NeVo: Advancing Volumetric Video Streaming with Neural Content Representation

Nan Wu
George Mason University
Fairfax, VA, USA
nwu5@gmu.edu

Bo Chen
University of Illinois at Urbana-Champaign
Urbana, IL, USA
boc2@illinois.edu

Ruizhi Cheng
George Mason University
Fairfax, VA, USA
rcheng4@gmu.edu

Klara Nahrstedt
University of Illinois at Urbana-Champaign
Urbana, IL, USA
klara@illinois.edu

Bo Han
George Mason University
Fairfax, VA, USA
bohan@gmu.edu

Abstract

Offering high-quality immersive content is the ultimate goal of volumetric video streaming. Although point clouds and meshes are dominant volumetric representations, their limitations in depicting photo-realistic content often undermine user experience. The recent advent of neural radiance fields (NeRF) offers a promising alternative content representation with superior photo-realism. However, streaming NeRF-based volumetric videos over wireless networks to mobile headsets faces significant challenges, including substantial bandwidth usage because of the large frame size, degraded visual quality due to even a low packet loss rate, and content artifacts caused by performance optimizations (*e.g.,* remote rendering at the network edge). To address these challenges, in this paper, we introduce NeVo, a next-generation volumetric video streaming system for efficient delivery of neural content such as NeRF. NeVo incorporates the following innovations into a holistic system: (1) a novel method to model visibility of implicitly encoded neural content, thereby avoiding non-essential transmission to drastically reduce network data usage, (2) a lightweight, learning-based model for real-time content reconstruction after packet loss with carefully chosen data, and (3) judicious identification and selective delivery of intermediate data in edge-based NeRF rendering to effectively mitigate artifacts. Our extensive experiments indicate that compared with the state-of-the-art, NeVo saves up to 68.3% of bandwidth usage, maintains high visual quality despite packet loss, and enhances user experience by reducing artifacts.

CCS Concepts

• Information systems → Multimedia streaming; **• Computing methodologies → Mixed / augmented reality**.

Keywords

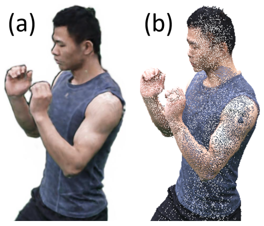
Volumetric Video Streaming and Neural Radiance Fields

ACM Reference Format:
Nan Wu, Bo Chen, Ruizhi Cheng, Klara Nahrstedt, and Bo Han. 2025. NeVo: Advancing Volumetric Video Streaming with Neural Content Representation. In *The 31st Annual International Conference on Mobile Computing and Networking (ACM MOBICOM '25), November 4–8, 2025, Hong Kong, China.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3680207.3723473

1 Introduction

Holographic communication [24], a key use case envisioned for 6G [74, 80], centers on achieving an interactive and engaging user experience. This communication paradigm benefits from delivering immersive content to represent 3D scenes, allowing users to change viewing perspectives with six degrees of freedom (6DoF) motion. While recent efforts [35, 43, 51, 52, 91, 99, 100] focus on the streaming of volumetric videos based on explicit 3D representations such as point clouds [16] and meshes [11], they oftentimes fall short of photo-realistic rendering [39, 47, 94, 95], degrading the quality of experience (QoE). This issue arises because point clouds and meshes struggle to accurately depict dynamic elements [62, 69] and lighting effects [25, 83], owing to their nature of *discrete* content representation.

Neural radiance fields (NeRF) [56] is a novel, implicit content representation that models a 3D scene with a *continuous*



	Quality	Bandwidth
NeRF	High	High
PtCl	Low	Medium
NeVo	High	Low

Figure 1: Left: volumetric content represented by NeRF (a) has better visual quality than point cloud (b). Right: Qualitative comparison of NeVo **and volumetric video streaming with vanilla NeRF and point clouds (PtCl).**

function by training a multi-layer perception (MLP) model. As shown in Figure 1 (left), NeRF can capture the detailed appearance of the human body and demonstrates much better visual fidelity than point clouds, which often result in low-quality content (*e.g.,* visible holes) due to discretized point distribution. While NeRF offers a more visually appealing experience, which makes it a suitable alternative for delivering volumetric content over wireless networks to mobile headsets, realizing NeRF-based volumetric video streaming in practice poses the following key challenges.

• *Bandwidth*: Delivering high-quality neural content incurs significant bandwidth consumption, for example, >1 Gbps when streaming vanilla NeRF [76] (*i.e.,* its MLP model) at 30 frames per second (FPS), which is required for real-time volumetric video streaming [32, 35, 41, 51, 91, 99].

• *Resiliency*: Due to the large size of neural content [56, 57, 63, 71, 84], even a low packet loss rate may result in numerous lost packets, leading to poor visual quality or video stalls.

• *QoE*: Content reprojection, which is essential for streaming remotely pre-rendered[1] neural content with predicted future viewport by aligning it with users' actual viewport [4], may produce artifacts [33, 42, 51, 52].

In this paper, we present NeVo, which is, to the best of our knowledge, the first volumetric video-on-demand (VOD) streaming system that explores neural content representations. NeVo employs an edge-assisted architecture, where the edge fetches neural content from a server, performs NeRF rendering, and sends rendered content to the client. We qualitatively compare NeVo and volumetric video streaming with vanilla NeRF and point clouds in Figure 1 (right). To address the aforementioned challenges, NeVo incorporates the following innovative solutions into a holistic system.

Optimizing Network Bandwidth Consumption (§3.2). State-of-the-art streamable NeRF in the computer vision community such as ReRF[2] utilizes shallow MLPs and voxels that store color features and density values of the scene to accelerate rendering [84]. These voxels are referred to as *feature voxels* (*i.e.,* feature values arranged in a 3D grid). Even after compression, ReRF may still require a bandwidth of 150+ Mbps. While visibility-aware optimizations, which selectively deliver mainly visible volumetric content, can reduce bandwidth consumption, they are typically designed for point-cloud-based systems, for which the visibility of points can be simply estimated based on their positions [35]. However, NeRF captures complex light interactions within a scene by considering refraction, reflection, and scattering of light, *making the existing definition of visibility ill-suited.* For instance, the content behind semi-transparent objects (*e.g.,* window glass) remains visible, but position-based visibility-aware optimizations deem it invisible due to occlusion.

Our key observation is that the weights (*e.g.,* opacity and transmittance) of sampled points in NeRF's ray marching[3], defined by us as *neural visibility*, indicate their importance to neural rendering. Thus, NeVo quantifies the importance of a feature voxel by examining the neural visibility of all sampled points in it and filters out unimportant voxels via learned thresholds, which can greatly reduce bandwidth consumption without noticeable visual quality degradation.

Recovering from Content Loss (§3.3). Packet loss includes both packets dropped during transmission and those not received before the decoding deadline [23, 45, 55], posing significant challenges to NeRF-based volumetric video streaming. Traditional schemes usually rely on retransmission, which causes extra latency, or forward error correction (FEC) [3] that faces difficulty in accurately predicting packet loss rate for adding a proper amount of redundant data for recovery [23, 55]. Recent endeavors for 2D video streaming [23, 45, 93] show that deep-learning models can reconstruct lost content with correctly received data. However, since the raw data size of frames with neural content for model training is much larger than that of 2D video frames (*e.g.,* ~800 MB [84] *vs.* ~6 MB for 1080p), a recovery model for neural content could be complex, *resulting in substantial computation latency and memory usage* (§3.3).

Our key observation is that NeRF content is spatially correlated in a confined space and temporally correlated within a short time window, which enables a potentially significant reduction in training data. NeVo leverages such spatial-temporal correlation to restore missing content from judiciously selected data, instead of entire frames, by training a simple yet effective model for real-time reconstruction. While 2D videos can also leverage the correlations for reconstruction, their efficiency gains may be less than NeVo. The

[1]Remote rendering of neural content on an edge server is necessary, due to the high computation overhead of NeRF, even with recent advancements in lightweight models and accelerated rendering [46, 63, 71, 73, 84].
[2]Note that ReRF [84] lacks an end-to-end system design. We integrate ReRF into our streaming framework to evaluate its performance in §5.

[3]Ray marching samples 3D points along rays from the viewpoint to each pixel of the rendered image by accumulating their color and density values.

reason is that 2D videos are captured through 3D to 2D projection, where even a small content movement can result in a large displacement in 2D frames, especially when the content is close to the camera. This often necessitates training with a large area of content for accurate reconstruction.

Alleviating Reprojection Artifacts (§3.4). Reprojection is essential in streaming remotely rendered content over wireless networks to mobile headsets, by aligning pre-rendered content from the edge server with the user's actual viewport to correct potential mismatch [4]. Without reprojection, there may be visible drifts of displayed content, notably degrading user experience [42]. However, when background content occluded by the foreground in the pre-rendered view becomes visible in the client's actual viewport, artifacts (*i.e.,* missing pixels) will occur (§2.2). While delivering additional data for occluded content can mitigate these artifacts, the challenge lies in accurately and efficiently identifying the *potential artifact locations from implicitly encoded neural content* on the edge server, without knowing users' actual viewport.

Our key observation is that when the previously occluded background becomes visible, missing pixels are usually located near the contours of foreground objects (§3.4). The distribution of opacity, a key attribute of ray-marching sampled points, can be used to identify contours. Thus, NeVo alleviates reprojection artifacts by transmitting only neural content around those contours.

Implementing and Evaluating NeVo **(§4, §5).** We build a prototype of NeVo and thoroughly evaluate its performance via controlled experiments and an IRB-approved user study.

- Compared with ReRF [84], NeVo reduces network data usage by up to 68.3%. Meanwhile, it does not affect visual quality, as its structural similarity index measure (SSIM) [87], a well-known metric for visual quality, is higher than 0.98 [26].
- NeVo's content recovery enables high-quality streaming in networks experiencing packet loss (*e.g.,* improving SSIM from 0.759 to 0.902 when packet loss rate is >50%).
- NeVo effectively mitigates reprojection artifacts. When the viewport prediction error is >4 cm (covering >50% of traces in a large viewport-trajectory dataset [91]), it increases SSIM from 0.899 to 0.923 with noticeable quality improvements.
- Subjective evaluations of user experience from 122 participants demonstrate that NeVo outperforms point-cloud-based volumetric video streaming systems Vues [51] and ViVo [35] by 73.9% and 90.4%, respectively.

2 Background and Motivation

2.1 Background

Conventional Volumetric Content Representations in existing streaming systems primarily center on explicit geometric structures such as point clouds [32, 35, 51, 91, 99, 100] and meshes [41, 49]. Point clouds are effective for non-manifold structures [28] but can lead to artifacts due to a lack of spatial connectivity [66, 75]. Meshes provide detailed surfaces and efficient rendering [17, 50], yet their fixed topology limits the modeling of dynamic topological changes [53, 58] and struggles with occlusions and optical effects [25, 83].

Neural Radiance Fields. Recent developments in neural networks result in advanced methods for volumetric content representations. Among them, NeRF [56] utilizes MLP to form an implicit and continuous depiction of scenes, enabling the rendering of photo-realistic content. Existing research on NeRF mainly explores efficiency optimizations through techniques such as sparse geometric representations [18, 31, 78, 84, 97], voxel compression [79], and voxel decomposition [12]. Beyond that, more recent research extends NeRF to represent dynamic scenes [30, 60, 61, 71, 73, 85].

NeRF Rendering leverages ray marching for content creation [56]. Taking the viewer's position as the origin, it generates rays for pixels of the to-be-rendered content, samples points on the ray, and derives their color with the trained model. These points are used to synthesize the pixels as follows: $C(r) = \sum_{i=1}^{N} T_i \cdot \alpha_i \cdot c_i$, where $\alpha_i = 1 - \exp(-\sigma_i \delta_i)$ and $T_i = \prod_{j=1}^{i-1}(1 - \alpha_j)$. Here, $C(r)$ denotes the color of a rendered pixel, derived from the cumulative contributions of N sampled points along ray r. Each point's contribution is determined by its color c_i, alpha value α_i, and accumulated transmittance T_i [56]. α_i, reflecting the point's opacity, is influenced by its density σ_i and the distance between samples δ_i. The transmittance T_i represents the cumulative transparency along the ray up to the i-th point. Thus, NeRF accumulates the color and intensity of sampled points as light travels in the 3D space.

Streamable NeRF. NeRF was initially designed for static scenes, and early attempts to extend it for dynamic scenes involved adding the time dimension to NeRF [30, 92] or using an additional MLP to learn deformations [60, 61, 65]. To handle long-sequence videos, ReRF [84] relies on groups of features (GoF), wherein each group starts with a leading intra-coded frame (*i.e.,* I-frame) that encapsulates complete voxel features, followed by multiple predictive frames (*i.e.,* P-frames) containing only *motion vectors and residual features.* Moreover, ReRF employs principal component analysis (PCA) [34], 3D discrete cosine transformation (DCT) [44], and entropy coding [84] for variable bitrate encoding.

2.2 Motivational Study

Visibility-aware Optimizations. Visibility-aware optimizations introduced in ViVo [35] are pivotal for reducing bandwidth usage by selectively delivering mainly visible volumetric content for a given viewport. However, they are designed for point-cloud-based systems, for which content visibility

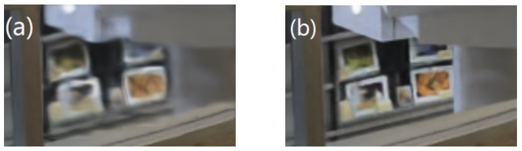


Figure 2: Content rendered with (a) and without (b) traditional visibility-aware optimizations.

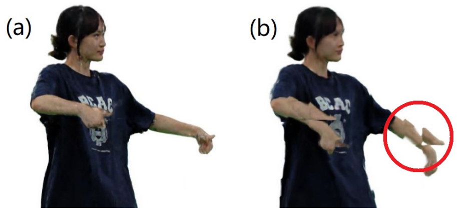


Figure 3: Ground truth (a) and rendering artifacts (b) caused by packet loss.

can be simply estimated based on points' positions and is agnostic to the content itself. In Figure 2, we present a scenario where the content behind a window glass is visible to the user. We follow the occlusion-aware algorithm in ViVo [35] to reduce the visual quality of "occluded" content. As observed, this approach overlooks complex light interactions captured in NeRF (*e.g.,* with semi-transparent materials), leading to an unintended reduction in the quality of content behind the glass (Figure 2(a)). Ideally, as shown in Figure 2(b), the quality should remain high due to the content's visibility.

By leveraging the unique properties of ray marching in NeRF rendering, NeVo introduces novel neural visibility to assess voxels' importance to rendering, which allows for selective transmission that reduces bandwidth usage without compromising visual quality.

Reliable Streaming. Packet loss can make feature voxels not decodable, leading to missing voxels that degrade visual quality (*e.g.,* distorted hands shown in Figure 3). While retransmission and FEC [3] have been proposed to handle lost packets in 2D video streaming, they are impractical in this context (as discussed in §1). Recent endeavors in 2D video streaming have shown that deep-learning models can reconstruct lost content by utilizing correctly received data [14, 23, 45, 93]. However, because the size of NeRF content is significantly larger than that of a 2D video frame (*e.g.,* 800 MB [84] vs. 6 MB in raw data), applying such recovery models in this context results in substantial computational latency and GPU memory usage. We train a model that utilizes full frames as inputs. The results show a reconstruction time of >60 ms and memory usage of >16 GB on an NVIDIA GeForce RTX 4090 GPU, making this approach unsuitable for NeRF-based volumetric video streaming.

These challenges motivate the design of NeVo's lightweight learning-based mechanism, which reduces input size

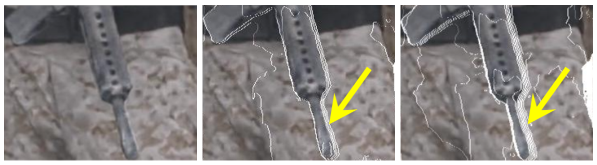


Figure 4: Ground truth and reprojection artifacts with translational prediction error of 10 cm (middle) and 20 cm (right).

by focusing on the most relevant voxels across frames, enabling timely recovery of lost content.

Content Reprojection. Existing schemes [4, 42] benefit from RGB and depth data[4] of displayed content for the reprojection into different viewports. However, as shown in Figure 4, for a pre-rendered view on the edge, the absence of RGB and depth data for occluded content leads to missing pixels after reprojection to the actual view on the client. While Outatime [42] leverages interpolation to fill the pixels, it often causes blurred artifacts. To address this problem, additional data for occluded content is needed. A naive solution is to transmit all ray-marching sampled points to the client for reprojection, since they include both visible foreground and occluded content, allowing for reprojection to the user's actual viewpoint without missing pixels. However, doing this significantly increases bandwidth usage (*e.g.,* ~2.4 Gbps after compression), and reprojection with all points is computationally intensive (*e.g.,* <1 FPS on HoloLens 2 [2]).

These challenges motivate NeVo to design an efficient reprojection approach for remote-rendered NeRF content, which transmits only critical points that can effectively fill missing pixels, reducing bandwidth usage and computational overhead while maintaining high reprojection quality.

NeRF vs. 3D Gaussian Splatting. Compared with NeRF, 3D Gaussian splatting (3DGS) [39] has recently demonstrated notable advancements in reducing rendering overhead and accelerating training, by augmenting point clouds with learned Gaussian parameters. However, despite these improvements, 3DGS and NeRF usually render content with comparable visual quality [39, 48, 89], and NeRF can already achieve the 30-FPS rendering required by video streaming on commercial machines with efficiency optimizations [31, 57, 97]. On the other hand, 3DGS may require higher bandwidth for video streaming. For instance, streaming the *longdress* video from the 8i [1] dataset using ReRF [84] necessitates a bandwidth of ~210 Mbps, while it may demand >400 Mbps for SpacetimeGaussians [48], a state-of-the-art 3DGS scheme. Thus, we focus on NeRF in this paper as a promising starting point for advancing volumetric video streaming and leave the exploration of 3DGS as future work.

[4]An accurate depth image boosts reprojection quality by offering a 3D understanding of the scene [42].

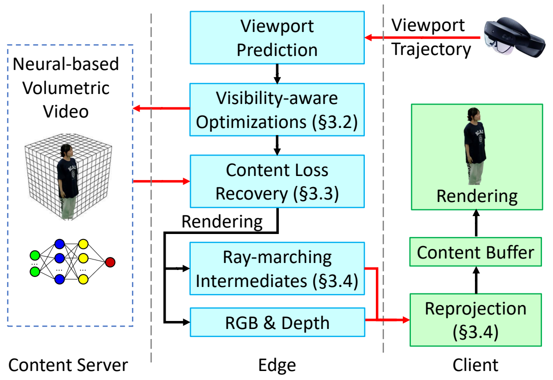

Neural-based Volumetric Video
Content Server
Viewport Prediction
Visibility-aware Optimizations (§3.2)
Content Loss Recovery (§3.3)
Rendering
Ray-marching Intermediates (§3.4)
RGB & Depth
Edge
Viewport Trajectory
Rendering
Content Buffer
Reprojection (§3.4)
Client

Figure 5: System architecture and workflow of NeVo.

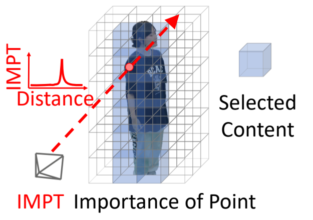


Figure 6: Ray-marching-aware importance modeling and voxel selection.

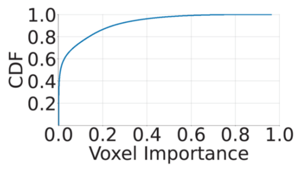


Figure 7: CDF of voxels' importance: ~60% show low importance for rendering.

3 System Design of NeVo

3.1 Overview

Similar to the design of traditional content delivery networks (CDN), in NeVo, the content server stores NeRF-based volumetric videos, and mobile clients request data from it [82]. However, due to the high computational demands of rendering NeRF-based video, the client should offload computation-intensive tasks to an edge server for real-time NeRF rendering. Unlike traditional 2D video, NeRF stores content as a large set of parameters. As introduced in §2.2, even with compression, fetching the entire parameter set can consume significant bandwidth (*e.g.,* >200 Mbps). To address this challenge, NeVo optimizes the streaming between the content server and the edge server to reduce bandwidth consumption and mitigate the effects of packet loss. Since this optimization relies on predicting the viewer's viewport, we do not prefetch or buffer video frames several seconds in advance, even though NeVo targets VoD streaming. This avoids potential degradation of streaming quality that is caused by increased prediction errors associated with the extended prediction window size [35]. The components in edge-to-client streaming address the visual artifacts in reprojection caused by inaccurate viewport prediction.

Figure 5 depicts the system architecture of NeVo. Based on the viewport trajectory sent by NeVo's client, the edge fetches NeRF's feature voxels from the content server with visibility-aware optimizations (§3.2). When packet loss happens, NeVo reconstructs missing feature voxels with a learning-based approach (§3.3). To accommodate potentially inaccurate viewport prediction on the edge, NeVo's edge selects and transmits ray-marching intermediates, which is utilized to mitigate reprojection artifacts (§3.4).

3.2 Optimizing Bandwidth Consumption

Problem & Challenges. High bandwidth consumption is a critical issue in NeRF-based volumetric video streaming (§2.1), which motivates us to incorporate visibility-aware optimizations for reducing network data usage. However, NeRF captures intricate light characteristics to represent phenomena such as refraction, reflection, and scattering, *making the existing definition of visibility ill-suited.* As illustrated in §2.2, existing optimizations target explicit content representations such as point clouds [35], where content visibility can be simply estimated based on points' locations. These optimizations cannot be directly applied to NeRF-based content, as they may fail to determine the visibility of feature voxels.

Solution. Our key insight is that in ray marching for NeRF rendering, the color of a pixel, given by the function $C(r) = \sum_{i=1}^{N} T_i \cdot \alpha_i \cdot c_i$, is the weighted sum of sampled points' colors along the marched ray. Importantly, the removal of points with low weight negligibly affects visual quality. Thus, *the weight can reflect the neural visibility* of sampled points, indicating their importance to neural rendering. Based on this observation, we propose to define a voxel's importance as the highest neural visibility of all ray-marching sampled points inside it and select only important voxels to stream, as shown in Figure 6. More specifically, we leverage the weight $T_i\alpha_i$ (§2.1) in NeRF's ray marching algorithm [56] to describe sampled points' neural visibility.

Figure 7 shows the cumulative distribution function (CDF) of the importance scores for non-empty voxels across 22 widely-used videos (§5.1), with each frame tested against 300 different viewports. We observe a long-tail pattern, indicating that the majority of voxels have relatively low importance. For example, with a threshold of 0.025, ~60% of voxels could be removed from delivery. We measure the SSIM of rendered content after removing those voxels, with the ground-truth content rendered with all voxels involved. The SSIM consistently exceeds 0.98 (*i.e.,* visually lossless [26]), indicating that removing these voxels does not impact visual quality, while greatly reducing the amount of network data.

The threshold selection necessitates careful design to balance the trade-off between bandwidth consumption and rendering quality. While we can increase the threshold to filter out more voxels and further reduce bandwidth consumption, it leads to degraded rendering quality. For instance, in the *kpop* video from the ReRF dataset, increasing the threshold from 0.025 to 0.04 reduces the SSIM from 0.986 to 0.959, but the data saving increases from 62% to only 64%. To balance the above trade-off, we propose to dynamically adjust the importance threshold as a per-video hyperparameter of the trained NeRF content. This is done by adding another loss function to optimize the threshold, in addition to the one for NeRF training. We include SSIM in this loss function, defined as $Loss = SSIM_T - SSIM_C$, to guide the hyperparameter selection. If the current $SSIM_C$ falls below the target $SSIM_T$ (*e.g.,* 0.98 for lossless visual quality), the loss function will be positive, prompting a decrease in the threshold to retain more voxels and improve visual quality. The computational demand of hyperparameter tuning primarily involves rendering NeRF content from various viewports, which is lightweight compared with training the NeRF content. For example, using an NVIDIA GeForce RTX 4090 GPU, we render 3,000 viewports for each frame during the threshold tuning, and it adds less than 7% overhead to the training time.

Ideally, when the edge fetches feature voxels of a video, it should calculate the voxels' importance scores based on the predicted viewport of the frame and filter out unimportant voxels with the video's threshold. However, the to-be-fetched frame is not yet available for the edge to determine voxels' importance scores. NeVo addresses this issue by first selecting important voxels based on the just received frame and then adjusting the voxel selection with motion vectors (§2.1) that could be delivered several frames ahead. For example, the edge could fetch the feature voxels of frame T along with the motion vectors of frame T+3. The vectors are lightweight (*e.g.,* 8 KB for each frame) and could be kept in memory for several frames (*e.g.,* until frame T+3 is rendered).

NeVo includes additional voxels around the selected ones, extending coverage by 20 cm, to accommodate potentially inaccurate viewport prediction. Assume a prediction window of 132 ms (*i.e.,* four frames), covering the computational latency on NeVo's edge and client (§5.5) and the transmission delay of 5G networks (§5.1). When the viewer moves at an average speed of 1.42 m/s [10], the translational error without prediction is ~18.8 cm.

3.3 Recovering from Content Loss

Problem & Challenges. Packet loss can lead to missing voxels, which results in noticeable artifacts in rendered content or video stalls, degrading user experience (§2.2). Thus, effectively recovering missing voxels is essential to maintain a high QoE in NeVo. As illustrated in §2.2, recent approaches [23, 45, 54, 93] for 2D video streaming indicate that leveraging deep-learning models to recover lost content from correctly received packets is a promising method to achieve reliable real-time streaming. However, the large size

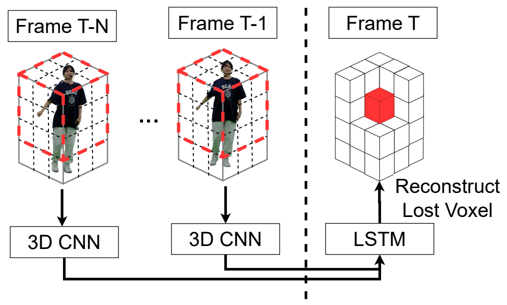


Figure 8: VRM **model architecture for content reconstruction. The input size is adjustable and can be larger than the illustrated 3×3×3 grid, depending on content movement.**

of NeRF content can lead to a complex recovery model with substantial computational latency and memory usage.

Solution. Our key observation is that content movement in neural-based volumetric videos is usually *confined to a limited number of feature voxels in 3D space.* For instance, when the content moves at 2 m/s (*i.e.,* the upper bound of walking speed), the motion vector between frames is typically within two feature voxels. This confined movement shows strong inter-frame content similarity, aligning with principles widely used in 2D video compression techniques [77, 88]. Thus, instead of using all voxels from previous frames as inputs, NeVo reconstructs a missing voxel by tracing only voxels at the same location in historical frames and their neighbors within a confined grid determined by the extent of content motion. For example, when the movement spans two voxels, which covers >99% of movements in 21 diverse videos (§5.1), we select neighbors in a 3×3×3 grid. In extreme cases where movement spans multiple voxels, we enlarge the grid to encompass more neighboring voxels.

Based on this observation, we propose a lightweight voxel recovery model, named VRM, that takes a sequence of historical voxels in N previous frames and their neighbors as inputs, as shown in Figure 8. We process the data in each history frame with a 3D convolutional neural network (CNN) to identify spatial patterns of feature voxels. The extracted features from CNN are passed through long short-term memory (LSTM) [36] units to maintain temporal continuity across frames. The value of N, empirically set as 9 (§4), is determined by balancing memory usage, computation overhead, and reconstruction quality.

Our training goal is to ensure the resulting model can recover missing feature voxels, considering that correctly received voxels may vary due to *not only packet loss but also visibility-aware optimizations* (§3.2). Existing methods for 2D videos [23, 45] simulate packet loss by randomly masking data as lost during training. However, this approach is inadequate because the training data does not account for the availability of voxels for reconstruction that is caused by visibility-aware optimizations. Unlike 2D videos, where all content in a frame is delivered and the training of reconstruction models simulates only packet loss, VRM must generalize to available content based on voxels' neural visibility under different viewports. Although we can follow the modeling of voxel importance in §3.2 for data augmentation, the involved ray marching is time-consuming, especially when we augment with a large number of different viewports.

We introduce a lightweight method for coarse visibility determination. Instead of marching rays for pixels, we generate rays for the selected training voxels (*e.g.,* 27 rays for 3×3×3 neighbors) to reduce computation overhead. The rays start from the viewpoint and end at the centers of the selected voxels. As these rays traverse the scene, they intersect with other feature voxels, including those not used for training but in front of the selected ones on the rays. These voxels are used to determine the neural visibility of the selected voxels (§3.2), and we mask voxels that will not be delivered by setting their values to zero. Our tests show that compared with pixel-level ray marching, this approach significantly reduces data augmentation time by >100× while offering comparable performance in reconstructing content.

3.4 Alleviating Reprojection Artifacts

Problem & Challenges. In a remote rendering system, reprojection plays a critical role in stabilizing the content by adjusting pre-rendered content to match the actual viewport of the headset. However, the absence of RGB and depth data for occluded content leads to missing pixels after reprojection, and existing methods that deliver additional data for occluded content are not practical for NeRF-based volumetric video streaming (§2.2). For example, while transmitting all ray-marching sampled points for reprojection ensures high visual quality, it leads to significant bandwidth usage and computational demand. Thus, to reduce the overheads, the challenge lies in *accurately and efficiently identifying potential locations of missing pixels on the fly from implicitly encoded NeRF content* without knowing the headset's actual viewport on the edge server.

Solution. Our key observation is that missing pixels usually *emerge near the contours of foreground objects in the rendered content*, particularly where the previously occluded background becomes visible under the current viewport. These missing pixels result from the parallax effect [9], which is influenced by viewport prediction errors and the relative depth between foreground and background content. This can be described mathematically as $M = d \times (1 - D_F/D_B)$, where M represents the width of missing content starting from each pixel located on the contours of foreground objects, d corresponds to viewport prediction error, and D_F and D_B are the distances of the foreground and background content to the viewpoint, respectively. To address the missing pixels, we propose to transmit intermediate sampled points on rays that intersect areas around these contours to fill missing pixels. Specifically, our solution involves (a) identifying the contours from the content rendered on the edge and (b) delivering additional points around them.

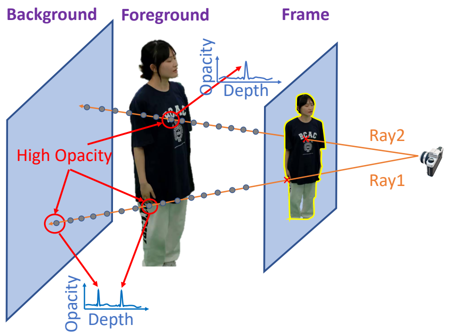


Figure 9: The distribution of points' opacity determines whether a ray intersects an object's contour.

Contour Identification. While existing edge detection [5] can be used for contour identification, it may include contours not in areas that occlude background content, wasting network bandwidth. Removing these irrelevant contours can introduce substantial processing latency (*e.g.,* >10 ms on the 8i dataset [1]). Thus, accurately and efficiently identifying the contours in neural content necessitates a nuanced examination of ray-marching sampled points. Our key insight is that a *marched ray with multiple clusters of high-opacity sampled points* is likely to intersect the contours of foreground content, as illustrated in Figure 9. When *Ray1* crosses the contour, points with high opacity values can be observed in both foreground and background content due to blending at this transition zone. However, when *Ray2* traverses the non-contour area, only high-opacity points near the foreground content are usually clustered. This observation allows us to leverage the opacity and depth of sampled points to efficiently identify the required points for reprojection.

To identify contours, we compute the weighted standard deviation of the depth of sampled points along each ray with their opacity as the weight. The depth values are used as an indication of whether high-opacity points along each ray are clustered. A high deviation suggests a ray intersects the contour of foreground content, on which high-opacity points are distributed across varying depths. It is essential to establish an appropriate threshold for this deviation to balance visual quality and bandwidth consumption. A higher threshold selects fewer contour areas, which may lower bandwidth usage but risks missing important contours. On the other hand, a lower threshold potentially improves reprojection quality by including more contour areas, while increasing bandwidth consumption.

To balance the above trade-off, we adaptively adjust the threshold for each video, adding it as a hyperparameter to the trained NeRF content. Let $A_C(T)$ represent the area of pixels covered by the selected sampled points after reprojection, which is affected by the threshold T, and let A_M denote the area of known missing pixels. We further define $A_I(T) = A_C(T) \cap A_M$, which is the overlapped area, and $A_U(T) = A_C(T) \cup A_M$, which is the total area covered by $A_C(T)$ and A_M. The loss function is defined as $Loss = 1 - \frac{A_I(T)}{A_U(T)}$, which penalizes both false negatives (the sampled points could not fill all the missing pixels) and false positives (sampled points are reprojected to non-missing pixels). This optimizes the threshold T to make the selected sampled points effectively fill only missing pixels.

Adaptive Point Delivering. Given that missing pixels usually emerge around the detected contours, NeVo adaptively selects and delivers ray-marching sampled points for reprojection. We dynamically expand each pixel in the identified contour to a circular area with a radius of M and transmit only ray-marching sampled points on rays within this area to the client. M is calculated based on the parallax effect [9], leveraging the relative distances of the foreground and background content to the viewpoint: $M = d \times (1 - D_F/D_B)$, and we set d as 20 cm, which is the upper bound of translational viewport prediction error in §3.2. This calculation accounts for the fact that as the viewer moves, the content closer to the viewpoint appears to move faster than that further away in the viewport. Thus, the closer content potentially reveals or obscures substantial portions of the content further away, leading to more missing pixels, which requires a larger M.

4 Implementation

Reducing Latency with Transmission Ordering. When streaming neural immersive content, prolonged end-to-end latency, including the time taken for content transmission and rendering, negatively impacts QoE by requiring a larger viewport prediction window, which leads to less accurate predictions [35, 67, 96]. The iterative nature of ray marching and reprojection of sampled points (§2.1) implies that along each marched ray, a received voxel/point with a higher index (further away from the viewpoint) cannot participate in rendering until all lower-index voxels/points (closer to the viewpoint) have arrived. As a result, the rendering of the scene can be notably delayed. To address this, we implement a transmission ordering method, where voxels/points are streamed based on their depth ranges to the viewpoint, in ascending order from nearest to farthest.

System Implementation. We implement the content server and the edge of NeVo in Python on Ubuntu 20.04, and develop a prototype of NeVo client on Microsoft HoloLens 2 [2] using Unreal Engine v4.27 [7]. On the edge, we modify the code from ReRF [84] for rendering. The communication is based on WebRTC (Web Real-Time Communication) [37] using the `aiortc` library [8], and for the edge-to-client streaming, we leverage existing designs for 2D videos to handle packet loss [68]. Similar to ViVo [35], we implement distance-aware optimization that adjusts the content quality based on its distance to the viewer, and rate adaptation that dynamically changes content quality based on network capacity.

For the VRM model, which handles content reconstruction, we simulate packet loss during training by randomly zero voxels for the augmented data (§3.3). To determine the optimal history window size, we compare the performance of VRM with window sizes of 9, 12, and 15 frames. Since the visual qualities for these settings are similar, we set the window size to 9 frames to reduce memory consumption and computation overhead. We train a VRM model for each video, given its better performance than training a model with all videos (§5.4). We utilize an Adam optimizer [40] with a learning rate of 0.001 and a batch size of 1024, and we train the model for a maximum of 50 epochs. The model with a size of ~15 MB is fetched by the edge from the content server at the startup stage. Our implementation consists of 7,000+ lines of code (LoC): 1,300+ for the content server, 3,000+ for the edge, 2,200+ for the client, and 500+ for the VRM model.

5 Performance Evaluation

5.1 Experimental Setup

Devices. Our client device is Microsoft HoloLens 2 [2] with a Qualcomm Snapdragon 850 chip. The edge server has an AMD Ryzen 9 7900X CPU with 12 cores, 32GB memory, and an NVIDIA GeForce RTX 4090 GPU. We set up a commodity machine with an Intel i7-11700 CPU as the content server.

Network Conditions. We connect the client, edge, and content server with a Linksys MR7500 WiFi router. The content server and the edge are connected with an Ethernet of ~800 Mbps throughput. The edge and client are wirelessly connected with ~450 Mbps throughput. We assess NeVo's effectiveness under fluctuating/limited bandwidth with packet loss in a controlled setting. We emulate the packet loss with real-world traces released by Hairpin [55]. For the link between the content server and the edge, we use `tc` [6] to replay three distinct traces with average bandwidths of 76.7±6.3 Mbps, 102.4±13.9 Mbps, and 150.4±35.6 Mbps, similar to those used by ViVo [35]. We increase the round-trip time between the content server and the edge to ~40 ms (*i.e.,* a typical latency within the U.S. [27]). For the link between the edge and the client, we increase the round-trip time to ~30 ms, emulating a 5G network connection [70]. We replay three other traces collected from different sites of a major U.S. cellular network, with average bandwidths of 50.1±5.4 Mbps,

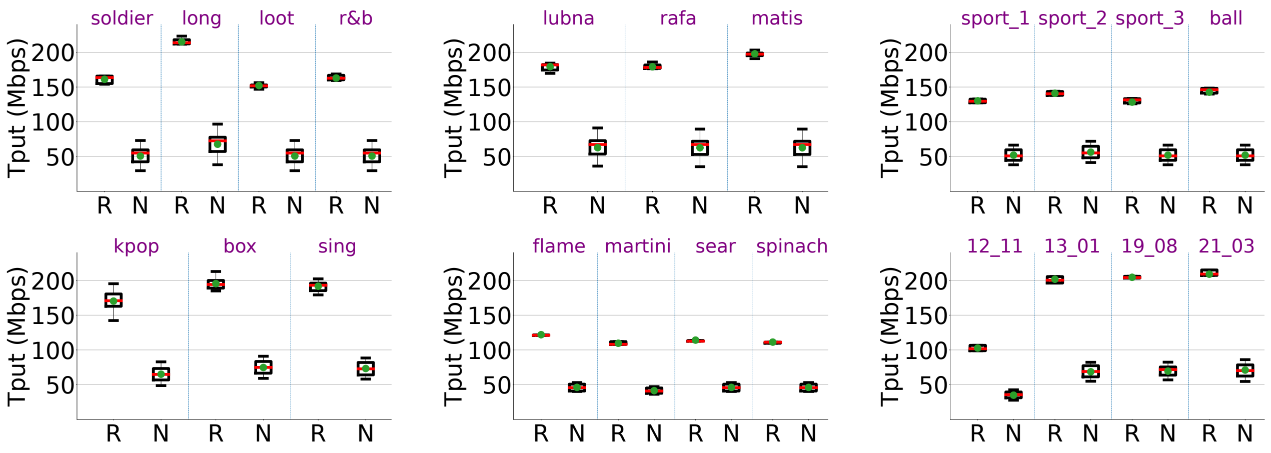


Figure 10: Throughput of ReRF (R) and NeVo (N) on 22 videos from six datasets.

61.2±6.6 Mbps, and 80.3±7.6 Mbps to represent different 5G network conditions [70].

Videos. To verify NeVo is generic across different NeRF content, we select 22 videos from six datasets for performance evaluation: *soldier*, *longdress*, *loot*, and *red and black* from 8i [1], *lubna*, *rafa*, and *matis* from V-SENSE [59], *sport_1*, *sport_2*, *sport_3*, and *basketball* from NHR [90], *kpop*, *box*, and *sing* from ReRF [84], *flame_steak*, *coffee_martini*, *sear_steak*, and *cook_spinach* from NV3D [46], and *0012_11*, *0013_01*, *0019_08*, and *0021_03* from DNA-Rendering [22]. The videos are generated offline by leveraging ReRF's libraries [84]. The NHR [90], ReRF [84], NV3D [46], and DNA-Rendering [22] datasets provide images taken from different angles for training NeRF videos. We render the 8i and V-SENSE datasets' high-quality point clouds to images from different viewports and use them to train NeRF videos.

Baselines. In this paper, we adopt ReRF [84] as the baseline system. Compared with other streamable NeRF (*e.g.,* MLP-maps [63] and Tensor4D [71]), ReRF achieves comparable visual quality, while handling long-sequence videos. Moreover, ReRF consumes less bandwidth, for example, requiring <150 Mbps on the *basketball* video, while MLP-maps and Tensor4D necessitate >300 Mbps. Since ReRF lacks an end-to-end system design, we integrate it into our streaming framework for performance evaluation. We also compare NeVo with ViVo [35], which streams point-cloud-based volumetric videos, and Vues [51], which leverages an edge server to transcode point clouds into 2D streams. For their comparison with NeVo, we use the 8i and V-SENSE point cloud datasets. We re-implement ViVo and Vues on HoloLens 2, and our results align well with their reported ones.

Viewing Traces. For 8i [1] and V-SENSE [59] videos, we use viewing traces released by Theia [91] with 52 participants. For videos from NHR [90], NV3D [46], DNA-Rendering [22] and ReRF [84], we use the simulated viewing traces provided by their authors. For a fair comparison, all systems use the same viewport prediction method in ViVo [35].

Metrics. We evaluate the network throughput and latency of NeVo under both unthrottled and fluctuating/limited bandwidth and monitor the CPU and GPU utilization on the edge and the client. For visual quality, we employ SSIM [87] and LPIPS [101] to compare point cloud and NeRF representations, as well as the impacts of our proposed components in NeVo. Moreover, we conduct a large-scale IRB-approved user study to evaluate the real-world user experience of NeVo.

5.2 Optimizing Bandwidth Consumption

In this section, we assess the proposed optimizations for reducing bandwidth consumption when streaming from the content server to the edge (§3.2). The baseline is ReRF [84] that fetches all feature voxels at the highest quality. We conduct our experiments on unthrottled networks.

Figure 10 compares the throughput of NeVo and ReRF with 22 videos from the six datasets with diverse bandwidth requirements, plotting the 25th and 75th percentiles, medium, mean (green dots), and lower and upper whiskers [29]. As NeVo omits feature voxels with low neural visibility, its average throughput is 68.3%, 66.1%, 60.8%, 61.8%, 60.7%, and 65.9% lower than ReRF on the six datasets, respectively. Table 1 shows the visual quality of NeVo in SSIM [87] and LPIPS [101] compared with the ground-truth content rendered with all voxels (*i.e.,* the baseline ReRF). We observe that NeVo can achieve >0.98 SSIM (visually lossless [26]) and its LPIPS is close to 0 (high similarity with ground truth [101]), validating that our trained threshold for delivering only voxels with high neural visibility can effectively optimize bandwidth usage while maintaining good visual quality. Compared with ReRF, NeVo adds only <1 ms latency for determining the neural visibility of voxels.

Video	SSIM↑	LPIPS↓
8i	0.981±0.007	0.011±0.003
V-SENSE	0.983±0.007	0.008±0.001
NHR	0.983±0.008	0.008±0.002
ReRF	0.985±0.007	0.006±0.001
NV3D	0.984±0.009	0.008±0.002
DNA	0.981±0.011	0.010±0.003

Table 1: SSIM and LPIPS of NeVo after optimizing the bandwidth consumption between content server and edge.

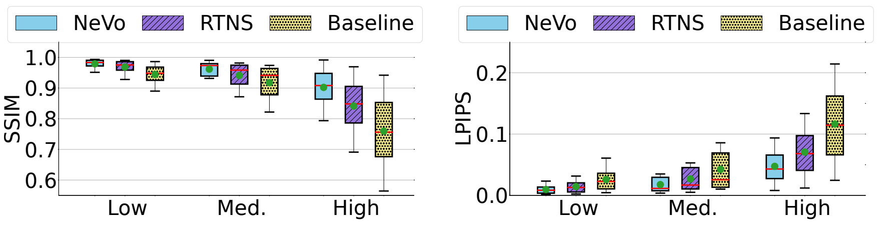


Figure 11: SSIM and LPIPS of NeVo, retransmission (RTNS), and baseline under low, medium, and high packet loss rates.

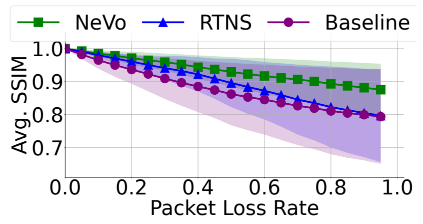


Figure 12: SSIM of NeVo, RTNS, and baseline under different packet loss rates.

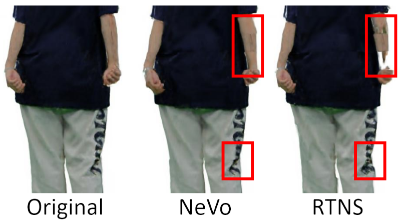


Figure 13: A sample frame from NeVo and RTNS under an 80% loss rate.

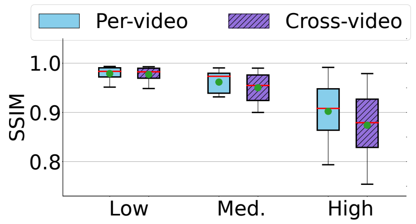


Figure 14: SSIM of per-video and cross-video trained models.

5.3 Recovering from Content Loss

We next evaluate the performance of the loss recovery model VRM for streaming from the content server to the edge (§3.3), under limited/fluctuating bandwidth. The baseline reuses feature voxels from the previous frame when packet loss results in missing voxels. Additionally, we compare NeVo with a retransmission-based method (RTNS) to recover lost voxels, while still reusing feature voxels if packets miss the rendering deadline. For comparing the reconstruction quality, we replay the viewing traces and render content based on the feature voxels from VRM, RTNS, and the baseline. The ground truth is rendered for scenarios without packet loss. The loss recovery model of NeVo incurs limited latency (<5 ms under all packet loss rates), ensuring that the content can be reconstructed on time.

Figure 11 compares the SSIM and LPIPS of content rendered by NeVo, RTNS, and the baseline under networks with low (<25%), medium (25% to 50%), and high (>50%) packet loss rates [23, 45]. NeVo outperforms other methods in all conditions, with >0.9 SSIM on average, indicating good visual quality [26]. When the packet loss rate is <25%, the average SSIM of NeVo is ~0.98, indicating lossless visual quality [26], and outperforms RTNS and the baseline by 0.011 and 0.035, respectively. When the network experiences medium loss rates, NeVo achieves an average SSIM of 0.962, outperforming RTNS and the baseline by 0.021 and 0.044, respectively. The advantages of NeVo are more pronounced under high packet loss rates (>50%), with an SSIM of 0.902, exceeding RTNS and the baseline by 0.061 and 0.143 on average, respectively. Moreover, when the packet loss rate is high, ~55% and ~77% of frames in RTNS and the baseline, respectively, have poor visual quality (SSIM <0.86) [26], whereas, for NeVo, only ~24% of frames exhibit subpar visual fidelity. Regarding LPIPS (where lower is better), NeVo achieves the lowest score, on average, across all packet loss conditions, indicating better reconstruction quality than RTNS and the baseline.

Figure 12 compares the SSIM of NeVo with RTNS and the baseline on individual frames under varying packet loss rates across different videos. The quality drops of NeVo are notably lower than RTNS and the baseline. On average, NeVo's SSIM drops below 0.90 when the loss rate reaches 80%. This ensures good reconstruction quality as the packet loss usually spans from 0% to 80% [23]. Figure 13 visualizes a sample frame from NeVo and RTNS under an 80% packet loss rate, showing that NeVo reconstructs frames with high quality.

In Figure 14, we compare the performance of models trained per video and across all videos. The per-video model achieves better visual quality in all packet loss rates, demonstrating that adapting VRM to the unique content of each video leads to more accurate reconstruction. This enhancement is due to VRM 's ability to optimize specifically for the distinct characteristics of individual videos. Since VRM is lightweight, its offline training time is <1 minute per frame, much shorter than the >30 minutes per frame offline training time of NeRF-based videos.

5.4 Alleviating Reprojection Artifacts

In this section, we evaluate the performance of content reprojection in NeVo for edge-to-client streaming (§3.4). We assume no data loss during transmission between the content server and the edge to eliminate VRM's impact on visual

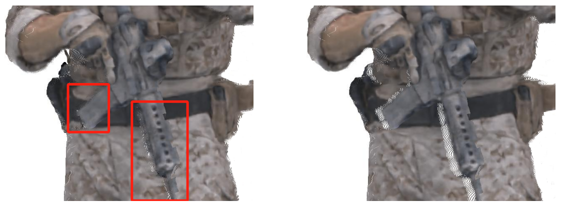


Figure 15: Incorporating NeRF's ray-marching sampled points (left) improves the visual quality compared to the baseline that relies on RGB and depth data (right).

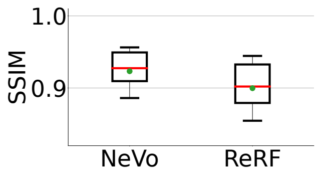


Figure 16: SSIM of reprojection in NeVo *vs.* ReRF (solely leverages RGB and depth data).

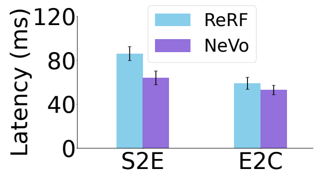


Figure 17: Latency comparison of ReRF and NeVo: S2E (content server to edge) and E2C (edge to client).

quality. The baseline is ReRF solely relying on RGB and depth data for reprojection. We replay the viewport traces for the 8i [1] and V-SENSE [59] datasets, captured when users view volumetric content on mobile headsets [91]. We set the viewport prediction window (for edge-to-client streaming) as 132 ms, which is the upper bound for the edge-to-client latency under 5G networks [70]. Specifically, we select the frames with a viewport prediction error >4 cm, which covers >50% of the viewing traces, to demonstrate the effectiveness of NeVo's reprojection design.

Figure 15 shows a qualitative comparison between the proposed reprojection approach in NeVo and ReRF. NeVo can reproject the content with high fidelity by judiciously leveraging ray-marching sampled points around the contours to fill the missing pixels, which are observed in ReRF. Figure 16 shows a quantitative comparison of NeVo and ReRF by calculating the SSIM with the ground truth rendered with the client's actual viewport. NeVo achieves good visual quality with an average SSIM of 0.923 [26]. On the other hand, ReRF simply leverages RGB and depth data for reprojection and results in a lower average SSIM of 0.899. Compared with the straightforward solution introduced in §3.4, which streams all ray-marching sampled points for reprojection (>2 Gbps), NeVo keeps bandwidth consumption at an acceptable level (§5.6) by selectively transmitting sampled points.

We also evaluate the trade-off between visual quality and bandwidth consumption when edge-to-client latency increases. In this case, the viewport prediction window must be increased to anticipate a more distant future viewport. However, increasing the window size amplifies prediction errors [35]. Based on the function $M = d \times (1 - D_F/D_B)$ used in §3.4, we need to expand the areas around the contours of foreground objects to reduce missing pixels after reprojection, while this leads to increased bandwidth consumption. For example, when the prediction window size is 132 ms, the average edge-to-client bandwidth consumption is 14.4±3.7 Mbps, and it can be more than 30 Mbps when the window size increases to 300 ms. On the other hand, if the expansion areas are not adjusted, the SSIM may drop below 0.85, leading to a noticeable degradation in visual quality.

5.5 End-to-end Latency

Next, we evaluate the end-to-end latency by breaking it down into two components:

- Content server to the edge (S2E): The time from when the content server begins transmitting NeRF content until the edge finishes rendering the content.
- Edge to the client (E2C): The time from when the edge starts streaming rendered content until the client puts the content into the rendering buffer.

Figure 17 compares NeVo with ReRF under fluctuating and limited bandwidth. The latency under unthrottled networks shows similar patterns and is thus omitted. NeVo achieves a 25.8% reduction for S2E (~64 ms *vs.* ~86 ms) and a 10.4% reduction for E2C (~53 ms *vs.* ~59 ms). The end-to-end latency of NeVo is ~120 ms, 21.1% lower than ReRF (~152 ms). Since NeVo reprojects content on the fly, the motion-to-photon latency (*i.e.*, the duration between a change in the viewport and the update of the rendered frame) is kept under 33 ms.

Compared with ReRF, NeVo improves QoE by introducing content reconstruction and delivering selected ray-marching sampled points for reprojection. The incorporated operations are lightweight, with <5 ms for content reconstruction, <3 ms for contour identification, and <5 ms for reprojecting the points. Although NeVo requires more operations, transmission ordering (§4) can reduce the latency by allowing simultaneous computation and content delivery.

5.6 Evaluation of Full-fledged NeVo

We compare the full-fledged NeVo with ReRF [84], ViVo [35], and Vues [51] using the 8i [1] and V-SENSE [59] datasets. We configure Vues [51] and ViVo [35] to run at 30 FPS, following their original design. After extensive engineering optimizations, ReRF [84] achieves around 20 FPS streaming. With fewer voxels to process, NeVo is capable of achieving 30 FPS streaming. The highest point density for ViVo is limited by the decoding capability of HoloLens 2 at 30 FPS, about 200K points per frame [91]. Vues leverages an edge server for rendering point clouds and thus can stream high-quality videos. In line with the evaluation presented in Vues, we set its point density to 456K. When comparing the visual quality, we show the SSIM of the content generated using these

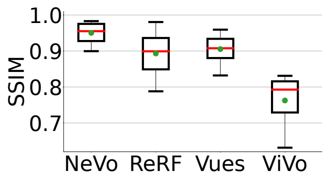


Figure 18: Visual quality of NeVo, **ReRF, Vues [51], and ViVo [35].**

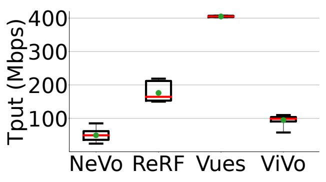


Figure 19: Streaming throughput of volumetric content with NeVo, **ReRF, Vues, and ViVo.**

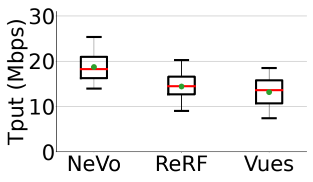


Figure 20: Streaming throughput of remote-rendered frames with NeVo, **ReRF, and Vues.**

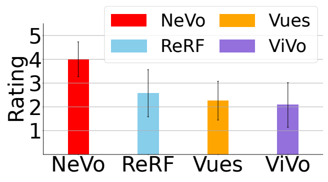


Figure 21: User rating scores for streaming with NeVo, **ReRF, Vues, and ViVo.**

systems with the ground truth rendered from the highest-quality point cloud provided by 8i and V-SENSE datasets (*e.g.,* ~1M points per frame for the *soldier* video).

Visual Quality. Figure 18 compares the SSIM of the content rendered by NeVo, ReRF, ViVo, and Vues under the network with fluctuating/limited bandwidth. The average SSIM of NeVo is 0.946, indicating good visual quality [26], and outperforms ReRF, Vues, and ViVo by 0.053, 0.039, and 0.253, respectively. Additionally, the 5th percentile SSIM for NeVo stands at 0.9 (good quality), whereas those for other systems are <0.86 (poor quality [26]). The superior visual quality of NeVo can be attributed to NeRF's ability to render high-fidelity content (compared with ViVo and Vues) and NeVo's content recovery under packet loss and reprojection (compared with ReRF), which drastically reduces rendering artifacts. In contrast, ReRF suffers from artifacts caused by packet loss and missing pixels during reprojection, Vues experiences content drift due to inaccurate viewport predictions, and ViVo is limited to streaming ~200K points per frame and reduces point density when the bandwidth drops.

Bandwidth Consumption. We first evaluate the streaming of volumetric content between the content server and the edge/client, such as delivering NeRF content in ReRF and NeVo, and point clouds in ViVo and Vues. As shown in Figure 19, NeVo consumes 49±17 Mbps bandwidth, which is 72.2% lower than ReRF (176±28 Mbps) and 51.6% lower than ViVo (95±15 Mbps). Vues does not optimize the content server to edge streaming, consuming >400 Mbps bandwidth, which is >8× higher than NeVo.

Next, we compare NeVo with Vues and ReRF for the streaming of remotely rendered content. We omit the comparison with ViVo, as it does not require an edge server to support rendering. As shown in Figure 20, the bandwidth consumption of NeVo (18.7±3.9 Mbps) is 29.9% higher than ReRF (14.4±3.7 Mbps) and 42.7% higher than Vues (13.1±3.6 Mbps). However, as shown in Figure 18, NeVo demonstrates superior visual quality with acceptable bandwidth consumption (lower than 25 Mbps for the standard broadband service required by the U.S. Federal Communications Commission (FCC) [21]).

User Experience. We evaluate the real-world user experience of NeVo, ViVo, ReRF, and Vues under fluctuating/limited bandwidth, by conducting an IRB-approved user study with 122 participants (female: 64, male: 58, average age: 31.6±10.1) recruited through the Prolific platform [64]. We ask each participant to watch eight playback groups (four groups each for *soldier* and *lubna* videos). The group is defined as (V, B, U). V contains the *soldier* and *lubna* videos; B contains the network bandwidth traces randomly selected from those introduced in §5.1; and U contains the viewport traces randomly selected from the 52 users' trajectories collected for the 8i dataset. For each group, we show the users four playbacks, with the same settings of V, B, and U. The difference is which system (NeVo, ReRF, Vues, or ViVo) generates the video. We randomly present the four video playbacks in each group. Thus, the participants do not know which system is used to create the video. For each playback, the participants provide ratings on a scale from 1 to 5 (1=bad, 2=poor, 3=fair, 4=good, 5=excellent). Each participant watches eight groups of recorded videos, leading to a total collection of 896 groups of ratings.

Figure 21 plots the scores of NeVo and the other three systems. On average, the rating for NeVo (4.0±0.7) is 53.8%, 73.9%, and 90.4% higher than ReRF (2.6±1.0), Vues (2.3±0.8), and ViVo (2.1±0.9), respectively. These results validate the high QoE of NeVo. Based on users' feedback comments for each video playback, we confirm that compared with ReRF, NeVo effectively conceals the missing pixels, thus enhancing the viewing experience. Moreover, Vues has content drifts, and ViVo has substantial holes in displayed content, both of which result in lower ratings.

5.7 Energy and Computation Utilization

Finally, we evaluate the resource utilization of NeVo on our edge server (*i.e.,* a commercially available machine introduced in §5.1) and the headset (*i.e.,* Microsoft HoloLens 2) by playing 8i videos with the full NeVo system under an unthrottled network. NeVo utilizes 20 GB of GPU memory, ~100% of CPU cycles, and ~5.3 GB of main memory on the edge server. Note that the edge is equipped with a 12-core CPU, allowing for a maximum CPU utilization of 1,200%. To measure the on-device resource utilization, we fully charge a HoloLens 2. After repeatedly playing the video for 30 minutes, the battery level drops to 82%. The average CPU/GPU utilization is ~60%/~80%, and the highest memory consumption is ~900 MB. The temperature of HoloLens 2 remains cool after the video playing. Overall, we believe the resource and energy consumption of NeVo are acceptable.

6 Discussion

Multi-user Scenarios. Simultaneously delivering volumetric content to multiple users with a single edge server is challenging [100]. As users may have different views, a trivial solution that renders NeRF content for each viewport may lead to scalability issues in terms of bandwidth consumption and compute resource utilization. A potential solution is to transmit the same content to multiple users and subsequently utilize reprojection (§3.4) to adjust it for each viewport. Nevertheless, employing this strategy in multi-user scenarios may be suboptimal due to the potential large translational differences between different users' viewports.

Live Streaming of NeRF content presents unique challenges, primarily due to the necessity for online training of NeRF models, coupled with substantial bandwidth requirements [19]. The high-level idea of NeVo can be extended to live streaming, for example, to reduce bandwidth consumption with visibility-aware optimizations (§3.2). Additionally, to expedite learning and rendering processes, an effective strategy could involve pre-training the model for the initial scene, followed by fine-tuning with only the altered content [19].

Streaming 3DGS-based content. Future systems can leverage 3DGS as the volumetric content representation, which may reduce the reliance on an edge server due to its lower computational demands. However, 3DGS typically incurs higher bandwidth requirements compared with NeRF, presenting a significant challenge in bandwidth-constrained environments. One promising solution is foveated streaming, which delivers higher quality content for only the viewer's foveal area. However, different from the foveated streaming of point clouds [91], 3DGS consists of discrete points with varying influence areas, making the generation of foveated content more complex. For example, reducing the quality of 3DGS content in the peripheral area (*e.g.,* decreasing the number of Gaussian points [72]) may affect the perceived quality within the foveation region when the influence area overlaps with this region. Thus, foveated streaming of 3DGS-based videos necessitates novel algorithms.

7 Related Work

Volumetric Video Streaming. Existing studies [35, 41, 43, 51, 52, 99, 100] primarily focused on addressing the high computational and bandwidth requirements inherent to volumetric video streaming. To achieve this goal, early work utilized visibility-aware optimizations (*e.g.,* ViVo [35]) and sped up point-cloud decompression via GPU acceleration (*e.g.,* GROOT [43]). More recent work leveraged mmWave for multi-user volumetric content delivery (*e.g.,* M5 [100]) and improved the practicality of live volumetric video streaming (*e.g.,* MetaStream [32] and MagicStream [20]). Different from these efforts on point-cloud-based systems, we focus on neural content representations such as NeRF for volumetric video streaming, which provides better visual quality.

Reliable Video Streaming. Current research [23, 45, 55] on reliable content delivery focused on addressing the impact of packet loss on 2D video streaming. For example, Hairpin [55] jointly optimizes retransmission and FEC's redundancy level to balance bandwidth consumption and latency. Grace [23] jointly trains the neural encoder and decoder under simulated packet losses, enabling robust content decoding despite poor network conditions. Reparo [45] leverages deep-learning models to reconstruct lost content at the receiver side. In this paper, we reconstruct neural content, which is more challenging and computation-intensive than 2D video.

Neural Radiance Fields. Existing research on NeRF mainly explores efficiency optimizations through techniques such as sparse geometric representations [18, 31, 97], voxel compression [57], voxel decomposition [12], and multi-modal compression [15]. Beyond that, recent research centers on the extension of NeRF to represent dynamic scenes [30, 60, 61, 71, 73, 85]. Additional efforts enhance the generalizability of NeRF [13, 38, 81, 86, 98] by reducing its dependence on per-scene training and densely captured images. In this paper, we design methods for streaming NeRF-based volumetric content from networking and systems perspectives.

8 Conclusion

In this paper, we presented the design, implementation, and evaluation of NeVo, a next-generation volumetric video streaming system with neural content representations. NeVo stands out in its ability to efficiently reduce bandwidth usage of NeRF content without a noticeable impact on visual quality. For networks experiencing packet loss, it maintains a high QoE via a learning-based content recovery model. Moreover, NeVo effectively mitigates rendering artifacts in content reprojection, improving visual fidelity. Our extensive performance evaluations indicate that NeVo significantly outperforms the state-of-the-art. We hope our initial attempts in NeVo can stimulate novel applications that benefit from photo-realistic neural immersive content.

Acknowledgment

We thank the anonymous reviewers and our shepherd Y. Charlie Hu for their insightful feedback. This work was partially supported by the National Science Foundation under Grants CNS-2402992, CNS-2212296, CCF-2217144, CNS-2106592, OAC-2126246, and CNS-1900875.

References

[1] 2017. 8i Voxelized Full Bodies (8iVFB v2) - Dynamic Voxelized Point Cloud Dataset. http://plenodb.jpeg.org/pc/8ilabs. [accessed on 03/20/2025].

[2] 2019. Microsoft HoloLens 2. https://www.microsoft.com/en-us/hololens. [accessed on 03/20/2025].

[3] 2020. PSA: WebRTC M88 Release Notes. https://groups.google.com/g/discuss-webrtc/c/A0FjOcTW2c0/m/UAv-veyPCAAJ. [accessed on 03/20/2025].

[4] 2022. Late Stage Reprojection. https://learn.microsoft.com/en-us/azure/remote-rendering/overview/features/late-stage-reprojection. [accessed on 03/20/2025].

[5] 2023. Edge Detection. https://en.wikipedia.org/wiki/Edge_detection. [accessed on 03/20/2025].

[6] 2023. tc(8) - Linux man page. https://linux.die.net/man/8/tc. [accessed on 03/20/2025].

[7] 2023. Unreal Engine. https://www.unrealengine.com. [accessed on 03/20/2025].

[8] 2024. aiortc. https://github.com/aiortc/aiortc. [accessed on 03/20/2025].

[9] 2024. Parallax. https://en.wikipedia.org/wiki/Parallax. [accessed on 03/20/2025].

[10] 2024. Preferred Walking Speed. https://en.wikipedia.org/wiki/Preferred_walking_speed. [accessed on 03/20/2025].

[11] Lukas Ahrenberg, Philip Benzie, Marcus Magnor, and John Watson. 2008. Computer Generated Holograms from Three Dimensional Meshes using an Analytic Light Transport Modell. *Applied Optics* 47, 10 (2008), 1567–1574. https://doi.org/10.1364/AO.47.001567

[12] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022. Tensorf: Tensorial Radiance Fields. In *Proceedings of European Conference on Computer Vision (ECCV)*.

[13] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. 2021. MVSNeRF: Fast Generalizable Radiance Field Reconstruction from Multi-View Stereo. In *Proceedings of the IEEE/CVF ICCV*.

[14] Bo Chen, Mingyuan Wu, Hongpeng Guo, Zhisheng Yan, and Klara Nahrstedt. 2024. Vesper: Learning to Manage Uncertainty in Video Streaming. In *Proceedings of the 15th ACM Multimedia Systems Conference*. 166–177.

[15] Bo Chen, Zhisheng Yan, Bo Han, and Klara Nahrstedt. 2024. Nerfhub: A context-aware nerf serving framework for mobile immersive applications. In *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services*. 85–98.

[16] Rick H-Y Chen and Timothy D Wilkinson. 2009. Computer Generated Hologram from Point Cloud using Graphics Processor. *Applied Optics* 48, 6 (2009), 6841–6850. https://doi.org/10.1364/AO.48.006841

[17] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. 2019. Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer. *Advances in neural information processing systems (NIPS)* (2019).

[18] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. 2023. Mobilenerf: Exploiting the Polygon Rasterization Pipeline for Efficient Neural Field Rendering on Mobile Architectures. In *Proceedings of the IEEE/CVF CVPR*.

[19] Ruizhi Cheng, Kaiyan Liu, Nan Wu, and Bo Han. 2023. Enriching Telepresence with Semantic-driven Holographic Communication. In *Proceedings of ACM Workshop on Hot Topics in Networks (HotNets)*.

[20] Ruizhi Cheng, Nan Wu, Vu Le, Eugene Chai, Matteo Varvello, and Bo Han. 2024. MagicStream: Bandwidth-conserving Immersive Telepresence via Semantic Communication. In *Proceedings of ACM SenSys*.

[21] Ruizhi Cheng, Nan Wu, Matteo Varvello, Songqing Chen, and Bo Han. 2022. Are We Ready for Metaverse? A Measurement Study of Social Virtual Reality Platforms. In *Proceedings of ACM IMC*.

[22] Wei Cheng, Ruixiang Chen, Siming Fan, Wanqi Yin, Keyu Chen, Zhongang Cai, Jingbo Wang, Yang Gao, Zhengming Yu, Zhengyu Lin, et al. 2023. DNA-Rendering: A Diverse Neural Actor Repository for High-Fidelity Human-Centric Rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

[23] Yihua Cheng, Ziyi Zhang, Hanchen Li, Anton Arapin, Yue Zhang, Qizheng Zhang, Yuhan Liu, Kuntai Du, Xu Zhang, Francis Y. Yan, Amrita Mazumdar, Nick Feamster, and Junchen Jiang. 2024. GRACE: Loss-Resilient Real-Time Video through Neural Codecs. In *Proceedings of USENIX NSDI*. https://www.usenix.org/conference/nsdi24/presentation/cheng

[24] Alexander Clemm, Maria Torres Vega, Hemanth Kumar Ravuri, Tim Wauters, and Filip De Turck. 2020. Toward Truly Immersive Holographic-type Communication: Challenges and Solutions. *IEEE Communications Magazine* 58, 1 (2020), 93–99. https://doi.org/10.1109/MCOM.001.1900272

[25] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. 2015. High-Quality Streamable Free-Viewpoint Video. *ACM Transactions on Graphics (ToG)* 34, 4 (2015), 1–13.

[26] Eduardo Cuervo, Alec Wolman, Landon P. Cox, Kiron Lebeck, Ali Razeen, Stefan Saroiu, and Madanlal Musuvathi. 2015. Kahawai: High-Quality Mobile Gaming Using GPU Offload. In *Proceedings of ACM MobiSys*.

[27] The Khang Dang, Nitinder Mohan, Lorenzo Corneo, Aleksandr Zavodovski, Jörg Ott, and Jussi Kangasharju. 2021. Cloudy with a Chance of Short RTTs: Analyzing Cloud Connectivity in the Internet. In *Proceedings of ACM Internet Measurement Conference (IMC)*.

[28] Leila De Floriani, Franco Morando, and Enrico Puppo. 2003. Representation of non-manifold objects. In *Proceedings of ACM Symposium on Solid Modeling and Applications*. https://doi.org/10.1145/781606.781656

[29] Frederik Michel Dekking, Cornelis Kraaikamp, Hendrik Paul Lopuhaä, and Ludolf Erwin Meester. 2005. *A Modern Introduction to Probability and Statistics: Understanding why and how*. Springer.

[30] 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 *Proceedings of the IEEE/CVF ICCV*.

[31] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2022. Plenoxels: Radiance Fields without Neural Networks. In *Proceedings of the IEEE/CVF CVPR*.

[32] Yongjie Guan, Xueyu Hou, Nan Wu, Bo Han, and Tao Han. 2023. MetaStream: Live Volumetric Content Capture, Creation, Delivery, and Rendering in Real Time. In *Proceedings of ACM MobiCom*.

[33] Serhan Gül, Dimitri Podborski, Jangwoo Son, Gurdeep Singh Bhullar, Thomas Buchholz, Thomas Schierl, and Cornelius Hellge. 2020. Cloud rendering-based volumetric video streaming system for mixed reality services. In *Proceedings of ACM MMSys*. https://doi.org/10.1145/3339825.3393583

[34] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. 2011. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM review* 53, 2 (2011), 217–288.

[35] Bo Han, Yu Liu, and Feng Qian. 2020. ViVo: Visibility-aware Mobile Volumetric Video Streaming. In *Proceedings of ACM MobiCom*. https://doi.org/10.1145/3372224.3380888

[36] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[37] Christer Holmberg, Stefan Hakansson, and Goran Eriksson. 2015. Web Real-Time Communication Use Cases and Requirements. RFC 7478. https://rfc-editor.org/rfc/rfc7478.txt

[38] Mohammad Mahdi Johari, Yann Lepoittevin, and François Fleuret. 2022. GeoNeRF: Generalizing NeRF with Geometry Priors. In *Proceedings of the IEEE/CVF CVPR*.

[39] 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), 1–14.

[40] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *In Proceedings of International Conference on Learning Representations (ICLR)*.

[41] Juheon Yi Kyungjin Lee and Youngki Lee. 2023. FarfetchFusion: Towards Fully Mobile Live 3D Telepresence Platform. In *Proceedings of ACM MobiCom*.

[42] Kyungmin Lee, David Chu, Eduardo Cuervo, Johannes Kopf, Yury Degtyarev, Sergey Grizan, Alec Wolman, and Jason Flinn. 2015. Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming. In *Proceedings of ACM MobiSys*.

[43] Kyungjin Lee, Juheon Yi, Youngki Lee, Sunghyun Choi, and Young Min Kim. 2020. GROOT: a real-time streaming system of high-fidelity volumetric videos. In *Proceedings of ACM MobiCom*. https://doi.org/10.1145/3372224.3419214

[44] MC Lee, Raymond KW Chan, and Donald A Adjeroh. 1997. Quantization of 3D-DCT Coefficients and Scan Order for Video Compression. *Journal of Visual Communication and Image Representation* 8, 4 (1997), 405–422.

[45] Tianhong Li, Vibhaalakshmi Sivaraman, Lijie Fan, Mohammad Alizadeh, and Dina Katabi. 2023. Reparo: Loss-resilient Generative Codec for Video Conferencing. https://arxiv.org/abs/2305.14135. [accessed on 03/20/2025].

[46] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. 2022. Neural 3D Video Synthesis from Multi-view Video. In *Proceedings of IEEE/CVF CVPR*.

[47] Xueting Li, Shalini De Mello, Sifei Liu, Koki Nagano, Umar Iqbal, and Jan Kautz. 2024. Generalizable One-shot 3D Neural Head Avatar. *Advances in Neural Information Processing Systems* 36 (2024).

[48] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. 2024. Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[49] Junhua Liu, Boxiang Zhu, Fangxin Wang, Yili Jin, Wenyi Zhang, Zihan Xu, and Shuguang Cui. 2023. CaV3: Cache-assisted Viewport Adaptive Volumetric Video Streaming. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*.

[50] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. 2019. Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. In *Proceedings of the IEEE/CVF CVPR*.

[51] Yu Liu, Bo Han, Feng Qian, Arvind Narayanan, and Zhi-Li Zhang. 2022. Vues: practical mobile volumetric video streaming through multiview transcoding. In *Proceedings of ACM MobiCom*. https://doi.org/10.1145/3495243.3517027

[52] Yu Liu, Puqi Zhou, Zejun Zhang, Anlan Zhang, Bo Han, Zhenhua Li, and Feng Qian. 2024. MuV^2: Scaling up Multi-user Mobile Volumetric Video Streaming via Content Hybridization and Sharing. In *Proceedings of ACM MobiCom*.

[53] Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. 2021. Mixture of Volumetric Primitives for Efficient Neural Rendering. *ACM Transactions on Graphics (ToG)* 40, 4 (2021), 1–13.

[54] Michael Mathieu, Camille Couprie, and Yann LeCun. 2015. Deep Multi-scale Video Prediction beyond Mean Square Error. https://arxiv.org/abs/1511.05440. [accessed on 03/20/2025].

[55] Zili Meng, Xiao Kong, Jing Chen, Bo Wang, Mingwei Xu, Rui Han, Honghao Liu, Venkat Arun, Hongxin Hu, and Xue Wei. 2024. Hairpin: Rethinking Packet Loss Recovery in Edge-based Interactive Video Streaming. In *Proceedings of USENIX NSDI*. https://www.usenix.org/conference/nsdi24/presentation/meng

[56] 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 (2021), 99–106.

[57] 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), 1–15.

[58] Richard A Newcombe, Dieter Fox, and Steven M Seitz. 2015. Dynamicfusion: Reconstruction and Tracking of Non-rigid Scenes in Real-time. In *Proceedings of the IEEE/CVF CVPR*. 343–352.

[59] Rafael Pagés, Konstantinos Amplianitis, Jan Ondrej, Emin Zerman, and Aljosa Smolic. 2022. Volograms & V-SENSE Volumetric Video Dataset. (2022). https://doi.org/10.13140/RG.2.2.24235.31529/1

[60] 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 *Proceedings of the IEEE/CVF CVPR*.

[61] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. 2021. HyperNeRF: a Higher-dimensional Representation for Topologically Varying Neural Radiance Fields. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–12.

[62] Jingliang Peng, Chang-Su Kim, and C-C Jay Kuo. 2005. Technologies for 3D Mesh Compression: A Survey. *Journal of visual communication and image representation* 16, 6 (2005), 688–733.

[63] Sida Peng, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. 2023. Representing Volumetric Videos as Dynamic MLP Maps. In *Proceedings of IEEE/CVF CVPR*.

[64] Prolific. 2024. Easily Find Vetted Research Participants and AI Taskers at Scale. https://www.prolific.com/. [accessed on 03/20/2025].

[65] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2021. D-nerf: Neural Radiance Fields for Dynamic Scenes. In *Proceedings of IEEE/CVF CVPR*.

[66] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of IEEE/CVF CVPR*.

[67] Feng Qian, Bo Han, Qingyang Xiao, and Vijay Gopalakrishnan. 2018. Flare: Practical Viewport-Adaptive 360-Degree Video Streaming for Mobile Devices. In *Proceedings of ACM MobiCom*.

[68] Michael Rudow, Francis Y Yan, Abhishek Kumar, Ganesh Ananthanarayanan, Martin Ellis, and KV Rashmi. 2023. Tambur: Efficient Loss Recovery for Videoconferencing via Streaming Codes. In *Proceedings of USENIX NSDI*.

[69] Radu Bogdan Rusu and Steve Cousins. 2011. 3D is here: Point Cloud Library (PCL). In *2011 IEEE international conference on robotics and automation*. 1–4.

[70] William Sentosa, Balakrishnan Chandrasekaran, P. Brighten Godfrey, Haitham Hassanieh, and Bruce Maggs. 2023. DChannel: Accelerating Mobile Applications With Parallel High-bandwidth and Low-latency Channels. In *Proceedings of USENIX NSDI*. https://www.usenix.org/conference/nsdi23/presentation/sentosa

[71] Ruizhi Shao, Zerong Zheng, Hanzhang Tu, Boning Liu, Hongwen Zhang, and Yebin Liu. 2023. Tensor4D: Efficient Neural 4D Decomposition for High-fidelity Dynamic Reconstruction and Rendering.

In *Proceedings of the IEEE/CVF CVPR.*

[72] Yuang Shi, Simone Gasparini, Géraldine Morin, and Wei Tsang Ooi. 2025. LapisGS: Layered Progressive 3D Gaussian Splatting for Adaptive Streaming. In *Proceedings of International Conference on 3D Vision.*

[73] 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. *IEEE Transactions on Visualization and Computer Graphics* 29, 5 (2023), 2732–2742.

[74] Emilio Calvanese Strinati, Sergio Barbarossa, Jose Luis Gonzalez-Jimenez, Dimitri Ktenas, Nicolas Cassiau, Luc Maret, and Cedric Dehos. 2019. 6G: The Next Frontier: From Holographic Messaging to Artificial Intelligence Using Subterahertz and Visible Light Communication. *IEEE Vehicular Technology Magazine* 14, 3 (2019), 42–50. https://doi.org/10.1109/MVT.2019.2921162

[75] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. 2018. Splatnet: Sparse Lattice Networks for Point Cloud Processing. In *Proceedings of the IEEE/CVF CVPR.*

[76] Shih-Yang Su, Frank Yu, Michael Zollhöfer, and Helge Rhodin. 2021. A-NeRF: Articulated Neural Radiance Fields for Learning Human Shape, Appearance, and Pose. *Proceedings of Advances in Neural Information Processing Systems (NIPS)* (2021).

[77] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. 2012. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Transactions on circuits and systems for video technology* 22, 12 (2012), 1649–1668.

[78] Cheng Sun, Min Sun, and Hwann-Tzong Chen. 2022. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.*

[79] Towaki Takikawa, Alex Evans, Jonathan Tremblay, Thomas Müller, Morgan McGuire, Alec Jacobson, and Sanja Fidler. 2022. Variable Bitrate Neural Fields. In *Proceedings of the ACM SIGGRAPH.*

[80] Faisal Tariq, Muhammad RA Khandaker, Kai-Kit Wong, Muhammad A Imran, Mehdi Bennis, and Merouane Debbah. 2020. A Speculative Study on 6G. *IEEE Wireless Communications* 27, 4 (2020), 118–125. https://doi.org/10.1109/MWC.001.1900488

[81] Alex Trevithick and Bo Yang. 2021. GRF: Learning a General Radiance Field for 3D Representation and Rendering. In *Proceedings of the IEEE/CVF ICCV.*

[82] Athena Vakali and George Pallis. 2003. Content Delivery Networks: Status and Trends. *IEEE Internet Computing* 7, 6 (2003), 68–74.

[83] Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matt J Kusner. 2021. Unsupervised Point Cloud Pre-training via Occlusion Completion. In *Proceedings of the IEEE/CVF ICCV.*

[84] Liao Wang, Qiang Hu, Qihan He, Ziyu Wang, Jingyi Yu, Tinne Tuytelaars, Lan Xu, and Minye Wu. 2023. Neural Residual Radiance Fields for Streamably Free-Viewpoint Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.*

[85] Liao Wang, Jiakai Zhang, Xinhang Liu, Fuqiang Zhao, Yanshun Zhang, Yingliang Zhang, Minve Wu, Jingyi Yu, and Lan Xu. 2022. Fourier PlenOctrees for Dynamic Radiance Field Rendering in Real-time. In *Proceedings of the IEEE/CVF CVPR.*

[86] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. 2021. IBRNet: Learning Multi-View Image-Based Rendering. In *Proceedings of the IEEE/CVF CVPR.*

[87] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image Quality Assessment: from Error Visibility to Structural Similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

[88] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. 2003. Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions on circuits and systems for video technology* 13, 7 (2003), 560–576.

[89] 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.*

[90] Minye Wu, Yuehao Wang, Qiang Hu, and Jingyi Yu. 2020. Multi-view Neural Human Rendering. In *Proceedings of the IEEE/CVF CVPR.*

[91] Nan Wu, Kaiyan Liu, Ruizhi Cheng, Bo Han, and Puqi Zhou. 2024. Theia: Gaze-driven and Perception-aware Volumetric Content Delivery for Mixed Reality Headsets. In *Proceedings of ACM MobiSys.*

[92] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. 2021. Space-time Neural Irradiance Fields for Free-viewpoint Video. In *Proceedings of IEEE/CVF CVPR.*

[93] Chongyang Xiang, Jiajun Xu, Chuan Yan, Qiang Peng, and Xiao Wu. 2019. Generative Adversarial Networks Based Error Concealment for Low Resolution Video. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).*

[94] Jun Xiang, Xuan Gao, Yudong Guo, and Juyong Zhang. 2024. FlashAvatar: High-fidelity Head Avatar with Efficient Gaussian Embedding. In *Proceedings of the IEEE/CVF CVPR.*

[95] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. 2022. Point-NeRF: Point-based Neural Radiance Fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*

[96] Tan Xu, Bo Han, and Feng Qian. 2019. Analyzing viewport prediction under different VR interactions. In *Proceedings of ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT).* https://doi.org/10.1145/3359989.3365413

[97] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. 2021. Plenoctrees for Real-time Rendering of Neural Radiance Fields. In *Proceedings of the IEEE/CVF ICCV.*

[98] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. 2021. pixelNeRF: Neural Radiance Fields from One or Few Images. In *Proceedings of IEEE/CVF CVPR.*

[99] Anlan Zhang, Chendong Wang, Bo Han, and Feng Qian. 2022. YuZu: Neural-enhanced Volumetric Video Streaming. In *Proceedings of USENIX NSDI.* https://www.usenix.org/conference/nsdi22/presentation/zhang-anlan

[100] Ding Zhang, Puqi Zhou, Bo Han, and Parth Pathak. 2022. M5: Facilitating Multi-User Volumetric Content Delivery with Multi-Lobe Multicast over mmWave. In *Proceedings of ACM SenSys.* https://doi.org/10.1145/3560905.3568540

[101] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition.*