Mobile Volumetric Video Streaming System through Implicit Neural Representation

Junhua Liu[1,2], Yuanyuan Wang[2], Yan Wang[4,5], Yufeng Wang[5], Shuguang Cui[3,1], Fangxin Wang[3,1]*
[1]FNii, CUHK-Shenzhen, [2]Sensetime Research, [3]SSE, CUHK-Shenzhen,
[4]Institute for AI Industry Research (AIR), Tsinghua University, [5]Tsinghua University.

ABSTRACT

Volumetric video (VV) emerges as a new video paradigm with six degree-of-freedom (DoF) immersive viewing experience. Most existing VV systems focus on the point cloud (PtCl)-based architecture, which is however far from effective due to the huge video size, unrealistic color variations, and specialized player platform requirement. The recent advance of implicit neural representations (INR) such as NeRF brings great opportunities to VV given its potential in creating photorealistic 3D appearances and lighting consistency. However, there still exist arduous challenges in many aspects such as model training, display rendering, streaming optimization, and system implementation. To address the above challenges, we develop `NeRVo`, an INR-based VV representation for mobile VV. `NeRVo` improves the training and rendering speed over 300x and 1000x with photorealism, mobile compatibility, and desirable datarates compared to NeRF. We adopt `NeRVo` as a building block, design and implement a holistic INR-enhanced VV streaming system `VoINR`.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; **Mixed / augmented reality**.

KEYWORDS

Volumetric Video Streaming, Implicit Neural Representation, Mobile Mixed Reality

ACM Reference Format:
Junhua Liu, Yuanyuan Wang, Yan Wang, Yufeng Wang, Shuguang Cui, Fangxin Wang. 2023. Mobile Volumetric Video Streaming System through Implicit Neural Representation. In *Workshop on Emerging Multimedia Systems (EMS '23), September 10, 2023, New York, NY, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3609395.3610593

1 INTRODUCTION

Volumetric video (VV) provides a foundation for mixed reality technologies. In contrast to traditional 2D videos, every frame of VV consists of a 3D scene, which surpasses traditional 2D video and 360° video [16] in terms of *photorealism* and *interactivity*. While volumetric primitives have been extensively studied in the computer graphics community, research from the perspective of VV streaming is still in the infant stage. In effect, streaming interactive high-quality VV on mobile networks and devices with consistently low latency is quite challenging. Several mobile VV systems have been proposed; however, they only focus on PtCl-based architecture and none have achieved the same level of *photorealism*, *cost-efficiency*, and *convenience* as seen in current commercial 2D video streaming platforms (e.g., YouTube or Netflix).

Recently, implicit neural representations (INR) have caused a paradigm shift in 3D scene representation, known as NeRF [23], a neural network that leverages multi-view RGB images to generate highly detailed and realistic 3D structures. The quality of NeRF surpasses scanned PtCl. Despite recent attention paid to INR in the machine learning and graphics communities, its potential applications to practical systems remain unexplored. Therefore, we employ a different approach to improve the QoE of VV streaming using INR (specifically NeRF-based INR) to represent every frame. However, building an INR-enhanced mobile VV system is unique and challenging for the following reasons:

• In each video, there are thousands of frames to process. Utilizing NeRF-based methods on such a high volume of data presents a formidable challenge due to the significant training costs and time required, a consequence of the original NeRF's computationally and bandwidth-demanding nature. To load, stream and render high-quality 3D content at an elevated frame rate would poses an additional, substantial challenge.

• The inherent characteristics of neural networks render the streaming process via NeRF fundamentally different from PtCl. This distinction results in the incompatibility of existing streaming methodologies and techniques with NeRF. Considering NeRF's recent introduction, there is a notable scarcity of research addressing fundamental infrastructures, including tools, models, and methods.

• Given the magnitude of big data streams, the implementation of commonly used methods such as compression, bitrate adaptation, and viewport adaptation on mobile devices, as well as system-level integration, presents a significant hurdle.

• Creating a unified platform that can support a myriad of devices and operating systems presents a significant challenge, one that current PtCl-based systems have not yet been able to overcome. Devices such as VR headsets and glasses necessitate stereoscopic, egocentric, and interactive frame rates. However, a majority of these devices are characterized by low throughput and computation. This deficiency can adversely affect the overall user interaction experience, potentially leading to discomfort or even inducing sickness.

*Fangxin Wang is the corresponding author.

EMS '23, September 10, 2023, New York, NY, USA

ACM ISBN 979-8-4007-0303-4/23/09. . . $15.00
https://doi.org/10.1145/3609395.3610593

Scheme	Ref	Advantages (⊕) and Disadvantage (⊖)
Direct Streaming (DS)	[27]	⊕ Easy to implement, best QoE (if bandwidth is sufficient). ⊖ Highest network bandwidth (BW) usage.
DS + VA	[13, 17]	⊕ Lower BW usage. ⊖ BW saving depends on user's motion, QoE depends on motion prediction.
DS + SR	[32, 33]	⊕ Good QoE, further lower BW usage, adaptively trades compute resource for BW ⊖ Requires training.
Transcoding	[19]	⊕ Lowest BW usage.⊖ QoE depends on motion prediction, need edge support (poor scalability).
DS + INR + VA	Melt	⊕ Best QoE and immersion, lower BW consumption, all-platform ⊖ QoE depends on motion prediction

Table 1: Comparision - Four categories of volumetric video streaming approaches (VA = Viewport Adaptation; SR = Super Resolution)

To mitigate the challenges and shortcomings discussed in Section 2, we introduce `NeRVo`, a mobile-friendly NeRF, for the first time. This novel design offers several enhancements, overcoming the hurdles typically encountered in commercial VV streaming. Following this, we design, implement, and evaluate `VoINR`, a practical end-to-end system that uses `NeRVo` as its foundation. `VoINR` facilitates and optimizes an INR-enhanced streaming pipeline, tailored specifically for a range of mobile devices and various network conditions. Our study encompasses:

NeRVo for Mobile VV (§3). `NeRVo`'s design is a careful trade-off between various targets, including quality, time, cost, size, and mobile compatibility. Existing work tends to optimize a particular target at the expense of others' resources. Such approaches are specific to certain applications. `NeRVo` optimizes by redesigning architecture, training, and rendering of NeRF while incorporating effective encoding techniques.

INR-enhanced VV Streaming (§4). `VoINR` deeply optimizes the end-to-end streaming pipeline in: content capture, loading, segmentation, compression, and viewport prediction. It then performs two-stage adaptation and hybrid encoding methods to adapt to network and computing resources. `VoINR` utilizes the standard rendering and conducts the system-level integration into a web interface, which enables the mobile VV streaming.

Implementation and Evaluation (§5). We implement `VoINR` as a practical end-to-end system using 46.1k LoC for training and streaming, and 2.3k LoC for the Web player on the client side. The player is cross-platform and runs in real-time on off-the-shelf mobile devices. Our preliminary results over WiFi, live LTE, low bandwidth and fluctuating network, with two INR datasets and one PtCl dataset as well as preliminary user study (§6) demonstrate its efficacy. We also discuss `VoINR` with three SOTA VV streaming approaches and PtCl-based architecture.

2 BACKGROUND AND MOTIVATION

Table 1 reveals the strengths and weaknesses of existing systems. Other flaws are summarized as:

System. None of them are particularly pragmatic. The highly discretized nature of PtCl results in huge bandwidth and overhead. The SOTA systems can save 40% data usage [13] on average. The requisite bandwidth remains substantial.

Representation. PtCl algorithm does not need complex pre-processing steps and is more widely used in VV streaming scenarios. However, existing systems seldom consider the visual artifacts that affect human perception. Low point density, rendering artifacts, and gaps or missing data in the point cloud can also reduce photorealism.

Device. The existing VV display demands considerable CPU and memory, even dedicated hardware and software. The absence of a unified framework supporting diverse devices and operating systems, akin to the video sites employed in routine usage, poses a significant challenge to the widescale deployment of commercial VV streaming.

INR. The inherent capability of INRs to represent spatiotemporal signals across arbitrary input/output dimensions makes them a versatile tool for interpreting real-world observations. These observations may include images, 3D scenes, videos, audio, or even more specialized modalities. The PtCl method, on the other hand, excels in achieving real-time capture and rendering, significantly enhancing the efficiency without compromising the quality to a large extent. Furthermore, NeRFs [23] train a MLP-based radiance and opacity field, thereby achieving SOTA quality.

Mobile Rendering. Several techniques are crucial for real-time rendering on mobile devices. For example, rasterization with z-buffering plays a vital role in achieving real-time rendering. GLSL shaders enable GPU-based rendering, while WebGL serves as the standard web-based rendering technology. These techniques are widely utilized in the industry, including popular frameworks such as Three.js, WebGPU, and web video games. Therefore, the key to enabling mobile VV streaming is to fully combine these techniques.

However, integrating NeRF into VV is not direct, practical volumetric primitives for streaming should achieve: (1) Fast and cost-effective to initiate and train. (2) Interactive rendering on a variety of mobile devices. (3) Desirable datarates for real-time streaming. The main impediments attribute to its inefficiency: (1) Ray casting-based neural models evaluating a large MLP at thousands of positions require much resource and time (2) The volume rendering is slow and not well-suited to the common hardware. (3) Existing training methods use specialized CUDA/C++ implementations and only expose low-level interfaces, which are unavailable for mobile devices. The above observations motivate us to design `NeRVo` for mobile VV streaming.

3 NERVO ARCHITECTURE

As shown in Fig. 4, NeRF involves three steps:

Step ①: Map pixels to rays: For each pixel that is to be rendered in the target novel view, a ray, denoted as $\mathbf{r} = \mathbf{o} + t\mathbf{d}$, is projected from the origin (for instance, the center of the camera) of the target novel view, represented as o. This ray travels in the direction d to pass through the specific pixel under consideration. Here, t signifies the distance between the sampled points along this ray, while the origin is represented as o.

Step ②: Query the features of points along the rays: For each point that has a distance t_k from $\mathbf{o}$, both its location $\mathbf{o} + t_k\mathbf{d}$ and direction $\mathbf{d}$ are sent as inputs to the MLP model $(\mathbf{o} + t_k\mathbf{d}, \mathbf{d}) \rightarrow (\sigma_k, \mathbf{c}_k)$, which outputs the corresponding density σ_k and an RGB color $\mathbf{c}_k$ as the extracted feature of this particular point;

Step ③: Render pixels' colors: The color $\mathrm{C}(\mathbf{r})$ of the pixel corresponding to the ray $\mathbf{r}$ can be computed by integrating the features of the points along the ray, which can be represented as:

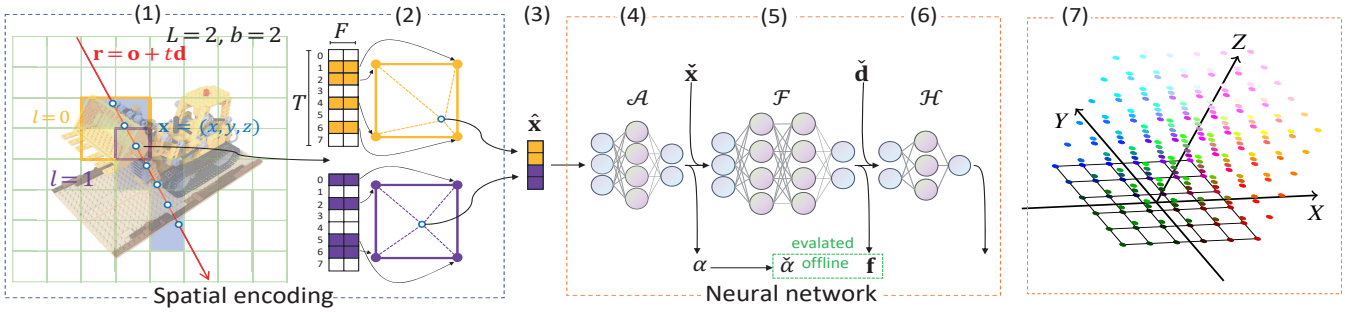

Figure 1: Illustration of NeRVo- **(1)** *Ray marching and quadrature point sampling*: For each ray $\vec{r} = \vec{o} + t\vec{d}$ we sample quadrature points $\mathcal{K} = \{t_k\}$ along it. We find the containing voxels of each position at L resolution levels. Voxels that are unlikely to have an effective opacity are pruned. **(2)** *Feature vectors lookup and interpolation*: We hash the vertices of the voxels and lookup feature vectors in the hash tables. The feature vectors are linearly interpolated according to the relative position of $\vec{x}$ within the respective l-th voxel. **(3)** The resulting feature vector of each level are *concatenated* to build the spatial encoding $\hat{\vec{x}}$. **(4)** *Network* $\mathcal{A}$ takes the spatial encoding $\hat{\vec{x}}$ and outputs $\vec{a}$, from which the continuous opacity α and the binarized opacity $\hat{\alpha}$ are calculated. **(5)** *Network* $\mathcal{F}$ takes $\vec{a}$ and anothor spatial encoding $\check{\vec{x}}$. It outputs $\vec{f}$. $\mathcal{A}$ and $\mathcal{F}$ are view-independent so α, $\hat{\alpha}$, and $\vec{f}$ can be evaluated offline. **(6)** *Network* $\mathcal{H}$ takes $\vec{f}$ and sinusoidal-encoded view direction $\check{\vec{d}}$. It outputs radiance $\vec{h}$. $\vec{h}$ combined with α (or $\hat{\alpha}$) is then rendered to pixel color. (7) A sample 3D visualization of feature vectors of a specific level.

$$\mathbf{C}(\mathbf{r}) = \sum_{k=1}^{N} T_k \left(1 - \exp\left(-\sigma_k \left(t_{k+1} - t_k\right)\right)\right) \mathbf{c}k, \quad T_k = \exp\left(-\sum_{j=1}^{k} \sigma_j \left(t_{j+1} - t_j\right)\right). \tag{1}$$

In further, we divide the training of MLP into three sub-parts.

- **Regular Training**: We train a NeRF-like model with successive opacity, where volume rendering quadrature points are derived from the polygonal mesh;
- **Bitmap encoding**: We binarize the opacities, as while classical rasterization can easily discard fragments, they cannot elegantly deal with semi-transparent fragments
- **Fine-tuning and Extracting**: We extract a sparse polygonal mesh, bake the opacities and features into texture maps, and store the weights of the neural deferred shader.

Therefore, we could express a 3D scene as an implicit function whose inputs are a location $\vec{x}$ and a unit view direction $\vec{d}$, and whose outputs are a radiance $\vec{h}$ and an opacity α, given a collection of photos with poses. Our model can be described as follows:

$$\vec{a} = \mathcal{A}\left(\hat{\vec{x}}; \Phi_{\mathcal{A}}\right), \ \vec{f} = \mathcal{F}\left(\vec{a}, \check{\vec{x}}; \Phi_{\mathcal{F}}\right), \ \vec{h} = \mathcal{H}\left(\vec{f}, \check{\vec{d}}; \Phi_{\mathcal{H}}\right) \tag{2}$$

where $\hat{\vec{x}}$ and $\check{\vec{x}}$ are two hash encodings of $\vec{x}$ called siamese neural network. $\check{\vec{d}}$ is the view direction $\vec{d}$ projected onto the spherical harmonics basis, as in the original NeRF. $\mathcal{A}$, $\mathcal{F}$ and $\mathcal{H}$ are MLPs. In summary, compared with the original NeRF, the following four major changes were made:

- Split the model into three MLPs to support *deferred rendering*. The first two, $\mathcal{A}$ and $\mathcal{F}$, are independent with the view direction $\vec{d}$, and can be evaluated offline. The deferred network $\mathcal{H}$ is designed to be a small network and can be evaluated efficiently in view-dependent rendering.)
- Encode the location $\vec{x}$ using trainable multicascade spatial hash tables and siamese encoding to provide *efficient training* and *rapid convergence*. The regional attributes are separated from MLPs and kept in hash tables. When each quadrature point is back-propagated, only a few parameters are modified.
- Use bitmap encoding to binarize the grid opacity instead of volume density. The rendering mechanism does not rely on volumetric integrals, but depth buffering, which enables efficient rendering on consumer-grade hardware.
- Use fine-tuning to further enhance the quality. Precomputing the explicit mesh and storing voxel grid with learned feature vectors and decoding with lightweight MLP. The hybrid representation significantly accelerates the rendering and enables the feasibility of the streaming pipeline in §4.

4 SYSTEM DESIGN OF VOINR

`NeRVo` extracts the explicit mesh, where vertex attributes store the features and opacity, allowing efficient rendering, segmentation, encoding, etc. We propose a novel VV streaming system to deliver a group of contiguous polygonal meshes and lightweight deferred shaders by leveraging our `NeRVo`. We next describe its nine main components denoted as **C1** to **C9**:

C1: Video Capture and Reconstruction. RGB-D cameras with depth sensors (e.g., Microsoft Kinect) enable the acquisition of dense point clouds in real time. It presents hardware requirements and restricts the scalability to single objects, thus limiting the potential for broader applications [1, 3, 28]. `VoINR` employs multi-view RGB inputs for model training. *The cameras could be simple as mobile phones*, which has lower hardware requirements and enables anyone to create content that could be consumed immersively and the capture of diverse scenes including indoor environment and city. During the training process, video frames captured by different cameras will be properly synchronized and their coordinates be unified. Noises and outliers in the images are removed. Different video streams and corresponding camera parameters (intrinsic and extrinsic) are then uploaded to the server and trained.

C2: Bitrate Control of `NeRVo`. The voxel grid is converted into explicit polygon mesh at different resolutions for bitrate adaptation.

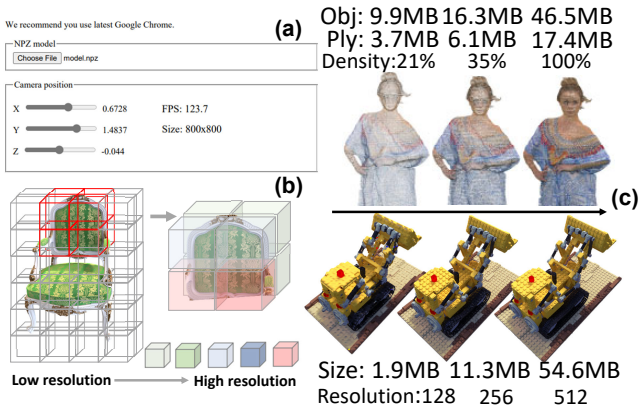

Figure 2: VoINR demo - (a) The renderer - The web demo enables the rendering. The viewer can alter their perspective by adjusting the camera poses through three buttons. (b) A demonstration of tile-based streaming - The chair is divided into several tiles, each allocated different resolutions. (c) The size and visualization under different resolutions, as well as a comparison with the PtCl dataset [1, 28].

The resolution is specified in terms of the size of the vertices and faces relative to the size of the NeRVo, which provides different levels of detail and expressiveness. In Fig. 2, compared with high-density PtCls, our model is able to *provide similar levels of detail at low resolutions, while high-resolution models offer more photorealistic and detailed visualizations,* which avoids data loss or buffering due to network congestion.

As shown in Fig. 2, the size of NeRVo is significantly less than PtCl. VoINR improves the loading time by 12.2x, 3.8x, and 31.6x than three open-source PtCl importers: Point cloud visualizer [6], StopMotion [7], and Blender [2] using a laptop.

C3: Enabling Tile-based Streaming. In tile-based streaming, PtCl is divided into tiles, streamed and rendered independently, allowing adaptation to the client's available bandwidth and computing resources. PtCls are widely used due to their simplicity. However, textured meshes are difficult to segment as they need to respect triangle and uv parameterization continuity. NeRFs are difficult to segment due to their inherent neural network structure. Therefore, our model bakes the features and opacity as vertex attributes to overcome these limitations. We pre-calculate the NeRF model as a frame-to-frame polygonal mesh rather than a topology-changing mesh and use a lightweight MLP for deferred shading to enhance realism. NeRVo is then straightforward to tile and encode. As described in Fig. 2(b), we divide the mesh into small 3D blocks with a size of $L \times L \times L$ (L=5) and adaptively allocate the bitrate.

C4: Volumetric Video Compression. Our analysis and summary of existing methods are shown in Table. 2. 2D-based method [18] is computationally intensive. VV of 511 MB will take 19 minutes to encode and 34 seconds to decode using V-PCC. VQ-based methods exhibit a strong dependency on GPU resources. 3D-based methods require low computational resources and decoding time. We evaluate the performance of three 3D-based compression solutions: Google Draco (k-d tree) [8], Point Cloud Library (Octree) [5], and LEPC [4] on mobile devices. They employ distinct algorithms. Draco achieves the best performance with compression rate over 4.5x, decoding fps over 40 fps, therefore, we use Draco for video compression.

	Model	Ref	Advantages and Disadvantages
3D	Draco	[8]	⊕ Low decoding overhead ⊖ Low compression rate
2D	V-PCC	[18]	⊕ Huge compression rate ⊖ High decoding overhead
VQ	VQRF	[30]	⊕ High compression rate. ⊖High GPU-dependence

Table 2: Comparision – Comparison of existing encoding methods. (2D=2D projection-based and 3D=3D-structure-based methods)

C5: Lightweight Viewport Prediction for Mobile Devices. As only a limited part of the VV is watched by the user, we use viewport direction model ARIMA [9]to decide the tile download selections. The predicted viewport for a chunk of frames is obtained using the actual viewports of the previous chunks, which helps to maintain the locality information and gives smooth predictions. It achieves prediction of less than 30 ms to avoid detrimental effects [26] and streaming overhead on our mobile devices.

C5: QoE Model for NeRVo. Factors affecting QoE of INR-enhanced VV include Quality, distance, invisibility, and impairment. Given a single frame i, tile j, we define quality Q_i as the average of all its visible tiles' quality values, δ_i as the viewing distances to all the tiles in frame i, we formulate the QoE model as:

$$QoE = \sum_i Q_i - \sum_i \mu_p(\delta_i) I_i^{\text{tile}} - \sum_i \mu_f(\delta_i) I_i^{\text{frame}} - \sum_i \mu_s(\delta_i) I_i^{\text{stall}} \quad (3)$$

where $v_{i,j}$ is 1 if the tile is visible, meaning it is within the viewport and not blocked by other tiles. We define the *inter-tile quality switch* I_i^{tilze} as the variation (Var(·)) in quality among the visible tiles in frame i . I_i^{frame} measures the quality switch impairment, the quality change from frame $i-1$ to i.

$$I_i^{\text{tile}} = \text{Var}(\{PSNR_{i,j} \mid \forall j, v_{i,j} > 0\}), I_i^{\text{frame}} = \|Q_i - Q_{i-1}\| \quad (4)$$

QoE model cannot measure other visual effects of NeRVo. We then conduct a user study for qualitative evaluation (§6).

The Hardness of Adaptation in Volumetric Video. VoINR adapts to fluctuating network and computing resource dispersion of different devices. The resource adaptation could be formulated into an optimization problem that finds the optimal policy to maximize the QoE. The adaptive video streaming [21] is efficient in 2D video scenarios. However, in the VV scenario, the solution space increases exponentially. Due to the large solution space, an exhaustive search is infeasible. [18] also demonstrate the considerable computational resource and communication overhead for VV.

C6: Network Adaptation. Current methodologies are ineffective at addressing the problem at hand and cannot be efficiently deployed on mobile devices. To overcome these limitations, we have developed a lightweight, two-stage adaptation method. In the first stage, prior to each chunk download, our system, VoINR, executes a coarse-grained allocation. This allocation process assumes that all tiles within a chunk are of identical quality. To manage the bitrate allocation for the upcoming video chunk, we employ the Asynchronous Advantage Actor-Critic (A3C) network [24].

Observation. The blocked and those tiles out of the viewport will not influence the QoE, viewers are more sensitive to nearby details. In the second stage, VoINR performs a fine-grained allocation. We adopt the allocated bitrate from the first stage as high quality and three visibility-aware optimizations [13] to adaptive allocation of each tile's bitrate. VoINR identifies the tiles of different conditions

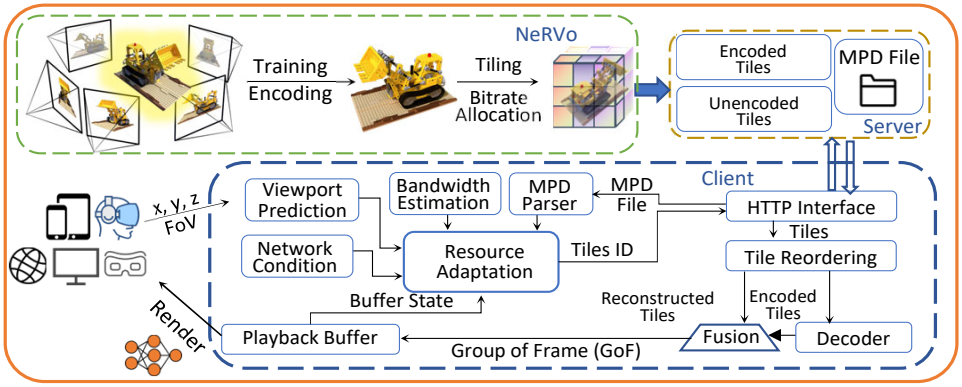

Figure 3: Illustration - Detailed Architecture of VoINR

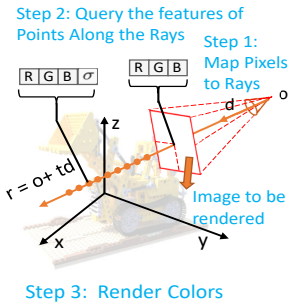

Figure 4: Preposition - Three steps of NeRF.

and assigns high-quality to tiles within the viewport, low-quality to tiles without the viewport or blocked tiles instead. The resolution is degraded when the distance is beyond 1 m, 3 m, and 5 m.

C7: Computing Resource Adaptation. Computing resource adaptation is less-well explored. We propose a hybrid encoding method without extra decision-making costs. We encode each tile into two versions. The encoded version is generated through video compression, resulting in short transmission time, but requiring decoding on the mobile device. The unencoded version needs a longer transmission time while eliminating decoding on mobile devices. Both versions are then fused into tiles to be played on the mobile device.

The point cloud size to be decoded in real-time is limited, which is approximate proportional to CPU status (CPU threads and CPU's current load). Therefore, we leverage the real-time CPU status to decide the fraction of two tile versions.

C8: Rendering. We aim to address the limitations of volume rendering. NeRF and its variants rely on complex MLPs for rendering, which demand general-purpose computing capabilities. Therefore, we redesign the rendering by implementing the rendering steps within the classic rasterization pipeline and using z-buffering with binary transparency, we are able to avoid the need for sorting polygons into depth order for each new view, allowing for efficient loading of the rendering data into the GPU at the start of execution. The optimization procedure yields a mesh embedded with feature vectors (Each vertex contains a feature vector instead of RGB) and a small MLP (converts view direction and features to RGB).

C9: System-level Integration. VoINR focuses on video-on-demand scenarios accessible cross a variety of devices and operating systems. (e.g., YouTube and Netflix kind of systems), which is to our knowledge the first INR-enhanced VV streaming system VoINR (§4). The proposed VoINR is illustrated in Fig. 3.

We conduct training and reconstruction of NeRVo from the ground up, treating it as a series of VV frames. The process of capturing the video involves the input of multi-view videos along with corresponding poses. On the server end, the VV is segmented into fixed-sized frame chunks and further divided into tiles. Each of these tiles undergoes a transformation into multiple versions, which are then encoded to diminish the overall size of the VVs. The information pertaining to each tile, including aspects such as resolution, resources, and URLs, is securely stored within the MPD file.

Video player. Our interactive video player is deployed on the web page, which is rendered by WebGL through the three.js library and canvas element. By providing a consistent and user-friendly interface, the system enables users to easily access and interact with VV content, regardless of their device or location. The viewer allows users to interact with a mouse, finger, and VR headset to rotate, pan, and zoom. The demo is shown in Fig. 2.

5 EXPERIMENTS

Devices. Commodity machine with an RTX 3090(24GB, primitive of training), an Intel(R) Platinum 8255C CPU@2.50GHz, 45GB RAM. Phone (Huawei Mate 30, Xiaomi 10, iPhone 12), tablet, laptop (iPad air 3), Gaming laptop (Legion), PC (an AMD 5950X CPU @ 3.40GHz, and an NVIDIA GeForce GTX 1060 GPU), VR headset(connected PC with RTX 3060)

Dataset. Realistic 360° Synthetic [23], LLFF dataset [22].

Metrics and Baselines. Quality is evaluated by PSNR, SSIM and LPIPS [34]. We compare the following baselines:

- Multi-View Stereo (MVS): frame-by-frame reconstructed of **meshes**, usually with better quality than **PtCl**.
- LLFF [22]: Frame-by-frame rendering of multiplane images with pretrained model. **Fast but huge size**.
- NeuralVolumes (NV) [20]: One prior-art volumetric video rendering method using a warped canonical model.
- SNeRG [14]: Except for MobileNeRF, this is the only NeRF method that can work on lower-powered devices.
- Instant-NGP (Instant) [25]: SOTA NeRF method on training speed using **a warped canonical CUDA model.**
- MobileNeRF [10]: Method that could run on mobile devices.

Implementation Details. We integrate all the components into VoINR, Our implementation consists of 46.1k lines of code (LoC) for the server-side training and streaming pipeline, and 2.3k LoC for the client-side renderer. After installation, the renderer takes up 148MB. Our implementation refers the implementation of Instant-NGP and JNeRF [15]. The video player runs on the web browser with multiple web libraries to support the functionality of modules and is uniformly compatible with most of existing devices.

NeRVo Performance. NeRVo is a careful trade-off between various targets. Notably, our results show a remarkable 200x-400x speedup in training time (1-2 days to 7 minutes) and a 400x-2000x speedup (from 10 seconds/frame to 40-200 frames/second) compared with the original NeRF. As shown in Table. 3, our method achieves faster training, rendering, and mobile device compatibility in comparison to other NeRF methods while maintaining video quality.

Method	Train. Speed	Render. Speed	Train. Cost	Video Size	Mobile Support	Edit Support	Streamable
NeRF					✗	✗	✗
Instant-NGP					✗	✗	✗
PlenOctrees					✗	✗	✗
MobileNeRF					✓	✓	✗
NeRVo (Ours)					✓	✓	✓

Table 3: Comparison of NeRVo with other NeRF variants (Green: Large, Streamable: support Adaptive Bitrate Streaming)

Comparison with PtCl-based VV. We evaluate over four network conditiosns: Home WiFi, life LTE, fluctuating, and low bandwidth network. We use 8 VVs from Synthetic dataset generated by NeRVo and 8i PtCl dataset [1]. The preliminary results show that NeRVo significantly boosts the QoE compared with PtCl. Our model can provide a similar level of detail with high-density PtCl-based architecture and achieve an acceptable QoE. The degradation of resolution will not bring a QoE leap unlike PtCl, which is of an enormous data stream. The network bandwidth is not the biggest bottleneck. The suboptimum resolution can also achieve a satisfying QoE. our model can swiftly adjust the resolution under a changeable environment. Even at low resolution, our model can provide a similar level of detail with high-density PtCl and achieve an acceptable QoE. Under fluctuating, VoINR can swiftly adjust the bitrate without excessive overhead. The stall time is largely reduced. INR-enhanced VV shows its unique advantages compared to PtCl.

Comparison with Other Methods. The VoINR has several common points with ViVo [13] and YuZu [33]. YuZu does not require viewport prediction, exhibiting stable performance but incurs extra transmission. Its SR rate is less than 4 and requires extra offline training for each chunk, thereby resulting in additional transmission and cost. Compared with ViVo, VoINR could fully utilize the network and computing resources.

Comparison with Transcoding-based Streaming. We compare VoINR with Vues [19]. It needs to be pointed out that the transcoding streaming actually transmits the 2D video stream. The QoE model of two methods is completely different. Although Vues could largely reduce data usage, it diminishes the resolution and fps, multiplies the playback latency, and need edge support. This can be particularly noticeable in our interactive applications and impact the user's ability to interact with the content in real-time. SR and transcoding are orthogonal approaches to VoINR.

6 DISCUSSION

User Study. The QoE metric alone is insufficient to fully evaluate our system. Therefore, we carried out IRB-approved user studies to gather ratings from actual users. Our system, referred to as *sysname*, was deployed on a variety of devices and used by 30 participants, who were given the opportunity to view nine Virtual Videos (VVs). We provided the viewers with the freedom to move, rotate, pause, and replay the videos as they wished. The findings from these studies reveal that *sysname* outperforms traditional methods in terms of users' QoE, as well as their satisfaction with the system's realism, immersion, and interactivity.

Real-world Applications. We also capture videos and use our technique to reconstruct real-world objects. We create an augmented reality application to render high-quality videos in our real-world using the rear camera from various views. It has realism comparable to human perception and certain robustness to noise and artifacts in input data. As we reconstruct the video frame-by-frame, VoINR could not handle dynamic scenes with super fast motions and occlusion. It could not handle semi-transparencies, glossy surfaces, and large-scale scenes due to memory limitations on mobile devices.

Method compatibility. INR-enhanced VV also support super resolution for SR-enhanced streaming [11] and layered video streaming [31]. The method could also offload the rendering to edge servers to support transcoding streaming. We plan to explore this direction in our future work.

On-going works. The number of input videos will directly influence the quality of VV. We are facilitating VoINR with few input images for large-scale learning of VV [12]. To handle the scenes above, we will reconstruct the whole video rather than frame-by-frame, which further reduces the file size and training cost of video and overcome the above limitations. VoINR streams VoD VV. Recent INR works such as NeRF-SLAM[29] and NICER-SLAM [35] have already achieved real-time training and processing. We are working on INR-enhanced live streaming.

7 CONCLUSION REMARKS

NeRVo addresses a variety of unique challenges applying NeRF to VV streaming and resolves the drawbacks of PtCl-based VV. Our proposed system demonstrates the superior effects and the feasibility of INR-enhanced VV without requiring additional infrastructure support. Current INR and NeRF methods are developed mostly for proof-of-concept vision and machine learning research. We view our work as an initial step in the promising field of VV and hope it will encourage future network research on INR. The orthogonal approach is compatible with existing methods and offers a new avenue for more realistic mobile VR/AR experiences.

ACKNOWLEDGMENTS

The work was supported in part the Basic Research Project No. HZQB-KCZYZ-2021067 of Hetao Shenzhen-HK S&T Cooperation Zone, by NSFC (Grant No. 62293482 and No. 62102342), the Guangdong Basic and Applied Basic Research Foundation (Grant No. 2023A1515012668), the Shenzhen Science and Technology Program (Grant No. RCBS20221008093120047), the Shenzhen Outstanding Talents Training Fund 202002, the Guangdong Research Projects No. 2017ZT07X152 and No. 2019CX01X104, the Guangdong Provincial Key Laboratory of Future Networks of Intelligence (Grant No. 2022B1212010001), the Shenzhen Key Laboratory of Big Data and Artificial Intelligence (Grant No. ZDSYS201707251409055).

REFERENCES

[1] 8i voxelized full bodies-a voxelized point cloud dataset. http://plenodb.jpeg.org/pc/8ilabs/.

[2] Blender.

[3] Jpeg pleno database:microsoft voxelized upper bodies-a voxelized point cloud dataset. https://plenodb.jpeg.org/pc/microsoft.

[4] Lepcc. https://github.com/Esri/lepcc.

[5] Point cloud library. http://pointclouds.org.

[6] Point cloud visualizer addon. https://www.blendermarket.com/products/pcv.

[7] Stop motion obj addon. https://github.com/neverhood311/Stop-motion-OBJ.

[8] Draco 3d. https://google.github.io/draco, 2018.

[9] R. Adhikari and R. K. Agrawal. An introductory study on time series modeling and forecasting. *arXiv preprint arXiv:1302.6613*, 2013.

[10] Z. Chen, T. Funkhouser, P. Hedman, and A. Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *Proceedings of the IEEE/CVF CVPR*, pages 16569–16578, 2023.

[11] L. De Luigi, A. Cardace, R. Spezialetti, P. Z. Ramirez, S. Salti, and L. Di Stefano. Deep learning on implicit neural representations of shapes. *arXiv preprint arXiv:2302.05438*, 2023.

[12] A. Hamdi, B. Ghanem, and M. Nießner. Sparf: Large-scale learning of 3d sparse radiance fields from few input images. *arXiv preprint arXiv:2212.09100*, 2022.

[13] B. Han, Y. Liu, and F. Qian. Vivo: Visibility-aware mobile volumetric video streaming. In *Proceedings of the 26th annual international conference on mobile computing and networking*, pages 1–13, 2020.

[14] P. Hedman, P. P. Srinivasan, B. Mildenhall, J. T. Barron, and P. Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5875–5884, 2021.

[15] S.-M. Hu, D. Liang, G.-Y. Yang, G.-W. Yang, and W.-Y. Zhou. Jittor: a novel deep learning framework with meta-operators and unified graph execution. *Science China Information Sciences*, pages 1–21, 2020.

[16] Y. Jin, J. Liu, and F. Wang. Ebublio: Edge assisted multi-user 360-degree video streaming. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 600–601. IEEE, 2022.

[17] K. Lee, J. Yi, Y. Lee, S. Choi, and Y. M. Kim. Groot: a real-time streaming system of high-fidelity volumetric videos. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pages 1–14, 2020.

[18] J. Li, C. Zhang, Z. Liu, R. Hong, and H. Hu. Optimal volumetric video streaming with hybrid saliency based tiling. *IEEE Transactions on Multimedia*, 2022.

[19] Y. Liu, B. Han, F. Qian, A. Narayanan, and Z.-L. Zhang. Vues: practical mobile volumetric video streaming through multiview transcoding. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 514–527, 2022.

[20] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019.

[21] H. Mao, R. Netravali, and M. Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the conference of the ACM special interest group on data communication*, pages 197–210, 2017.

[22] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[23] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020.

[24] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[25] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022.

[26] J. Obert, J. van Waveren, and G. Sellers. Virtual texturing in software and hardware. In *ACM SIGGRAPH 2012 Courses*, pages 1–29. 2012.

[27] J. Park, P. A. Chou, and J.-N. Hwang. Rate-utility optimized streaming of volumetric media for augmented reality. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):149–162, 2019.

[28] I. Reimat, E. Alexiou, J. Jansen, I. Viola, S. Subramanyam, and P. Cesar. Cwipc-sxr: Point cloud dynamic human dataset for social xr. In *Proceedings of the 12th ACM Multimedia Systems Conference*, pages 300–306, 2021.

[29] A. Rosinol, J. J. Leonard, and L. Carlone. Nerf-slam: Real-time dense monocular slam with neural radiance fields. *arXiv preprint arXiv:2210.13641*, 2022.

[30] T. Takikawa, A. Evans, J. Tremblay, T. Müller, M. McGuire, A. Jacobson, and S. Fidler. Variable bitrate neural fields. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022.

[31] T. Takikawa, A. Evans, J. Tremblay, T. Müller, M. McGuire, A. Jacobson, and S. Fidler. Variable bitrate neural fields. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022.

[32] H. Yeo, C. J. Chong, Y. Jung, J. Ye, and D. Han. Nemo: enabling neural-enhanced video streaming on commodity mobile devices. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pages 1–14, 2020.

[33] A. Zhang, C. Wang, B. Han, and F. Qian. Yuzu: Neural-enhanced volumetric video streaming. In *19th USENIX Symposium on Networked Systems Design and Implementation*, pages 137–154, 2022.

[34] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[35] Z. Zhu, S. Peng, V. Larsson, Z. Cui, M. R. Oswald, A. Geiger, and M. Pollefeys. Nicer-slam: Neural implicit scene encoding for rgb slam. *arXiv preprint arXiv:2302.03594*, 2023.