data. To facilitate the evaluation of generalization to out-of-distribution viewpoints, we added an additional dedicated camera positioned outside the regular camera arrangement to act as a challenging *test viewpoint*.

# 7 Evaluation

In order to evaluate our telepresence pipeline, we analyze runtime performance, data transmission bandwidth, as well as reconstruction quality. An ablation study further demonstrates the impact of our system design in terms of both quality and runtime. For evaluation, the reconstruction server runs on two NVIDIA A100 with an AMD Epyc 7713 CPU whereas for the remote client, we use a Laptop with a NVIDIA RTX 2070 Max-Q. For VR applications, we send the final rendered image to a Meta Quest Pro.

## 7.1 Reconstruction Quality

We compare the reconstruction quality of our method to the out-of-the-box implementation of an offline NeRF (Nerfacto) and 3DGS (Splatfacto) pipeline via Nerfstudio [65]. For the splatting approach, we initialize the scenes with point clouds obtained from the visual hull since random initializations led to poor results. Fig. 4 presents a qualitative comparison between Splatfacto and our method. Our approach achieves a reconstruction quality comparable to the offline

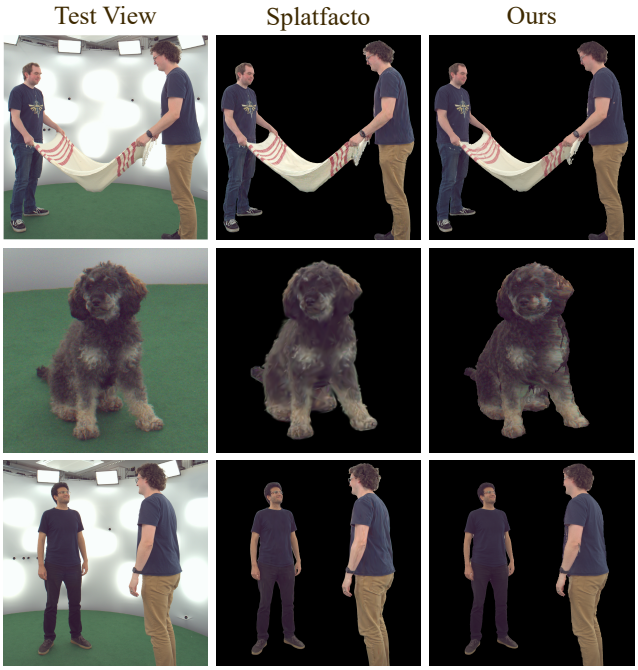


**Figure 4: Qualitative comparison between Splatfacto, and our method across different scenes from an unseen test view.**

method while faithfully capturing multiple subjects and objects, including intricate fabric details. Notably, the second row of the figure reveals distinctly clearer textures, highlighting our method's ability to capture fine details. For additional qualitative results, we refer to the supplemental video.

For a quantitative comparison, we evaluated the methods on a subset of the DNA-Rendering [11] dataset in Table 1. They provide data from a capture stage similar to ours with 48 cameras at $2048 \times 2448$ pixels and we used the masks provided with the data. The table shows a clear quality-performance trade-off between the methods. For a fair comparison, we evaluated our method only on one GPU like the other methods. Note that in our complete pipeline, the total frame-time is lower due to parallelization and distributed computing as described previously. While Splatfacto achieves the best quality in terms of PSNR and SSIM, this comes at a high computational cost. This is because Splatfacto uses all available images which provides more information in occluded regions (Fig. 5). Our method also struggles with highly reflective objects because our blending approach creates artifacts in the highlights.

**Table 1: Quality comparison and GPU time between offline reconstruction and our method on DNA-Rendering [11].**

| Method | PSNR ↑ | SSIM ↑ | LPIPS ↓ | Time ↓ |
|---|---|---|---|---|
| Splatfacto-big | 34.5(29) | 0.976(16) | 0.028(8) | 11.25(43) min |
| Nerfacto-big | 29.5(35) | 0.920(71) | 0.061(28) | 1.48(14) h |
| Ours w/o inp. | 29.8(27) | 0.930(41) | 0.031(9) | 35.8(25) ms |
| Ours | 29.9(27) | 0.932(40) | 0.030(9) | 41.5(82) ms |


**Figure 5: Quality-performance trade-off of our method compared to Splatfacto on challenging scenes with occlusions.**

**Figure 6: Reflective materials can create artifacts if seen from slightly different views.**

In contrast, our method offers a competitive improvement in quality over Nerfacto but only requires a tiny fraction of its computation time. The disparity in quality between Nerfacto and Splatfaco can partly attributed to the relatively low amount of total camera viewpoints compared to NeRF-related datasets which manifests in several floaters in Nerfacto's model. These results clearly demonstrate that our method substantially outperforms both approaches in terms of computational speed, while still providing sufficient reconstruction quality, leading to an effective trade-off between image quality and processing time.

In addition, we evaluated our framework on our recorded benchmark dataset introduced in Sec. 6. We divided our experiments into two distinct sub-experiments. The upper part of Table 2 presents a comparison to Splatfactor on a randomly sampled subset of 32 frames from the dataset – a necessary approach given that 3DGS would require prohibitively long training times on the full dataset. To ensure best coverage of the dataset, we sampled one random data point from each video. The lower part of the table presents our method's performance on the complete dataset. In both cases, our method achieves a similar quality as offline reconstruction, while reaching real-time reconstruction and rendering speed.

**Table 2: Quality comparison and GPU time between offline Splatfacto and our method on our provided benchmark. ‡ indicates evaluation on a subset and † on the full dataset.**

| Method | PSNR ↑ | SSIM ↑ | LPIPS ↓ | Time ↓ |
|---|---|---|---|---|
| Splatfacto$^{‡}$ | 30.7(49) | 0.920(237) | 0.058(62) | 12.7(4) min |
| Ours$^{‡}$ | 29.4(33) | 0.960(25) | 0.030(16) | 52.3(36) ms |
| Ours$^{†}$ | 29.1(33) | 0.953(25) | 0.033(18) | 51.3(43) ms |

**Figure 8: Two viewing angles of the reconstruction from our minimal capture setup.**

**Figure 9: Direction-based (left) vs. our (right) view selection.**

To demonstrate the adaptability of our framework, we also evaluated its performance using a minimal capturing setup consisting of 4 cameras, as described in Sec. 4.1. Fig. 8 illustrates the qualitative results for two novel views, showing the ability to capture geometry and texture details even under constrained data conditions.

## 7.2 Runtime and Bandwidth

We evaluated the bandwidth for a live telepresence transmission over an interval of 60 s. In our experiments, the bandwidth generally fluctuates between 10 Mbit/s and 40 Mbit/s, depending on the visible scene content and the relative position of the remote view. In general, the average bandwidth in this scenario is 21.21 Mbit/s, which shows that our approach can be used in bandwidth limited scenarios. For the connection between capture and reconstruction server, the average bandwidth is about 1.4 Gbit/s. It is important to note, however, that this only impacts the capturing side and does not affect the remote client's capabilities.

In Table 3, we report the runtime across various processing stages in our system. By adding all of these together, we get an overall average latency of 141.08 ms between a captured frame and the final reconstruction observed by the client. Thus, the system performs within the range necessary for seamless human interaction, aligning with the typical response times observed in comparable systems. Note that due to the parallel processing of frames, the total frame rate is determined by the maximum of the individual times, i.e. 23 FPS. This shows that our system is capable of low-latency, real-time streaming for immersive telepresence applications.

We also evaluated the system scalability in a multi-user scenario in Fig. 11. While in its current form, the runtime scales linearly with the number of clients, we still achieve interactive frame rates when streaming up to 4 users. It is also important to note that this only affects the smoothness of the reconstructed motion. The client still has a lag-free viewing experience because it can rerender the already transmitted geometry from previous frames.

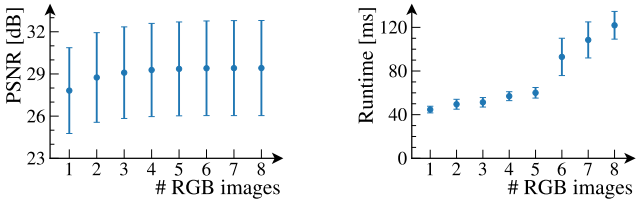


**Figure 10: Effect of image count on system performance.**

| Stage | Time (ms) |
|---|---|
| Capturing | 34.14(437) |
| Mask Proc. + Reconst. | 25.61(631) |
| RGB Proc. + Rend. | 45.40(936) |
| Post-Processing | 16.14(213) |
| Streaming to Client | 19.09(676) |
| Display | 0.70(100) |
| Latency | 141.08(1406) |

**Table 3: Breakdown of system runtime in different processing stages.**

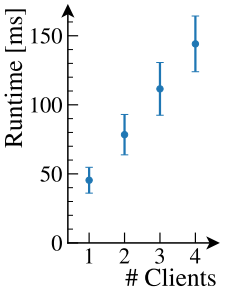


**Figure 11: Number of clients vs. runtime.**

## 7.3 Ablation and Limitations

*Ablation.* We evaluated how different parts of our system affect the overall reconstruction quality and runtime. Fig. 9 shows a comparison between a naive direction-based camera selection and our proposed method. Since the naive method does not take geometry into account, regions that are not seen by any camera can not be textured which leads to holes. In contrast, our method does handle these cases and thus finds a better coverage of the target geometry.

Because the pipeline is easily extendable, it is also possible to consider more images for the reconstruction. As shown in Fig. 10, the quality only marginally improves since the additional information is mainly used to fill in the progressively shrinking gaps, which highlights the efficiency of our greedy selection strategy. However, this comes with significantly higher runtimes as using more images increases memory pressure on both the network and rendering pipeline. We also evaluated the impact of the inpainting. However, because the affected regions are small, the metrics (Table 1) are not well suited to capture the impact, yet the visual differences are clearly noticeable (see supplemental material).

*Limitations.* Our method relies on the visual hull, which inherently struggles to accurately reconstruct concavities. This can lead to ghosting artifacts as the texture may be projected onto wrong or missing parts of the geometry. Additionally, we assume diffuse materials, which limits the usability of our system for specular or transparent objects. Currently, only geometry data is shared between users, so additionally caching and sharing RGB data could further improve scalability and reduce the latency of our system.

## 8 Conclusion

We presented RIFTCast, a multi-view telepresence framework targeting a wide variety of setups and dynamic scenes without relying on depth data or template models. Our method leverages a lightweight visual hull-based representation coupled with a bandwidth-efficient architecture. We openly share our code along with a novel benchmark dataset to foster reproducibility and to push the boundaries of immersive telepresence research. Furthermore, the real-time performance and lightweight representation achieved by RIFTCast suggest its potential utility beyond live telepresence, for instance, as an efficient preview mechanism for navigating and inspecting large pre-recorded volumetric video datasets before committing to more computationally intensive, high-fidelity rendering processes.

## Acknowledgments

This work has been funded by the Ministry of Culture and Science North Rhine-Westphalia under grant number PB22-063A (InVirtuo 4.0: Experimental Research in Virtual Environments), and by the state of North Rhine Westphalia as part of the Excellency Start-up Center.NRW (U-BO-GROW) under grant number 03ESCNW18B.

## References

[1] Benjamin Attal, Eliot Laidlaw, Aaron Gokaslan, Changil Kim, Christian Richardt, James Tompkin, and Matthew O'Toole. 2021. Törf: Time-of-flight radiance fields for dynamic scene view synthesis. *Advances in Neural Information Processing Systems (NeurIPS)* 34 (2021).

[2] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. 2021. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *IEEE/CVF International Conference on Computer Vision (ICCV)*.

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. 2022. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2023. Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields. In *IEEE/CVF International Conference on Computer Vision (ICCV)*.

[5] Pablo Carballeira, Carlos Carmona, César Díaz, Daniel Berjón, Daniel Corregidor, Julián Cabrera, Francisco Morán, Carmen Doblado, Sergio Arnaldo, María del Mar Martín, et al. 2021. FVV live: A real-time free-viewpoint video system with consumer electronics hardware. *IEEE Transactions on Multimedia (TMM)* 24 (2021).

[6] Gaurav Chaurasia, Sylvain Duchêne, Olga Sorkine-Hornung, and George Drettakis. 2013. Depth Synthesis and Local Warps for Plausible Image-based Navigation. *ACM Transactions on Graphics (TOG)* 32, 3 (2013).

[7] Gaurav Chaurasia, Olga Sorkine, and George Drettakis. 2011. Silhouette-Aware Warping for Image-Based Rendering. *Computer Graphics Forum (CGF)* 30, 4 (2011).

[8] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*.

[9] Jun Chen, Ryosuke Watanabe, Keisuke Nonaka, Tomoaki Konno, Hiroshi Sankoh, and Sei Naito. 2019. Fast free-viewpoint video synthesis algorithm for sports scenes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[10] Ruizhi Cheng, Nan Wu, Vu Le, Eugene Chai, Matteo Varvello, and Bo Han. 2024. Magicstream: Bandwidth-conserving immersive telepresence via semantic communication. In *ACM Conference on Embedded Networked Sensor Systems*.

[11] Wei Cheng, Ruixiang Chen, Wanqi Yin, Siming Fan, Keyu Chen, Honglin He, Huiwen Luo, Zhongang Cai, Jingbo Wang, Yang Gao, Zhengming Yu, Zhengyu Lin, Daxuan Ren, Lei Yang, Ziwei Liu, Chen Change Loy, Chen Qian, Wayne Wu, Dahua Lin, Bo Dai, and Kwan-Yee Lin. 2023. DNA-Rendering: A Diverse Neural Actor Repository for High-Fidelity Human-centric Rendering. *arXiv preprint* arXiv:2307.10173 (2023).

[12] Yanran Dai, Jing Li, Yuqi Jiang, Haidong Qin, Bang Liang, Shikuan Hong, Haozhe Pan, and Tao Yang. 2024. Real-time distance field acceleration based free-viewpoint video synthesis for large sports fields. *Computational Visual Media* 10, 2 (2024).

[13] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. 2022. Depth-supervised nerf: Fewer views and faster training for free. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[14] Mingsong Dou, Philip Davidson, Sean Ryan Fanello, Sameh Khamis, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, and Shahram Izadi. 2017. Motion2fusion: Real-time volumetric performance capture. *ACM Transactions on Graphics (TOG)* 36, 6 (2017).

[15] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, et al. 2016. Fusion4d: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics (TOG)* 35, 4 (2016).

[16] Hannah Dröge, Janelle Pfeifer, Saskia Rabich, Markus Plack, Reinhard Klein, and Matthias B Hullin. 2025. Capture Stage Environments: A Guide to Better Matting. *arXiv preprint arXiv:2507.07623* (2025).

[17] Ruofei Du, Ming Chuang, Wayne Chang, Hugues Hoppe, and Amitabh Varshney. 2018. Montage4D: interactive seamless fusion of multiview video textures.. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*.

[18] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B Tenenbaum, and Jiajun Wu. 2021. Neural radiance flow for 4d view synthesis and video processing. In *IEEE/CVF International Conference on Computer Vision (ICCV)*.

[19] Martin Eisemann, Bert De Decker, Marcus Magnor, Philippe Bekaert, Edilson De Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. 2008. Floating textures. *Computer Graphics Forum (CGF)* 27, 2 (2008).

[20] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2022. Plenoxels: Radiance Fields Without Neural Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[21] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. 2021. Dynamic view synthesis from dynamic monocular video. In *IEEE/CVF International Conference on Computer Vision (ICCV)*.

[22] Wei Gao and Russ Tedrake. 2019. Surfelwarp: Efficient non-volumetric single view dynamic reconstruction. In *Robotics: Science and Systems*.

[23] Michael Goesele, Jens Ackermann, Simon Fuhrmann, Carsten Haubold, Ronny Klowsky, Drew Steedly, and Richard Szeliski. 2010. Ambient point clouds for view interpolation. *ACM Transactions on Graphics (TOG)* (2010).

[24] Suhas Gopal, Rishabh Dabral, Vladislav Golyanik, and Christian Theobalt. 2025. Betsu-Betsu: Multi-View Separable 3D Reconstruction of Two Interacting Objects. In *International Conference on 3D Vision (3DV)*.

[25] Kaiwen Guo, Jonathan Taylor, Sean Fanello, Andrea Tagliasacchi, Mingsong Dou, Philip Davidson, Adarsh Kowdle, and Shahram Izadi. 2018. Twinfusion: High framerate non-rigid fusion through fast correspondence tracking. In *International Conference on 3D Vision (3DV)*.

[26] Marc Habermann, Weipeng Xu, Michael Zollhoefer, Gerard Pons-Moll, and Christian Theobalt. 2019. Livecap: Real-time human performance capture from monocular video. *ACM Transactions on Graphics (TOG)* 38, 2 (2019).

[27] Leif Van Holland, Patrick Stotko, Stefan Krumpen, Reinhard Klein, and Michael Weinmann. 2023. Efficient 3D Reconstruction, Streaming and Visualization of Static and Dynamic Scene Parts for Multi-client Live-telepresence in Large-scale Environments. In *IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*.

[28] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. 2024. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[29] Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. 2016. Volumedeform: Real-time volumetric non-rigid reconstruction. In *European Conference on Computer Vision (ECCV)*.

[30] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. 2011. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *ACM Symposium on User Interface Software and Technology (UIST)*.

[31] Yuheng Jiang, Suyi Jiang, Guoxing Sun, Zhuo Su, Kaiwen Guo, Minye Wu, Jingyi Yu, and Lan Xu. 2022. Neuralhofusion: Neural volumetric rendering under human-object interactions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[32] Brett Jones, Rajinder Sodhi, Michael Murdock, Ravish Mehra, Hrvoje Benko, Andrew Wilson, Eyal Ofek, Blair MacIntyre, Nikunj Raghuvanshi, and Lior Shapira. 2014. RoomAlive: Magical Experiences Enabled by Scalable, Adaptive Projector-Camera Units. In *ACM Symposium on User Interface Software and Technology (UIST)*.

[33] Hanbyul Joo, Tomas Simon, Xulong Li, Hao Liu, Lei Tan, Lin Gui, Sean Banerjee, Timothy Scott Godisart, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. 2017. Panoptic Studio: A Massively Multiview System for Social Interaction Capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2017).

[34] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)* 42, 4 (2023).

[35] Carmel Kozlov, Miroslava Slavcheva, and Slobodan Ilic. 2018. Patch-based non-rigid 3d reconstruction from a single depth stream. In *International Conference on 3D Vision (3DV)*.

[36] Jason Lawrence, Danb Goldman, Supreeth Achar, Gregory Major Blascovich, Joseph G Desloge, Tommy Fortes, Eric M Gomez, Sascha Häberling, Hugues Hoppe, Andy Huibers, et al. 2021. Project starline: a high-fidelity telepresence system. *ACM Transactions on Graphics (TOG)* 40, 6 (2021).

[37] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. 2024. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[38] Haotong Lin, Sida Peng, Zhen Xu, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. 2022. Efficient neural radiance fields for interactive free-viewpoint video. In *SIGGRAPH Asia 2022 Conference Papers*.

[39] Shanchuan Lin, Andrey Ryabtsev, Soumyadip Sengupta, Brian L Curless, Steven M Seitz, and Ira Kemelmacher-Shlizerman. 2021. Real-time high-resolution background matting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[40] Wenbin Lin, Chengwei Zheng, Jun-Hai Yong, and Feng Xu. 2022. Occlusionfusion: Occlusion-aware motion estimation for real-time dynamic 3d reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[41] W. E. Lorensen and H. E. Cline. 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, Maureen C. Stone (Ed.). ACM Press, 163–169. doi:10.1145/37401.37422

[42] A. Maimone, J. Bidwell, K. Peng, and H. Fuchs. 2012. Enhanced personal autostereoscopic telepresence system using commodity depth cameras. *Computers & Graphics* 36, 7 (2012).

[43] A. Maimone and H. Fuchs. 2012. Real-time volumetric 3D capture of room-sized scenes for telepresence. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*.

[44] Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, Anastasia Tkach, Peter Lincoln, et al. 2018. Lookingood: Enhancing performance capture with real-time neural re-rendering. *ACM Transactions on Graphics (TOG)* (2018).

[45] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *European Conference on Computer Vision (ECCV)*.

[46] Muhammad Husnain Mubarik, Ramakrishna Kanungo, Tobias Zirr, and Rakesh Kumar. 2023. Hardware acceleration of neural graphics. In *Annual International Symposium on Computer Architecture*.

[47] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (TOG)* 41, 4 (2022).

[48] Richard A Newcombe, Dieter Fox, and Steven M Seitz. 2015. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[49] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. 2011. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*.

[50] Keisuke Nonaka, Ryosuke Watanabe, Jun Chen, Houari Sabirin, and Sei Naito. 2018. Fast plane-based free-viewpoint synthesis for real-time live streaming. In *IEEE Visual Communications and Image Processing (VCIP)*.

[51] NVIDIA Corporation. 2025. *NVIDIA nvJPEG Library*. https://developer.nvidia.com/nvjpeg Accessed: 2025-03-28.

[52] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L Davidson, Sameh Khamis, Mingsong Dou, et al. 2016. Holoportation: Virtual 3d teleportation in real-time. In *ACM Symposium on User Interface Software and Technology (UIST)*.

[53] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. 2021. Nerfies: Deformable neural radiance fields. In *IEEE/CVF International Conference on Computer Vision (ICCV)*.

[54] Benjamin Petit, Jean-Denis Lesage, Clément Menier, Jérémie Allard, Jean-Sébastien Franco, Bruno Raffin, Edmond Boyer, and François Faure. 2010. Multicamera Real-Time 3D Modeling for Telepresence and Remote Collaboration. *International journal of Digital Multimedia Broadcasting* 2010, 1 (2010).

[55] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2021. D-nerf: Neural radiance fields for dynamic scenes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[56] Konstantinos Rematas, Andrew Liu, Pratul P Srinivasan, Jonathan T Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. 2022. Urban radiance fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[57] Hanno Scharr, Christoph Briese, Patrick Embgenbroich, Andreas Fischbach, Fabio Fiorani, and Mark Müller-Linow. 2017. Fast high resolution volume carving for 3D plant shoot reconstruction. *Frontiers in Plant Science* 8 (2017).

[58] Ashwath Shetty, Marc Habermann, Guoxing Sun, Diogo Luvizon, Vladislav Golyanik, and Christian Theobalt. 2024. Holoported Characters: Real-time Free-viewpoint Rendering of Humans from Sparse RGB Cameras. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[59] Miroslava Slavcheva, Maximilian Baust, Daniel Cremers, and Slobodan Ilic. 2017. Killingfusion: Non-rigid 3d reconstruction without correspondences. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[60] Miroslava Slavcheva, Maximilian Baust, and Slobodan Ilic. 2018. Sobolevfusion: 3d reconstruction of scenes undergoing free non-rigid motion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[61] Patrick Stotko, Stefan Krumpen, Matthias B. Hullin, Michael Weinmann, and Reinhard Klein. 2019. SLAMCast: Large-Scale, Real-Time 3D Reconstruction and Streaming for Immersive Multi-Client Live Telepresence. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 25, 5 (2019).

[62] Patrick Stotko, Stefan Krumpen, Max Schwarz, Christian Lenz, Sven Behnke, Reinhard Klein, and Michael Weinmann. 2019. A VR system for immersive teleoperation and live exploration with a mobile robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[63] Zhuo Su, Lan Xu, Dawei Zhong, Zhong Li, Fan Deng, Shuxue Quan, and Lu Fang. 2022. Robustfusion: Robust volumetric performance reconstruction under human-object interactions from monocular rgbd stream. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 45, 5 (2022).

[64] Xin Suo, Yuheng Jiang, Pei Lin, Yingliang Zhang, Minye Wu, Kaiwen Guo, and Lan Xu. 2021. Neuralhumanfvv: Real-time neural volumetric human performance rendering using rgb cameras. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[65] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. 2023. Nerfstudio: A Modular Framework for Neural Radiance Field Development. In *ACM SIGGRAPH 2023 Conference Proceedings (SIGGRAPH '23)*.

[66] Guillaume Thiry, Hao Tang, Radu Timofte, and Luc Van Gool. 2024. Towards Online Real-Time Memory-based Video Inpainting Transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[67] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. 2021. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *IEEE/CVF International Conference on Computer Vision (ICCV)*.

[68] Matt Trumble, Andrew Gilbert, Charles Malleson, Adrian Hilton, and John Collomosse. 2017. Total Capture: 3D Human Pose Estimation Fusing Video and Inertial Sensors. In *2017 British Machine Vision Conference (BMVC)*.

[69] Hanzhang Tu, Ruizhi Shao, Xue Dong, Shunyuan Zheng, Hao Zhang, Lili Chen, Meili Wang, Wenyu Li, Siyan Ma, Shengping Zhang, Boyao Zhou, and Yebin Liu. 2024. Tele-Aloha: A Telepresence System with Low-budget and High-authenticity Using Sparse RGB Cameras. In *ACM SIGGRAPH 2024 Conference Papers*.

[70] Shengze Wang, Ziheng Wang, Ryan Schmelzle, Liujie Zheng, YoungJoong Kwon, Roni Sengupta, and Henry Fuchs. 2024. Learning view synthesis for desktop telepresence with few RGBD cameras. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2024).

[71] Taku Watanabe and Toshiyuki Tanaka. 2010. Free viewpoint video synthesis on human action using shape from silhouette method. In *SICE Annual Conference 2010*.

[72] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[73] Zhen Xu, Sida Peng, Haotong Lin, Guangzhao He, Jiaming Sun, Yujun Shen, Hujun Bao, and Xiaowei Zhou. 2024. 4k4d: Real-time 4d view synthesis at 4k resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[74] Chip Turner Yann Collet. 2016. Smaller and faster data compression with Zstandard. https://engineering.fb.com/2016/08/31/core-infra/smaller-and-faster-data-compression-with-zstandard/ Accessed: 2025-04-04.

[75] Jacob Young, Tobias Langlotz, Steven Mills, and Holger Regenbrecht. 2020. Mobileportation: Nomadic telepresence for mobile devices. *ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 2 (2020).

[76] Tao Yu, Zerong Zheng, Kaiwen Guo, Pengpeng Liu, Qionghai Dai, and Yebin Liu. 2021. Function4D: Real-time Human Volumetric Capture from Very Sparse Consumer RGBD Sensors. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[77] Uemoto Yusuke, Keita Takahashi, and Toshiaki Fujii. 2018. Free viewpoint video generation system using visual hull. In *International Workshop on Advanced Image Technology (IWAIT)*.

[78] Yizhong Zhang, Jiaolong Yang, Zhen Liu, Ruicheng Wang, Guojun Chen, Xin Tong, and Baining Guo. 2022. Virtualcube: An immersive 3d video communication system. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 28, 5 (2022).

[79] Zerong Zheng, Tao Yu, Hao Li, Kaiwen Guo, Qionghai Dai, Lu Fang, and Yebin Liu. 2018. Hybridfusion: Real-time performance capture using a single depth sensor and sparse imus. In *European Conference on Computer Vision (ECCV)*.

[80] Yifeng Zhou, Shuheng Wang, Wenfa Li, Chao Zhang, Li Rao, Pu Cheng, Yi Xu, Jinle Ke, Wenduo Feng, Wen Zhou, et al. 2023. Live4D: A Real-time Capture System for Streamable Volumetric Video. In *SIGGRAPH Asia 2023 Technical Communications*.

[81] Domenic Zingsheim, Patrick Stotko, Stefan Krumpen, Michael Weinmann, and Reinhard Klein. 2021. Collaborative VR-based 3D Labeling of Live-captured Scenes by Remote Users. *IEEE Computer Graphics and Applications* (2021).