FSVFG: Towards Immersive Full-Scene Volumetric Video Streaming with Adaptive Feature Grid

Daheng Yin
Simon Fraser University
Burnaby, BC, Canada
dya64@sfu.ca

Jianxin Shi
Nankai University
Tianjin, China
Simon Fraser University
Burnaby, BC, Canada
shijianxin@mail.nankai.edu.cn

Miao Zhang
Simon Fraser University
Burnaby, BC, Canada
mza94@sfu.ca

Zhaowu Huang
Southeast University
Nanjing, China
zwh@seu.edu.cn

Jiangchuan Liu*
Simon Fraser University
Burnaby, BC, Canada
jcliu@cs.sfu.ca

Fang Dong
Southeast University
Nanjing, China
fdong@seu.edu.cn

ABSTRACT

Given the truly immersive viewing experiences, full-scene volumetric videos have received increasing attention from both academia and industry. Their vast data volumes, however, present significant challenges for real-time streaming over today's bandwidth-limited Internet. Considering the vast amount of full-scene volumetric data to be streamed and the limited bandwidth on the Internet, achieving adaptive full-scene volumetric video streaming over the Internet presents a significant challenge. Inspired by the advantages offered by neural fields, especially the feature grid method, we propose FSVFG, a novel full-scene volumetric video streaming system integrated feature grids as the representation of volumetric content. FSVFG employs an incremental training approach for feature grids and stores the features and residuals between adjacent grids as frames. To support adaptive streaming, we delve into the data structure and rendering processes of feature grids and propose bandwidth adaptation mechanisms. The mechanisms involve a coarse ray-marching for the selection of features and residuals to be sent, and achieve variable bitrate streaming by Level-of-Detail (LoD) and residual filtering. Based on these mechanisms, FSVFG achieves adaptive streaming by adaptively balancing the transmission of feature and residual according to the available bandwidth. Our preliminary results demonstrate the effectiveness of FSVFG, demonstrating its ability to improve visual quality and reduce bandwidth requirements of full-scene volumetric video streaming.

CCS CONCEPTS

• **Information systems → Multimedia streaming**; • **Networks** → *Application layer protocols.*

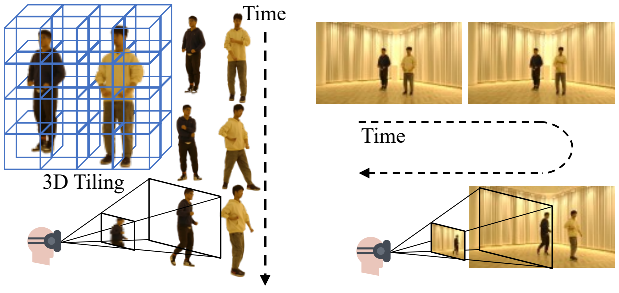
KEYWORDS

Volumetric Video Streaming, Neural Field, Full-scene Volumetric Video

ACM Reference Format:
Daheng Yin, Jianxin Shi, Miao Zhang, Zhaowu Huang, Jiangchuan Liu, and Fang Dong. 2024. FSVFG: Towards Immersive Full-Scene Volumetric Video Streaming with Adaptive Feature Grid. In *Proceedings of the 32nd ACM International Conference on Multimedia (MM '24), October 28–November 1, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3664647.3680908

1 INTRODUCTION

Volumetric video (VV) captures content in 3D, providing viewers with a six-degree-of-freedom (6DoF) motion. Recently, volumetric video has been at the forefront of immersive media and holds immense potential for revolutionizing sectors such as entertainment, gaming, education, and telecommunications. Since the volumetric video content is represented by dense explicit geometric structure, e.g., point cloud (PtCl), streaming volumetric video over the Internet consumes huge bandwidth. Recent studies have proposed solutions to mitigate data usage, such as leveraging tile-based viewport, occlusion, and distance visibility [6, 12, 45]. Most of them focus on streaming individual objects, e.g., a single person.



(a) Object-based video **(b) Full-scene volumetric video**

Figure 1: Illustrations of the object-based and full-scene VV

In contrast, full-scene volumetric video [8, 35] comprises both 3D objects and the environment, leading to a more immersive experience than the object-based volumetric video. As is shown in Figure 1, full-scene volumetric video typically has a significantly

*Corresponding author

MM '24, October 28–November 1, 2024, Melbourne, VIC, Australia.

ACM ISBN 979-8-4007-0686-8/24/10
https://doi.org/10.1145/3664647.3680908

large but less dynamic environment and relatively small areas with highly dynamic objects. The huge difference in object sizes poses a significant challenge for adaptive streaming [7].

Neural fields have demonstrated powerful capabilities in viewing quality with a compact model, offering new opportunities in full-scene volumetric video streaming. Neural fields such as NeRF store the volumetric content in a parameterized neural network. This network takes coordinates as inputs and outputs color and density, which can then be used in volume rendering. These methods have proven their ability to enhance visual quality while reducing storage costs compared to explicit representations. Recent studies have extended neural fields by incorporating time as an additional input to represent volumetric videos [34, 36]. However, the complexity of spatial changes in dynamic 3D scenes necessitates the use of numerous parameters in the neural network for accurate representation [29, 30], posing a substantial challenge to real-time rendering [21] and often resulting in compromised rendering quality. The use of numerous parameters also enlarges the data volume and makes it too large to stream, especially for full-scene volumetric content.

To achieve faster rendering and improve visual quality, recent studies have proposed the feature grid [27, 39] that stores a series of trainable positional encodings in a grid of features for each region in the scene. During inference, the input coordinates are first replaced by related encodings in the grids before being fed into the decoder, a single-layer MLP that outputs color and density. With fast query on grids and low complexity of the decoder, the feature grid can achieve better quality than other neural fields while maintaining real-time rendering performance [18]. In the field of streaming, most current feature grid streaming studies focus on streaming a single moving object [20, 36, 43, 47]. When considering the full scene, the environment data size significantly outweighs that of the moving objects [7], making these streaming methods impractical.

To this end, we propose FSVFG, an adaptive full-scene volumetric video streaming system that integrates feature grids as the representation of volumetric content. Our design and implementation of FSVFG pose two technical challenges.

How to avoid massive data transfer? Full-scene volumetric video typically has a relatively large but less dynamic environment with relatively small areas containing highly dynamic objects. Therefore, there is a strong inter-frame similarity among full-scene volumetric video frames. Inter-frame similarity is widely utilized to reduce the size of video frames. It allows for the storage and streaming of only the residuals in dynamic regions, thereby significantly reducing the bandwidth consumption of streaming. However, leveraging it within a feature grid presents a non-trivial challenge. Due to the high degree of freedom among feature grid parameters, it is difficult to maintain similarity between adjacent frames in the training process [44]. To address this issue, we propose inter-frame regularized training that exploits the inter-frame similarity of volumetric video frames by applying a regularization term in incremental training [44] to the feature grid.

How to support bandwidth adaptation? To support bandwidth adaptation, the streaming system should have a bitrate ladder that trades off quality and bandwidth to deal with fluctuating bandwidth. The traditional tile-based adaptive point cloud streaming [6] is implemented by controlling the density of tiled point clouds for network adaptation. However, in feature grid rendering, the density of the network does not necessarily represent the trade-off between quality and transmission. Therefore, they cannot achieve satisfactory performance in the streaming of feature grids. To tackle this problem, we take an insight into the rendering process and propose to select features and residuals to be sent according to a coarse ray-marching. We also delve into the data structure of the feature grid and achieve variable bitrate streaming by LoD (Level-of-Detail) and residual filtering. Based on these mechanisms, we achieve a heuristic algorithm to select the LoD of features and change the value of the threshold.

We have implemented FSVFG based on InstantNGP and conducted extensive experiments on typical datasets and comparisons with baselines. The key contributions are summarized as follows:

(1) To the best of our knowledge, FSVFG is a full-scene video streaming system integrated feature grid. By leveraging the ability of the feature grid in adaptive streaming, it reduces the bandwidth and optimizes conventional transmission.

(2) Based on the insight into the unique characteristics of the feature grid data structure and rendering process, we tackle these problems and design adaptive streaming mechanisms that improve visual quality and reduce network bandwidth requirements.

(3) We develop a prototype to evaluate the effectiveness of our approach. Our evaluation of typical datasets and comparisons with baselines demonstrate promising results.

2 BACKGROUND AND MOTIVATION

2.1 Volumetric Video Streaming

Volumetric video streaming has drawn substantial research attention in recent years. Current streaming systems can divided into two scopes. Some studies such as V-PCC [32] codec or Vues [23] project volumetric videos into 2D videos before streaming, while others stream the volumetric content directly. Our work falls within the latter scope.

Some previous studies in this scope have investigated volumetric video streaming based on point cloud and have achieved adaptive streaming [1, 37, 38, 42]. ViVo [6] was a pioneer in tiling-based visibility-aware adaptive volumetric video streaming, and other studies like GROOT [12], AITransfer [9], and YuZu [49] introduced different technologies to optimize tiling-based adaptive streaming. More recently, CaV3 [19] and Hermes [45] utilized inter-frame similarity of volumetric video frames to decrease bandwidth consumption of adaptive streaming, and MetaStream [5] integrates several innovations into a comprehensive system to accomplish volumetric video live streaming. However, with the huge amount of attributes (i.e. position and color) in the point cloud and other explicit representations, these methods suffer from large data volumes [18, 21] even with the use of well-designed encoders [41, 45].

Recently, innovative explicit representation methods such as 3D Gaussian splatting [11, 46, 48] and Plenoxels [4, 14] are proposed, but they are far from practical for networked applications due to their additional attribute requirements, such as spherical harmonic coefficients, which pose significant challenges for compression.

2.2 Neural Field

Neural fields such as NeRF [26] approximate the continuous signal in 3D space via a parametric continuous function that takes coordinates as inputs and outputs attribute vectors. These methods have proven their ability to enhance visual quality while reducing storage costs compared to explicit representations [3, 15]. Besides, they also eliminate the need for depth cameras in 3D reconstruction, thereby significantly reducing its costs and making it accessible to a wider audience. As such, these methods exhibit considerable potential for volumetric video streaming and have attracted significant research attention [10]. Unfortunately, despite the growing interest in streamable neural fields [39], most current research primarily focuses on compression [3, 14, 43], but not considering practical issues on volumetric video streaming such as visibility and bandwidth adaptation. To address this gap, our paper aims to leverage the benefits of neural fields to facilitate adaptive full-scene volumetric video streaming.

Within the large community of neural field research, many types of neural fields can represent volumetric video. The selection of neural field representation is critical for achieving adaptive volumetric video streaming and should consider several factors. Essentially, the chosen representation should deliver high quality and be capable of real-time rendering on client devices. Besides, it should be divisible temporally and spatially into smaller segments to achieve visibility-aware adaptation [6]. Furthermore, it should be capable of partitioning into different quality levels of varying sizes to accommodate varying bandwidth requirements for transmission [19, 49].

2.3 Feature Grid

Through investigation, we found that the feature grid can meet the above requirements [18]. Feature grid [2, 22, 28] is a special class of neural fields that is renowned in the research community for its superior quality and the ability to render at interactive rates. As is shown in Figure 2, a typical feature grid method comprises a feature grid, an MLP, and global Spherical Harmonics. Given a viewport as input, the feature grid method operates as follows [25, 40]: **(1)** Ray marching: sampling points along the ray $\vec{r} = \vec{o} + t\vec{d}$ from the target image and find the containing voxels in the feature grid of each point. **(2)** Positional embedding: the embedding vector of a point is linearly interpolated based on its relative position within the containing voxel in different resolutions. **(3)** Directional embedding: direction $\vec{d}$ of the ray is transformed into an embedding vector by Spherical Harmonics. **(4)** Inference: the positional and directional embedding vectors of each sample point are fed into the MLP, outputting the color and density. **(5)** Volume rendering: the color and density of the sample points are accumulated to derive the pixel color on the target image.

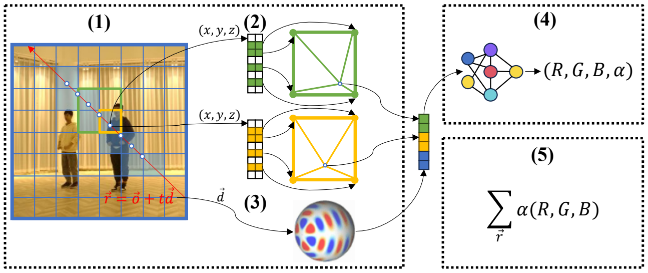

(1)
(2)
(x,y,z)
(x,y,z)
$\vec{r} = \vec{o} + t\vec{d}$
$\vec{d}$
(3)
(4)
(R,G,B,α)
(5)
$\sum_{\vec{r}} \alpha(R,G,B)$

Figure 2: Illustration of the feature grid method

By embedding coordinates into a high-dimensional space, the feature grid method shifts the complexity of the signal representation from the MLP to the feature grid, thereby achieving superior quality and rendering performance, and its grid structure facilitates spatial splitting. In recent studies, Liu et al. [18] have preliminarily explored achieving adaptive streaming by exploiting the separability of the feature grid, they did not consider inter-frame similarity to reduce bandwidth consumption. Liao et al. [43] have leveraged inter-frame similarity for frame compression based on feature grids, but they did not consider adaptive streaming and did not perform well for long videos. In summary, the feature grid is suitable for adaptive volumetric video streaming, but no practical system based on the feature grid exists.

To this end, this paper proposes achieving adaptive volumetric video streaming based on the feature grid. Specifically, we mainly use NGP [27] that stores the features in a compact table in our implementation and experiments, as this method represents the current mainstream of feature grid studies. As such, our innovations and methods could be extended to most other feature grid methods such as dense grid [39, 40], VQAD [24, 39] or VQRF [13].

3 SYSTEM OVERVIEW

The FSVFG system is designed to train a sequence of feature grids for volumetric video representation, and streams video-on-demand volumetric content represented by these feature grids from an Internet server to the client. The workflow of the FSVFG system consists of an offline training phase and an online adaptation phase, as is shown in Figure 3. In the offline phase, a scene is captured by a synchronized camera array from various perspectives. The captured data are then used to train a series of feature grids by inter-frame regularized training (§4). After the training process, each feature grid is partitioned into different Levels of Detail (LoD) and stored separately. In the online phase, the FSVFG server receives the predicted viewport from the client and sends the features or residuals selectively (§5.1). Then the selected features are filtered by LoD and inter-frame residuals are filtered by threshold according to the network condition, thereby achieving adaptation (§5.2).

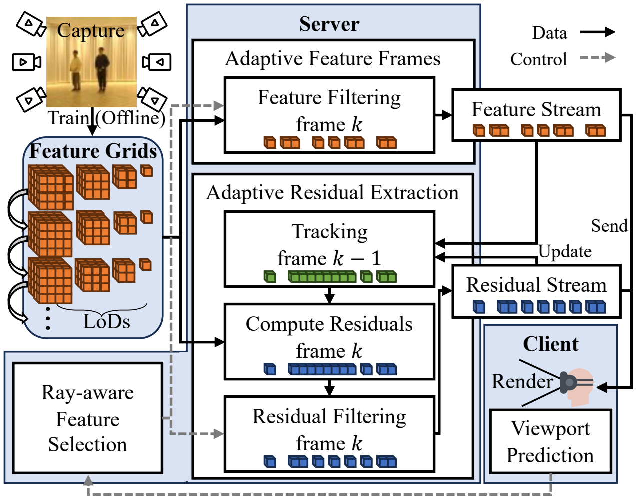


Figure 3: Overview of FSVFG streaming system

4 INTER-FRAME REGULARIZED TRAINING

The concept of neural field incremental training [44] stems from the understanding of 3D scenes. Unlike 2D videos, there is little "camera movement" in a 3D case. Those environment elements such as floors, walls, or buildings are unlikely to demonstrate high dynamics throughout a volumetric video [36]. Therefore, the volumetric video frames are well-aligned and inter-frame residuals most exist in those regions containing moving objects. This well-aligned inter-frame similarity can be leveraged to reduce the storage size of the frame and the bandwidth consumption of the streaming, without the need for motion estimation and compensation for alignment.

To leverage inter-frame similarity, incremental training loads the neural field parameters of the previous frame at the beginning of the training and freeze some parameters in the neural field during the training [14]. Ideally, the training process should only make minor adjustments to the parameters, and the inter-frame residual can be extracted and stored, which would be more space-efficient than storing the parameters of the neural fields.

However, the naive application of incremental training to the feature grid does not ensure minor changes to the parameters. In the feature grid, the parameters lack correlation with each other. In this case, some parameters in static areas may change during the incremental training process, leading to relatively larger residuals.

To address this issue, we propose to add a regularization term that minimizes the residual between the parameters of the current and previous frames in the loss function. The modified loss function $\mathcal{L}(\theta)$ is represented by Equation 1.

$$\mathcal{L}(\theta) = \mathcal{L}_0(\theta) + \lambda \sum_{i \in N} \theta_{i,k} - \theta_{i,k-1} \tag{1}$$

where $\mathcal{L}_0(\theta)$ denotes the original loss function (e.g. rendering loss [26, 44]), λ represents the weight of the regularization, $\theta_{i,k}$ represents one of the parameters in the feature grid of frame k, and N is the total numbers of the parameters in the feature grid. This regularization term can be incorporated into the loss function of any type of neural field, including feature grids, and can significantly reduce the volume of residuals that need to be stored, leading to improved space efficiency and less bandwidth consumption of streaming.

In practice, using frames captured simultaneously from different perspectives by the synchronized camera array [50] as training data, our inter-frame regularized training process operates as follows: for the initial frame, we train the feature grid without freezing and regularization. For each subsequent frame, we load the parameters of the previous frame and update only the parameters of the feature grid with the regularization term in the loss function. Through this process, we can sequentially train the feature grids for each frame, resulting in a sequence of files containing parameters of feature grids, each representing a frame of the volumetric video.

5 ONLINE ADAPTATION

5.1 Ray-aware Feature Selection

To achieve adaptation in volumetric video streaming systems, the common strategy is to partition a frame into tiles and stream the tiles within the viewport [6, 12]. However, for the feature grid, the density of the points does not necessarily represent the trade-off between quality and transmission. The rendering process of feature grids is significantly different from traditional explicit volumetric video representations such as point clouds. These traditional representations are typically rendered through rasterization or splatting that projects 3D objects into a raster image. In contrast, feature grids are rendered by ray marching, where each pixel on the output image corresponds to a ray, and each ray is traced within the grid. The pixel color is then calculated using the features of the voxels along the ray (Figure 2). Therefore, unlike in rasterization where each element in the received tiles contributes to the output image, in ray marching, only the features along the ray contribute to the output image. As a result, packaging feature grids in tiles for transmission could lead to the transmission of redundant features that do not contribute to the output image (Figure 4). Moreover, compact representations [27, 39] are widely adopted in storage-efficient feature grids such as VQAD [24, 39] or VQRF [13]. In these feature grids, one feature may be used in different positions more than once. Thus, existing spatially tiling methods would cause a single feature to appear in multiple tiles, causing transmission redundancy.

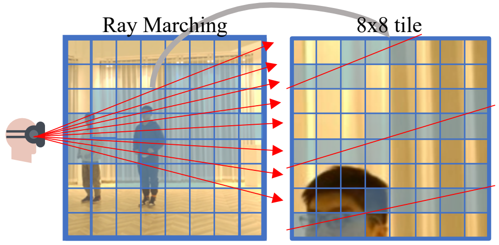


Figure 4: Illustration of ray marching and tiling

Considering the characteristics of feature grid rendering, we propose the Ray-aware Feature Selection mechanism that selects and sends only those features along the ray to the client. In practice, after the training process, the scene is divided into coarse cubes (i.e., 128^3 cubes), and sampling is performed in each cube to compute the density value inside it. The highest and lowest density values of each cube are then stored in a density grid. During the streaming process, our ray-aware adaptation component receives the predicted viewports from the client and performs a coarse ray marching (Figure 5) on the server.

The step size is set to match the ray marching in the client-side rendering process. Each step corresponds to a sampling point along the ray, and features of the voxels containing this point are selected, and their index is added to a set V_k. For empty regions (highest density=0), we apply the empty space skipping method [16] to skip the sampling point to the next occupied region (highest density>0). The output of ray-aware feature selection is a set of indices V_k, which will be used in the subsequent feature filtering and residual filtering processes.

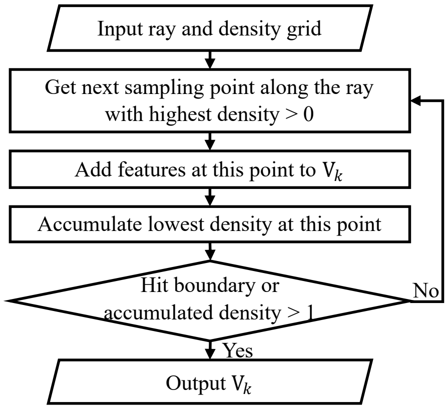

Input ray and density grid
Get next sampling point along the ray with highest density > 0
Add features at this point to V_k
Accumulate lowest density at this point
Hit boundary or accumulated density > 1
No
Yes
Output V_k

Figure 5: Illustration of Ray-aware Feature Selection

Since this process only involves query and accumulation operations without any time-consuming rendering processes (e.g., MLP inference), it can keep pace with the video playback frame rate when run on GPU devices on the server.

5.2 Variable Bitrate

Adaptive streaming aims to trade off video quality and bandwidth consumption under fluctuating network conditions. In full-scene volumetric video streaming, the adaptation process should also incorporate the progressive refinement of the environment due to its substantial size. Related works include Fumos [17] that integrate neural compression and octree-based coding to achieve progressive refinement for continuous point cloud video streaming. Similarly, LiveVV [7] enables adaptation by using different Point Cloud Density Levels (PDL) for full-scene volumetric video streaming. However, as mentioned before, the density of the points does not necessarily represent the trade-off between quality and transmission for the feature grid. Therefore, layer split methods based on density are not capable of feature grid streaming.

Besides, existing adaptation methods in volumetric video streaming do not take into account the dependencies between frames as they do not employ inter-frame correlation [31, 45] for compression. When the scene is represented by feature grids, we can leverage inter-frame correlation by computing and streaming the residuals of the features from adjacent frames for less dynamic regions to reduce bandwidth consumption. Despite the benefits, sending both features and residuals in a stream can introduce complexities in adaptive streaming, e.g. how to achieve variable bitrate on residuals and how to balance the streaming of features and residuals.

The goal of this section is to design mechanisms to deal with such complexities and support adaptive streaming.

5.2.1 Adaptive Feature Frames. To enable adaptation in streaming features, we divide the feature grid into different quality levels using level-of-detail (LoD), which is a characteristic provided by multiresolution encoding in most feature grid implementations [27, 39, 40]. In our feature stream, LoD can provide adaptation through an adjustable value L_k for the frame k, where only the features in those LoDs lower than this L_k will be sent to the user. For instance, at the beginning of the video, a lower L_k can be set to stream the scene, allowing the user to quickly achieve an immersive experience, then progressively increasing the L_k to refine the quality during playback.

The feature filtering process is outlined in Algorithm 1. In essence, feature filtering aims to quickly capture new features in the changing viewport, preventing incomplete views and degraded visual quality. It gathers those selected parameters in V_k that have not been sent to the client from lower to higher LoD levels, until the number of parameters reaches the given limit M.

5.2.2 Adaptive Residual Extraction. To enable adaptation in streaming residuals, an intuitive method is to store the residuals of different LoDs as separate quality levels. However, in adaptive streaming, after switching from a higher level to a lower level, the transmission of residuals for the higher level ceases. As a result, features in the higher level remain constant, while the features in the lower level keep changing, leading to a mismatch of features across different LoDs and degrading the visual quality. Therefore, simply applying LoD in both feature and residual streaming is not advisable.

Algorithm 1 Feature Filtering

```
Input: parameters of feature grid $\{\theta_{i,k}\}$, results of ray-aware feature selection $V_k$, set of index $S_{k-1}$ of parameters that have been sent to the client, limit $M$ on the number of parameters
Output: filtered features $F_k$, the updated set of index $S_k$
function FEAT_FILTER($\{\theta_{i,k}\}, V_k, S_{k-1}, M$)
    $F_k^{all} \leftarrow \{i | i \in V_k$ and $i \notin S_{k-1}$ and $\theta_{i,k} \neq 0\}$;
    $C_k \leftarrow |\mathrm{LoD}_1 \cap F_k^{all}|$; $L_k \leftarrow 1$; $l \leftarrow 2$;
    while $C_k + |\mathrm{LoD}_l \cap F_k^{all}| \leq M$ and $l \leq L$ do
        $C_k \leftarrow C_k + |\mathrm{LoD}_l \cap F_k^{all}|$;
        $L_k \leftarrow l$; $l \leftarrow l + 1$;
    end while
    $F_k \leftarrow \{\theta_{i,k} | i \in \mathrm{LoD}_l \cap F_k^{all}$ and $l \leq L_k\}$;
    if $M - |F_k| > 0$ and $L_k < L$ then
        $F_k^{rand} \leftarrow \mathrm{RandomSelect}(\mathrm{LoD}_{L_k+1} \cap F_k^{all}, M - |F_k|)$;
        $F_k \leftarrow F_k \cup \{\theta_{i,k} | i \in F_k^{rand}\}$;
    end if
    $S_k \leftarrow S_{k-1} \cup \{i | \theta_{i,k} \in F_k\}$
    return $F_k, S_k$;
end function
```

Based on the observation that larger residuals have a greater impact on visual quality, we divide the residual grid into different quality levels by their numerical value, thereby avoiding the quality degradation caused by mismatched features. Specifically, a threshold T_k is chosen adaptively for residual filtering for the frame k, and the residual $\Delta\theta_{i,k}$ for parameter $\theta_{i,k}$ of the features is computed according to Equation 2:

$$
\begin{aligned}
\Delta\theta_{i,k} &= f(\theta_{i,k} - \hat{\theta}_{i,k-1}, T_k) \\
\hat{\theta}_{i,k} &= \Delta\theta_{i,k} + \hat{\theta}_{i,k-1} \\
\Delta\theta_{i,1} &= f(\theta_{i,1} - \theta_{i,0}, T_1) \\
\hat{\theta}_{i,1} &= \Delta\theta_{i,1} + \theta_{i,0} \\
f(x,t) &= \begin{cases} x & |x| \geq t \\ 0 & |x| < t \end{cases}
\end{aligned}
\tag{2}
$$

where $\Delta\theta_{i,k}$ represents the residual of parameter i in frame k, while $\theta_{i,k}$ and $\hat{\theta}_{i,k}$ represents true value and tracked value of the parameter respectively. Therefore, during streaming, adjusting T_k can control the number of residuals to be sent.

Note that the feature grid and the residuals have been quantized to float16 (16-bit floating point) to reduce bandwidth consumption, which can cause quantization errors. In Equation 2, the tracked value $\hat{\theta}_{i,k-1}$ represents what the parameters would be after adding the residual, and residuals computed based on these tracked values would reset the error, preventing the errors from accumulating.

The residual filtering process is described in Algorithm 2, it aims to deliver those important residuals within the user's viewport. Given a limit M on the number of residuals, residual filtering first identifies the Mth largest residual using an efficient TopK algorithm on the GPU [33]. Considering that the small values have little effect on the visual quality, we set a lower bound T_m and take the larger value between Mth largest residual and T_m as the threshold T_k. Finally, the filtered residuals are computed according to Equation 2.

Algorithm 2 Residual Filtering

```
Input: parameters of feature grid $\{\theta_{i,k}\}$ and tracked values $\{\hat{\theta}_{i,k-1}\}$, results of ray-aware feature selection $V_k$, set of index $S_{k-1}$ of parameters that have been sent to the client, limit $M$ on the number of parameters, lower bound $T_m$ of residuals
Output: filtered residuals $R_k$, updated tracked values $\{\hat{\theta}_{i,k}\}$
function RES_FILTER($\{\theta_{i,k}\}, \{\hat{\theta}_{i,k-1}\}, V_k, S_{k-1}, M, T_m$)
    $R_k^{all} \leftarrow V_k \cap S_{k-1}$;
    $T_k \leftarrow \max(\mathrm{TopK}(\{|\theta_{i,k} - \hat{\theta}_{i,k}| | i \in R_k^{all}\}, M), T_m)$;
    for $i \in R_k^{all}$ do
        $\Delta\theta_{i,k}, \hat{\theta}_{i,k} \leftarrow \mathrm{ComputeResidual}(\theta_{i,k}, \hat{\theta}_{i,k-1}, T_k)$; ▷ Eq(2)
    end for
    $R_k \leftarrow \{\Delta\theta_{i,k} | i \in R_k^{all}$ and $\Delta\theta_{i,k} \neq 0\}$;
    for $i \notin R_k^{all}$ do
        $\hat{\theta}_{i,k} \leftarrow \hat{\theta}_{i,k-1}$
    end for
    return $R_k, \{\hat{\theta}_{i,k}\}$;
end function
```

5.2.3 Rate Allocation. As visual quality improves with the transmission of more features and residuals, maximizing visual quality equates to maximizing the use of available bandwidth. In this context, the goal of our allocation is to fully utilize the bandwidth to transmit as many features and residuals as possible.

Based on this idea, we provide a bandwidth constraint B_k for the frame k to control the bandwidth consumption. Assuming that the compression ratio r can be estimated, and the frame rate f is fixed, the capacity of parameters or residuals for each frame can be computed as $M = \frac{B_k}{fr}$. The adaptation decision is then to allocate the M to the residuals and features of each frame. We provide another control parameter $\gamma \in [0, 1]$ for the rate allocation, fill the γM by feature filtering and fill the rest part by residual filtering. Considering that in some cases the number of residuals may be few, we perform a reallocation of features after the residual filtering.

Algorithm 3 Rate Allocation at frame k

```
Input: frame rate $f$, bandwidth limitation $B_k$ and compression ratio $r$, bitrate allocation factor $\gamma$, viewport $\mathrm{Cam}_k$ uploaded by the client
Output: The set of features $F_k$ and residuals $R_k$ to be sent to client
$M \leftarrow \frac{B_k}{fr}$
Loading parameters $\{\theta_{i,k}\}$ and density grid for frame $k$
$V_k \leftarrow \mathrm{FeatureSelect}(\mathrm{Cam}_k)$;    ▷ Ray-aware Feature Selection
$F_k, S_k \leftarrow$ FEAT_FILTER($\{\theta_{i,k}\}, V_k, S_{k-1}, \gamma M$);    ▷ Alg.1
$R_k, \{\hat{\theta}_{i,k}\} \leftarrow$ RES_FILTER($\{\theta_{i,k}\}, V_k, S_{k-1}, M - |F_k|, T_m$);▷ Alg.2
if $M - |R_k| - |F_k| > 0$ and $L_k < L$ then
    $F_k, S_k \leftarrow$ FEAT_FILTER($\{\theta_{i,k}\}, V_k, S_{k-1}, M - |R_k|$);    ▷ Alg.1
end if
Save $S_k$ and $\{\hat{\theta}_{i,k}\}$ for next frame;
```

The rate allocation algorithm is described in Algorithm 3. With these algorithms, the bandwidth is filled with features in low LoD in the first frame. Then, in the early period of streaming, the bandwidth is allocated among features and residuals, progressively refining the visual quality while ensuring the smooth movement of the object in the scene. After a period of streaming, most of the features will have been sent to the client, and the streaming is now mainly filled by residuals. In this process, the control parameters B_k and γ can be used to trade off the quality and bandwidth consumption. Note that our allocation algorithms are highly parallelizable and can be executed on the server at high frame rates.

In a nutshell, our system provides parameters such as M and γ to trade off visual quality and bandwidth consumption, meeting the necessary conditions for adaptive streaming strategies. Existing adaptation algorithms, such as Lyapunov optimization [17] or multi-armed bandit [19] perform well and can be applied within this context. The specific design of these algorithms is not the primary focus of this study.

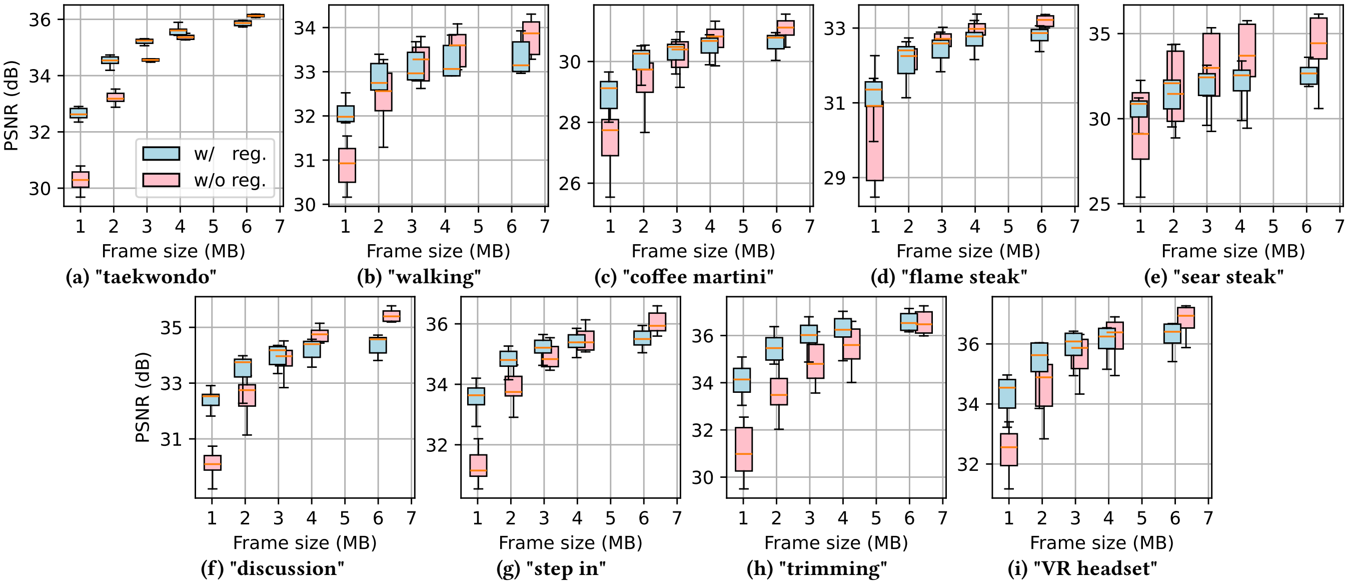
6 EVALUATION

According to the methods introduced above, we build a streaming system and evaluate FSVFG at a system level. Our dataset includes 9 stereo videos from 3 datasets: "taekwondo", "walking" from the st-nerf dataset [50], "coffee martini", "flame steak", "sear steak" from the Neural 3D Video Synthesis dataset [15], and "discussion", "step in", "trimming", "VR headset" from the Meet Room dataset [14]. We use the Instant-NGP [27] system as our testing platform, with the feature grid size set to 2^{19}, and the number of Levels of Detail (LoDs) fixed at 8.

6.1 System-Level Evaluation

6.1.1 Performance of Inter-frame Regularized Training. To demonstrate the performance of inter-frame regularized training, we compared the quality and size of models trained with and without inter-frame regularization. We train two sequences of feature grids for each video in the dataset using the incremental training and inter-frame regularized training introduced in Section 4. The feature grid for the initial frame is trained for 30,000 steps, while feature grids for subsequent frames are trained for 10,000 steps. After training, the features and residuals are extracted for each frame and stored individually as binary files. Then we set different thresholds to filter and compress the residuals as Equation 2, adjusting the threshold so that the size of the compressed frame ranged from 1 to 6MB. For each threshold configuration, we obtain the rendering results of viewports in the dataset and assess the PSNR (Peak Signal-to-Noise Ratio) against the ground truth.

Figure 6 shows the impact of thresholds on visual quality and frame size across different videos. It is evident that as the frame size increases (threshold of residual filtering decreases), incremental training without inter-frame regularization can achieve higher quality. However, as the frame size decreases, the quality degradation and the quality fluctuation are more severe. This is because the parameters within the feature grid are less interdependent compared to NeRF, giving them higher degrees of freedom to fluctuate during training. As a result, the incremental training cannot effectively control the parameter changes, leading to larger inter-frame residuals. Therefore, to achieve the same frame size, the threshold for residual filtering should be larger, which filters out more residuals and causes lower quality. Furthermore, with a larger threshold, the parameter errors must accumulate to higher values before residuals are used for compensation, which introduces greater instability to the video quality.



(a) "taekwondo" (b) "walking" (c) "coffee martini" (d) "flame steak" (e) "sear steak"

(f) "discussion" (g) "step in" (h) "trimming" (i) "VR headset"

Figure 6: PSNR distribution and average size of frames trained with and without inter-frame regularization

6.1.2 *Performance of Online Adaptation.* To demonstrate the performance of our ray-aware feature selection and variable bitrate mechanisms, we build a prototype system and evaluate it. We use a traditional tile-based method as the baseline. In the baseline scenario, the parameters of the features and residuals are clustered by the size of 4096 (16^3) as tiles. In streaming, a tile is sent whenever a ray passes through any feature within it. We play the volumetric video and collect viewport movement using a Quest 3 device for our evaluation. We set the bandwidth limit to 40Mbps ($M = 2 \times 10^5$) and $\gamma = 0.5$ and compare the video quality achieved by our methods and the baseline under this network constraint. Under this configuration, we also conduct an ablation study on the algorithms introduced in section 5.2 to demonstrate their impact on the overall performance. Due to the limited availability of ground truth data for fixed viewpoints in the dataset, we could not conduct quality tests on continuous viewpoint data using the dataset. Instead, we use the output of the original feature grid trained on each frame under the corresponding viewpoint as the ground truth, and the PSNR is calculated.

Our experiment results is shown in Figure 7. Under bandwidth constraints, our ray-aware feature selection method generally outperforms the traditional tiling strategy. This is because our method can transmit those features that contribute to the output image more precisely. The traditional tiling strategy transmits a large number of redundant features that do not contribute to the output image. Additionally, our ray-aware feature selection method can ensure relatively stable video quality when the user's viewpoint observes highly dynamic areas, e.g., "coffee martini" frames 20-40, "VR headset" frames 40-60. This is because our method saves the bandwidth required for feature transmission, allowing more bandwidth to be used for transmitting residuals.

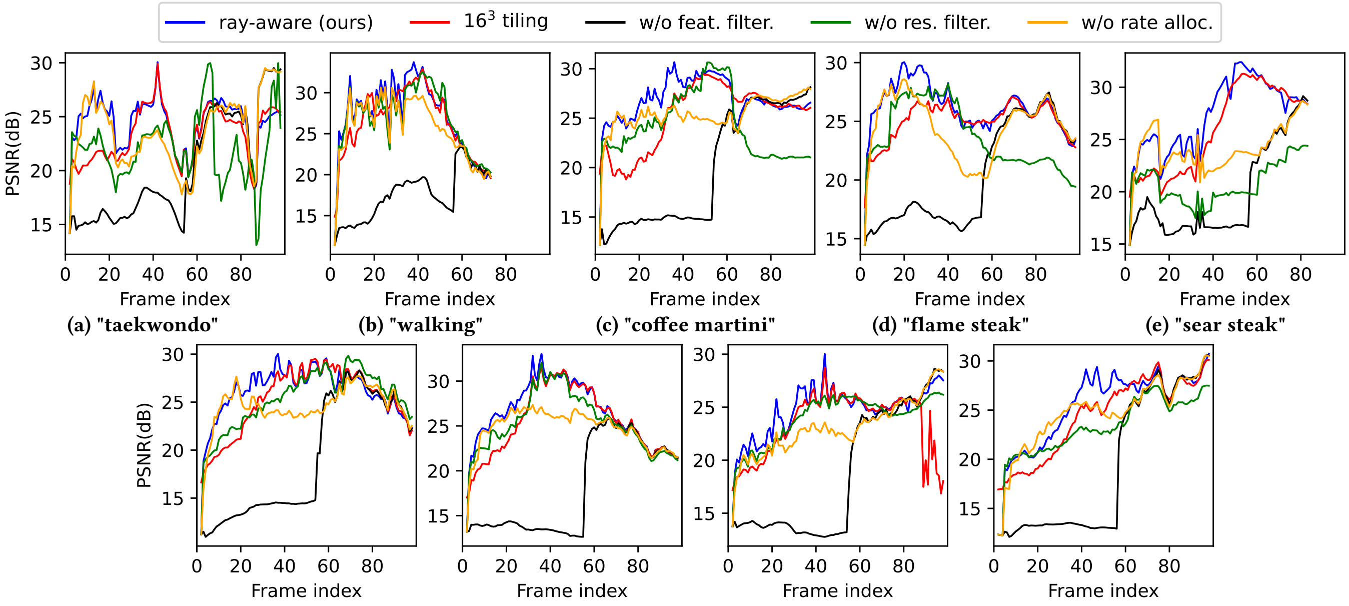
In the absence of feature filtering, the bandwidth is full of features, leaving minimal room for residuals, thus impeding the transmission of motion information and causing the video to freeze or stutter. In contrast, without residual filtering, the bandwidth is overwhelmed with residuals, leaving insufficient space for feature transmission to elevate LoD, thus resulting in a slower refinement of visual quality. Furthermore, without rate allocation, the entire bandwidth is allocated to either features or residuals, leading to fluctuations in visual quality.

When video quality tends to stabilize, our method can sometimes cause a quality decrease in a short period, e.g., "taekwondo" frames 90-100, "coffee martini" frames 80-100. In these video frames, the user's viewpoint moves quickly and just happens to move back to a location previously viewed. Our more precise feature selection strategy results in a lack of features for the current viewpoint that needs to be transmitted, leading to a decrease in video quality. This phenomenon can be further studied in future research.

6.1.3 *Performance of Runtime.* To evaluate the performance of our FSVFG runtime, we run our prototype system on PCs with Intel i7-12700 CPU and NVIDIA GeForce RTX 3080 8G GPU. Table 1 shows the average frame rate of frame loading and rendering, demonstrating that our implementation can support real-time playback experience. As the frame loading depends on the viewport data received from the client side, its frame rate is inherently constrained by the rendering rate. When the frame rate escalates to a higher range (above 30 fps), it is observable that the frame loading rate lags behind the rendering rate. Profiling indicates that the primary bottleneck resides in the disk read speed for frame retrieval.

Table 1: Average frame rate of rendering and frame loading process

Video	Process	1080p	720p	540p
"coffee martini"	rendering	12.0	20.3	34.5
	frame loading	12.0	20.3	28.3
"trimming"	rendering	12.1	22.4	35.6
	frame loading	12.1	22.4	27.3



(a) "taekwondo" (b) "walking" (c) "coffee martini" (d) "flame steak" (e) "sear steak"

(f) "discussion" (g) "step in" (h) "trimming" (i) "VR headset"

Figure 7: Comparison of PSNR fluctuations in volumetric video between our adaptation methods and the tiling-based method with a bandwidth limit of 40Mbps

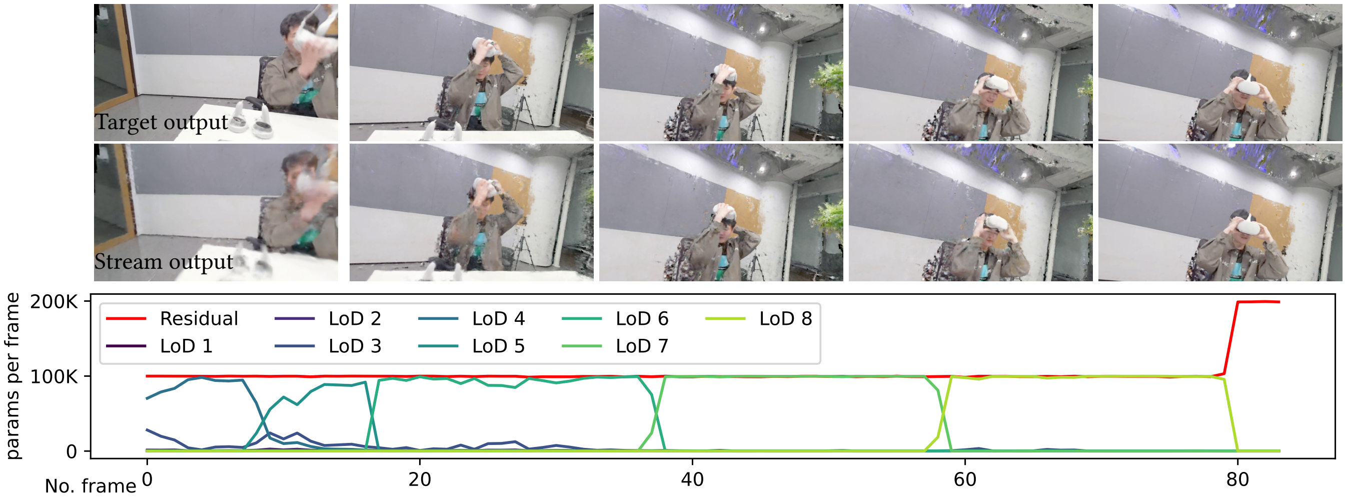


Figure 8: Case study: A timeline of visual samples and number of parameters sent for each frame (Video: "VR heasdet")

6.2 Case study

We conduct a case study on "VR headset" to present the detailed rate allocation and visual samples of our FSVFG system. Figure 8 illustrates the timeline in the FSVFG system when streaming "VR headset" with $M = 2 \times 10^5$ and $\gamma = 0.5$. In first 10 frames, the first few LoDs are downloaded, offering the user a blurry outline of the video content. As the streaming progresses through the subsequent frames, the transmission of data at lower LoDs is largely completed. The subsequent arrival of features from higher LoDs progressively refines the video quality. In the output visual samples, the scene gradually becomes clearer and more detailed as the streaming process continues. At the 80th frame, features from all the LoDs have fully arrived. From this point onwards, all bandwidth resources are dedicated to the transmission of residual data. As a result, the video quality stabilizes, maintaining a consistent level of clarity and detail for the remainder of the video stream.

7 CONCLUSION

This paper introduced the FSVFG system to enable adaptive volumetric video streaming based on feature grids. The proposed method utilized incremental frozen training to enable effective compression of frame size and achieve rate adaptation by ray-aware feature selection and filtering. By integrating our implementation into high-level applications, we can further develop practical implementations for real-world deployment.

8 ACKNOWLEDGMENTS

This research is supported by an NSERC Discovery Grant, a British Columbia Salmon Recovery and Innovation Fund (BCSRIF_2022_401), and a MITACS Accelerate Cluster Grant.

REFERENCES

[1] Gianluca Cernigliaro, Marc Martos, Mario Montagud, Amir Ansari, and Sergi Fernandez. 2020. PC-MCU: Point Cloud Multipoint Control Unit for Multi-User Holoconferencing Systems. In *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '20)*. 47–53.

[2] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. 2020. Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6970–6981.

[3] Junwoo Cho, Seungtae Nam, Daniel Rho, Jong Hwan Ko, and Eunbyung Park. 2022. Streamable Neural Fields. In *Computer Vision – ECCV 2022 (Lecture Notes in Computer Science)*, Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). 595–612.

[4] 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 Conference on Computer Vision and Pattern Recognition*. 5501–5510.

[5] 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 the 29th Annual International Conference on Mobile Computing and Networking*. Number 29. 1–15.

[6] Bo Han, Yu Liu, and Feng Qian. 2020. ViVo: Visibility-Aware Mobile Volumetric Video Streaming. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. 1–13.

[7] Kaiyuan Hu, Yongting Chen, Kaiying Han, Junhua Liu, Haowen Yang, Yili Jin, Boyan Li, and Fangxin Wang. 2023. LiveVV: Human-Centered Live Volumetric Video Streaming System. arXiv:2310.08205 [cs]

[8] Kaiyuan Hu, Yili Jin, Haowen Yang, Junhua Liu, and Fangxin Wang. 2023. FSVVD: A Dataset of Full Scene Volumetric Video. In *Proceedings of the 14th Conference on ACM Multimedia Systems* (Vancouver, BC, Canada) *(MMSys '23)*. 410–415.

[9] Yakun Huang, Yuanwei Zhu, Xiuquan Qiao, Zhijie Tan, and Boyuan Bai. 2021. AITransfer: Progressive AI-powered Transmission for Real-Time Point Cloud Video Streaming. In *Proceedings of the 29th ACM International Conference on Multimedia (MM '21)*. 3989–3997.

[10] Yili Jin, Kaiyuan Hu, Junhua Liu, Fangxin Wang, and Xue Liu. 2023. From Capture to Display: A Survey on Volumetric Video. *arXiv preprint arXiv:2309.05658* (2023).

[11] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (July 2023), 139:1–139:14.

[12] 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 the 26th Annual International Conference on Mobile Computing and Networking*. 1–14.

[13] Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Liefeng Bo. 2023. Compressing Volumetric Radiance Fields to 1 MB. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4222–4231.

[14] Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Ping Tan. 2022. Streaming Radiance Fields for 3D Video Synthesis. *Advances in Neural Information Processing Systems* 35 (Dec. 2022), 13485–13498.

[15] Tianye Li, Mira Slavcheva, Michael Zollhöfer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. 2022. Neural 3D Video Synthesis From Multi-View Video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5521–5531.

[16] Wei Li, K. Mueller, and A. Kaufman. 2003. Empty space skipping and occlusion clipping for texture-based volume rendering. In *IEEE Visualization, 2003. VIS 2003.* 317–324.

[17] Zhicheng Liang, Junhua Liu, Mallesham Dasari, and Fangxin Wang. 2024. Fumos: Neural Compression and Progressive Refinement for Continuous Point Cloud Video Streaming. *IEEE Transactions on Visualization and Computer Graphics* (2024), 1–11.

[18] Junhua Liu, Yuanyuan Wang, Yan Wang, Yufeng Wang, Shuguang Cui, and Fangxin Wang. 2023. Mobile Volumetric Video Streaming System through Implicit Neural Representation. In *Proceedings of the 2023 Workshop on Emerging Multimedia Systems (EMS '23)*. 1–7.

[19] 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)*. 173–183.

[20] Jia-Wei Liu, Yan-Pei Cao, Weijia Mao, Wenqiao Zhang, David Junhao Zhang, Jussi Keppo, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. 2022. DeVRF: Fast Deformable Voxel Radiance Fields for Dynamic Scenes. *Advances in Neural Information Processing Systems* 35 (Dec. 2022), 36762–36775.

[21] Kaiyan Liu, Ruizhi Cheng, Nan Wu, and Bo Han. 2023. Toward Next-generation Volumetric Video Streaming with Neural-based Content Representations. In *Proceedings of the 1st ACM Workshop on Mobile Immersive Computing, Networking, and Systems (ImmerCom '23)*. 199–207.

[22] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. 2020. Neural Sparse Voxel Fields. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS'20)*. 15651–15663.

[23] 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 the 28th Annual International Conference on Mobile Computing And Networking (MobiCom '22)*. 514–527.

[24] Omnia Mahmoud, Théo Ladune, and Matthieu Gendrin. 2023. CAwa-NeRF: Instant Learning of Compression-Aware NeRF Features. arXiv:2310.14695 [cs]

[25] Julien N. P. Martel, David B. Lindell, Connor Z. Lin, Eric R. Chan, Marco Monteiro, and Gordon Wetzstein. 2021. Acorn: Adaptive Coordinate Networks for Neural Scene Representation. *ACM Transactions on Graphics* 40, 4 (July 2021), 58:1–58:13.

[26] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2021. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *Commun. ACM* 65, 1 (Dec. 2021), 99–106.

[27] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Transactions on Graphics* 41, 4 (July 2022), 102:1–102:15.

[28] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 165–174.

[29] 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 International Conference on Computer Vision*. 5865–5874.

[30] 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 Trans. Graph.* 40, 6, Article 238 (dec 2021), 12 pages.

[31] Evaristo Ramalho, Eduardo Peixoto, and Edil Medeiros. 2021. Silhouette 4D With Context Selection: Lossless Geometry Compression of Dynamic Point Clouds. *IEEE Signal Processing Letters* 28 (2021), 1660–1664.

[32] Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo Cesar, Philip A. Chou, Robert A. Cohen, Maja Krivokuća, Sébastien Lasserre, Zhu Li, Joan Llach, Khaled Mammou, Rufael Mekuria, Ohji Nakagami, Ernestasia Siahaan, Ali Tabatabai, Alexis M. Tourapis, and Vladyslav Zakharchenko. 2019. Emerging MPEG Standards for Point Cloud Compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 (March 2019), 133–148.

[33] Anil Shanbhag, Holger Pirk, and Samuel Madden. 2018. Efficient Top-K query processing on massively parallel hardware. In *Proceedings of the 2018 International Conference on Management of Data*. ACM, 1557–1570.

[34] 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 Conference on Computer Vision and Pattern Recognition*. 16632–16642.

[35] Jianxin Shi, Miao Zhang, Linfeng Shen, Jiangchuan Liu, Yuan Zhang, Lingjun Pu, and Jingdong Xu. 2024. Towards Full-scene Volumetric Video Streaming via Spatially Layered Representation and NeRF Generation. In *Proceedings of the 34th Edition of the Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '24)*. Association for Computing Machinery, New York, NY, USA, 22–28. https://doi.org/10.1145/3651863.3651879

[36] 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 (May 2023), 2732–2742.

[37] Shishir Subramanyam, Irene Viola, Alan Hanjalic, and Pablo Cesar. 2020. User Centered Adaptive Streaming of Dynamic Point Clouds with Low Complexity Tiling. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20)*. 3669–3677.

[38] Shishir Subramanyam, Irene Viola, Jack Jansen, Evangelos Alexiou, Alan Hanjalic, and Pablo Cesar. 2022. Evaluating the Impact of Tiled User-Adaptive Real-Time Point Cloud Streaming on VR Remote Communication. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22)*. 3094–3103.

[39] Towaki Takikawa, Alex Evans, Jonathan Tremblay, Thomas Müller, Morgan McGuire, Alec Jacobson, and Sanja Fidler. 2022. Variable Bitrate Neural Fields. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings*. 1–9.

[40] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2021. Neural Geometric Level of Detail: Real-Time Rendering With Implicit 3D Shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11358–11367.

[41] Jeroen van der Hooft, Tim Wauters, Filip De Turck, Christian Timmerer, and Hermann Hellwagner. 2019. Towards 6DoF HTTP Adaptive Streaming Through Point Cloud Compression. In *Proceedings of the 27th ACM International Conference on Multimedia (MM '19)*. 2405–2413.

[42] Irene Viola and Pablo Cesar. 2023. Chapter 15 - Volumetric Video Streaming: Current Approaches and Implementations. In *Immersive Video Technologies*, Giuseppe Valenzise, Martin Alain, Emin Zerman, and Cagri Ozcinar (Eds.). 425–443.

[43] 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. arXiv:2304.04452 [cs]

[44] Shengze Wang, Alexey Supikov, Joshua Ratcliff, Henry Fuchs, and Ronald Azuma. 2023. INV: Towards Streaming Incremental Neural Videos. arXiv:2302.01532 [cs]

[45] Yizong Wang, Dong Zhao, Huanhuan Zhang, Chenghao Huang, Teng Gao, Zixuan Guo, Liming Pang, and Huadong Ma. 2023. Hermes: Leveraging Implicit Inter-Frame Correlation for Bandwidth-Efficient Mobile Volumetric Video Streaming. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23)*. 9185–9193.

[46] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 2023. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. arXiv:2310.08528 [cs]

[47] Minye Wu and Tinne Tuytelaars. 2023. NeVRF: Neural Video-based Radiance Fields for Long-duration Sequences. arXiv:2312.05855 [cs]

[48] Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. 2023. Real-Time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting. arXiv:2310.10642 [cs]

[49] Anlan Zhang, Chendong Wang, Bo Han, and Feng Qian. 2022. YuZu: Neural-Enhanced Volumetric Video Streaming. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. 137–154.

[50] Jiakai Zhang, Xinhang Liu, Xinyi Ye, Fuqiang Zhao, Yanshun Zhang, Minye Wu, Yingliang Zhang, Lan Xu, and Jingyi Yu. 2021. Editable Free-Viewpoint Video Using a Layered Neural Representation. *ACM Transactions on Graphics* 40, 4 (July 2021), 149:1–149:18.