TV-GS: Triplane Video-based Dynamic 3D Gaussian Splatting Representation for 2D Video Coding

Birendra Kathariya†, Dae Yeol Lee†, Tsung-Wei Huang†, Fangjun Pu,
Guan-Ming Su, Peng Yin and Sean McCarthy
Dolby Laboratories Inc.
1275 Market St, San Francisco, CA, 94103, USA
{birendra.kathariya, daeyeol.lee, tsung-wei.huang, fangjun.pu, pyin, sean.mccarthy}@dolby.com, guanmingsu@ieee.org
†Authors with equal contribution.

***Abstract*—3D Gaussian Splatting (3DGS) enables high-quality free-viewpoint rendering of volumetric video but transmitting 3DGS information efficiently is challenging. It is difficult to exploit spatial and temporal redundancies in the millions of independent Gaussians, each with separate attributes, that represent dynamic scenes. To address this, we propose Triplane Video–based 3DGS (TV-GS). TV-GS is an anchor-based neural Gaussian framework compatible with 2D video codecs. TV-GS uses anchor positions, triplanes, and lightweight Multi-Layer Perceptrons (MLPs) to represent the appearances and locations of neural Gaussians relative to their anchors. For dynamic scenes, 3DGS information is organized into 2D planes to form triplane videos that capture the scene's structure and motion over time. TV-GS enables spatial and temporal redundancies to be efficiently leveraged using standard 2D video compression, making it a practical and deployable solution for volumetric video streaming.**

***Index Terms*—3D Gaussian splatting, triplanes, compression, 2D video codec**

I. Introduction

3D Gaussian Splatting (3DGS) [1] has recently emerged as a new paradigm of 3D scene representation using sets of 3D Gaussians optimized via differentiable rendering. By capturing the full spatial and photometric structure of a scene, 3DGS enables free-viewpoint rendering and interactive navigation, making it a promising technology for next-generation immersive media such as AR, VR, and telepresence. Current research extends 3DGS to dynamic scenes through three main categories: (1) Per-Frame Gaussian Updates, which learn separate Gaussian parameters for each frame; (2) Deformation Fields, which map canonical Gaussians to frame-specific positions and attributes; and (3) Time-Conditioned Attributes (4DGS), which encode time as an additional input and jointly optimize Gaussian parameters in a continuous-time representation. Previous works on dynamic 3DGS primarily use deformation fields or time-conditioned attributes, which exploit temporal redundancy within the representation itself. While these methods reduce temporal overhead, they can struggle to accurately model fast motions or long range sequences.

In this paper, we introduce Triplane Video–based 3DGS (TV-GS), a per-frame update approach that encodes dynamic scenes as triplane feature videos. Feature planes are updated at each frame under supervision to ensure temporal stability, while spatio-temporal redundancy is captured implicitly through standard 2D video codecs rather than explicit motion coefficients or attributes. TV-GS replaces the per-anchor explicit parameters in Scaffold-GS [2] with two sets of triplanes and Multi-Layer Perceptrons (MLPs) that implicitly predict the locations and appearances of neural Gaussians. As a result, 68 of the 71 per-anchor parameters in [2] are no longer stored explicitly but are instead encoded in this implicit representation, substantially reducing the overhead.

During optimization, each anchor's 3D position is projected onto the 2D triplane planes to retrieve its features. This naturally organizes the triplane planes so that the 3D structure and motion of the scene appear as their 2D projections, forming video-like sequences well-suited for compression with standard codecs such as High Efficiency Video Coding (HEVC) [3] or Versatile Video Coding (VVC) [4]. Also, unlike prior categories, our method can model sequences of arbitrary length, maintaining high rendering fidelity even over long-range dynamics. The main contributions of this work are summarized below.

- We propose a Gaussian Splatting framework compatible with conventional 2D video codecs. The representation comprises triplanes (2D feature planes), anchor positions, and lightweight MLPs. This structure allows temporal redundancies in triplanes to be effectively exploited through standard video compression.
- We introduce supervision strategies that enforce sparsity and temporal consistency in triplanes, enabling 2D codecs to fully leverage frame-to-frame redundancy for maximum compression efficiency.
- We design a temporal update scheme with an Anchor Refresh Period (ARP), where anchor points and MLPs are refreshed periodically rather than every frame, reducing bitstream size and accelerating playback.

979-8-3315-7111-5/26/$31.00 ©2026 IEEE

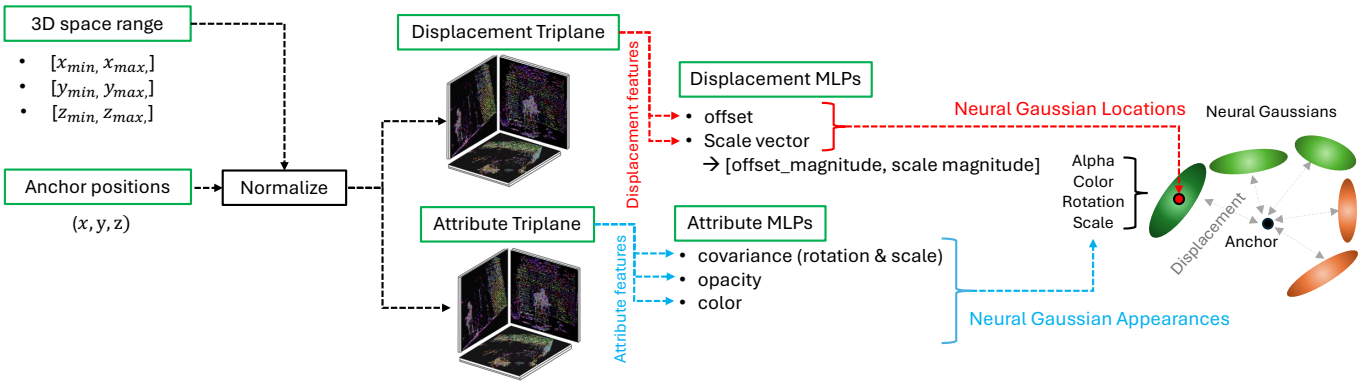


Fig. 1. Proposed Triplane-based 3DGS representation. Indicated in green boxes are what needs to be transmitted.

II. Related Work

Several works [5] aim to exploit correlations between nearby Gaussians to create more compact 3DGS representations. Scaffold-GS [2] introduces sparse anchor points, from which per-view neural Gaussians are generated and their attributes dynamically adjusted based on anchor features and view directions. While effective, it still requires transmitting explicit attributes for each anchor and does not exploit temporal redundancy in dynamic scenes. HAC [6] builds on this idea using a binary hash grid to provide context for anchors and an Adaptive Quantization Module that dynamically adjusts step sizes for different attributes, yielding more compact representation. CompGS [7] predicts coupled primitives from anchors and applies learned entropy modeling for compression. LightPred [8] achieves compactness through hierarchical representation, deriving child Gaussian attributes from parent attributes stored in a hash grid via MLP and attention mechanism. CSTG [9] proposes a learnable space-time mask to remove ineffective Gaussians and represents the geometry and view-dependent color using codebook and neural field, respectively, to reduce storage size.

A branch of works explored leveraging conventional 2D image or video codecs to compress 3DGS. In V^3 [10], Gaussian attributes are optimized across frame groups with an entropy-based loss for temporal consistency, integer-quantized, then Morton-sorted into 2D planes. These planes are compressed with Advanced Video Coding (AVC/H.264). SOG [11] integer-quantizes 3DGS primitives, arranges them into 2D square grids with Parallel Linear Alignment Sorting (PLAS), and compresses them using JPEG-XL. GIFStream [12] augments a canonical 3DGS with a deformation field with sparse time-dependent feature streams, prunes and reorganizes them into two temporally aligned videos, and compresses them using entropy coding or HEVC/VVC with end-to-end entropy supervision. CodecGS [13] represents 3DGS attributes with triplanes, applies frequency-domain entropy modeling with channel-wise bit allocation, and compresses the planes using HEVC.

III. Methodology

A. Representation Overview

Our representation builds on the anchor-based framework [2], where neural Gaussians are spawned from these anchors according to the given camera pose to effectively represent the 3D scene. Figure 1 illustrates the proposed representation, which consists of two branches: one predicting neural Gaussian locations (red arrows) and another predicting their attributes (blue arrows). Rather than explicitly storing all locations and attributes for each anchor, we achieve a compact representation using learned triplane feature maps [14] [15] and MLPs. Given an anchor position query, features are extracted from the triplane and passed through the MLPs to predict both the spatial distribution and view-dependent appearance attributes of the neural Gaussians.

We use two separate triplanes to model the distinct characteristics of location and appearance features. Details of each component—the anchors, the MLPs, and the triplanes—are described below.

Anchors. As in [2], we voxelize the scene using a voxel size ϵ, where the center of each voxel serves as a candidate anchor position. Assuming the optimal anchors for representing the scene have been determined, we denote their indices as $v \in [0, N_v - 1]$, where N_v is the total number of anchors. Let $\mathbf{x}_v = (x_v, y_v, z_v)$ denote the 3D coordinates of anchor v. The 3D space is bounded by $[x_{\min}, x_{\max}]$, $[y_{\min}, y_{\max}]$, and $[z_{\min}, z_{\max}]$. These bounds are used to normalize anchor positions, ensuring that the 3D range remains consistent throughout frame progression, along with triplane resolution.

The normalized anchor position $\hat{\mathbf{x}}_v = (\hat{x}_v, \hat{y}_v, \hat{z}_v)$ is computed using min-max normalization:

$$\hat{\mathbf{x}}_v = \left(\frac{x_v - x_{\min}}{x_{\max} - x_{\min}}, \frac{y_v - y_{\min}}{y_{\max} - y_{\min}}, \frac{z_v - z_{\min}}{z_{\max} - z_{\min}} \right) \quad (1)$$

so that each component lies in $[0, 1]$.

Triplanes. Triplanes encode 3D features by projecting them onto three orthogonal 2D planes. In our representation, we use two triplanes: *displacement* and *attribute*, each consisting of XY, YZ, and XZ planes with C feature dimensions per plane.

TABLE I
SUMMARY OF MLP INPUTS AND OUTPUTS.

Type	MLP	Input	Output
Displacement	Offset (Φ_o)	$\{\mathbf{f}_v^d\}_{v\in V}$, $\{\hat{\mathbf{x}}_v\}_{v\in V}$	Offset: $\{\mathbf{o}_{v,k}\}_{v\in V,k=0}^{N_k-1}$
	Scale vector (Φ_l)	$\{\mathbf{f}_v^d\}_{v\in V}$, $\{\hat{\mathbf{x}}_v\}_{v\in V}$	Offset Magnitude: $\{\mathbf{l}_v\}_{v\in V}$, Scale Magnitude: $\{\mathbf{m}_v\}_{v\in V}$
Attribute	Color (Φ_c)	$\{\mathbf{f}_v^a\}_{v\in V}$, $\{\boldsymbol{\omega}_{vc}\}_{v\in V}$	Color: $\{\mathbf{c}_{v,k}\}_{v\in V,k=0}^{N_k-1}$
	Opacity (Φ_α)	$\{\mathbf{f}_v^a\}_{v\in V}$, $\{\boldsymbol{\omega}_{vc}\}_{v\in V}$	Opacity: $\{\alpha_{v,k}\}_{v\in V,k=0}^{N_k-1}$
	Covariance (Φ_{cov})	$\{\mathbf{f}_v^a\}_{v\in V}$, $\{\boldsymbol{\omega}_{vc}\}_{v\in V}$	Rotation: $\{\mathbf{q}_{v,k}\}_{v\in V,k=0}^{N_k-1}$, Scale: $\{\mathbf{s}_{v,k}\}_{v\in V,k=0}^{N_k-1}$

The triplane resolutions along the x-, y-, and z-axes, denoted R_x, R_y, and R_z, are defined as

$$R_j = \left\lfloor \frac{j_{\max} - j_{\min}}{\epsilon\,\eta} + 0.5 \right\rfloor, \qquad j \in \{x, y, z\}, \tag{2}$$

where η is a subsampling factor used to downsample the triplane for a more compact representation.

Displacement triplanes $\mathbf{M}^d = \{\mathbf{M}^d_{XY}, \mathbf{M}^d_{YZ}, \mathbf{M}^d_{XZ}\}$ store features that guide where to spawn the neural Gaussians, effectively encoding information to predict their displacements from each anchor. They take the normalized anchor position $\hat{\mathbf{x}}_v$ as input and project it onto the XY, YZ, and XZ planes. Interpolated features from all planes are concatenated to form the displacement feature $\mathbf{f}_v^d$.

Attribute triplanes $\mathbf{M}^a = \{\mathbf{M}^a_{XY}, \mathbf{M}^a_{YZ}, \mathbf{M}^a_{XZ}\}$ encode appearance-related features, including color, alpha, rotation, and scale. They also take $\hat{\mathbf{x}}_v$ as input, project it onto the three planes, and concatenate the interpolated features to derive the attribute feature $\mathbf{f}_v^a$.

MLPs. MLPs are used to interpret triplane features and determine the location and appearance of neural Gaussians. Some attributes are view-dependent and therefore take the viewing direction of the anchor as an additional input alongside the triplane features. Given a camera position $\mathbf{x}_c = (x_c, y_c, z_c)$ and an anchor position $\mathbf{x}_v$, the viewing direction is computed as $\boldsymbol{\omega}_{vc} = \frac{\mathbf{x}_v - \mathbf{x}_c}{\|\mathbf{x}_v - \mathbf{x}_c\|_2}$.

Other attributes that may depend on an anchor's position incorporate the normalized anchor location $\hat{\mathbf{x}}_v$ (Eq. 1) as an additional input alongside the triplane features. Each MLP is a lightweight two-layer network. We evaluate the MLPs only on the set of visible anchors $v \in V$ for a given camera pose, rather than on all anchors. Table I summarizes the inputs and outputs of each MLP. For each anchor, we spawn N_k neural Gaussians indexed by $k = 0, \ldots, N_k - 1$ (we use $N_k = 10$). The positions and attributes of neural Gaussians are determined as follows:

$$\text{Positions:} \quad \{\boldsymbol{\mu}_{v,k}\}_{k=0}^{N_k-1} = \mathbf{x}_v + \{\mathbf{o}_{v,k}\}_{k=0}^{N_k-1}\,\mathbf{l}_v, \tag{3}$$

$$\text{Color:} \quad \{\mathbf{c}_{v,k}\}_{k=0}^{N_k-1}, \tag{4}$$

$$\text{Alpha:} \quad \{\alpha_{v,k}\}_{k=0}^{N_k-1}, \tag{5}$$

$$\text{Rotation:} \quad \{\mathbf{q}_{v,k}\}_{k=0}^{N_k-1}, \tag{6}$$

$$\text{Scale:} \quad \{\hat{\mathbf{s}}_{v,k}\}_{k=0}^{N_k-1} = \{\mathbf{s}_{v,k}\}_{k=0}^{N_k-1}\,\mathbf{m}_v. \tag{7}$$

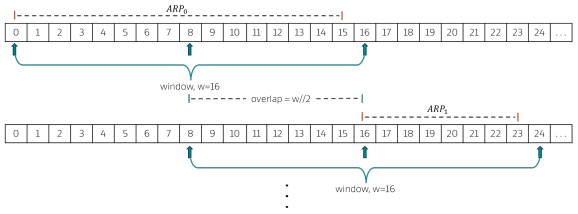


Fig. 2. Depiction of Overlapped-Sliding Window(OSW) and Anchor Refresh Period (ARP) through an example of window-size, $w = 16$.

B. Optimization and compression

Triplane optimization through rate-estimation. During training, we optimize both attribute and displacement triplanes through an entropy rate loss. This ensures the learned triplane representation is robust to quantization during the compression stage and exhibits low entropy. This also has a sparsification effect on the triplane features, which boosts 2D codec compression efficiency. Let $P(\cdot)$ and $P_{\text{cdf}}(\cdot)$ denote the probability mass function (PMF) and cumulative distribution function (CDF), respectively. We assume triplanes follow Gaussian distribution. For any plane $\mathbf{M}_p^t \in \mathbb{R}^{W\times H\times C}$ with type $t \in \{a, d\}$ and plane $p \in \{XY, YZ, XZ\}$, let $\Omega^{t,p}$ be the set of its element indices, and let $m_i^{t,p}$ denote the scalar at index $i \in \Omega^{t,p}$. The per-plane rate loss is

$$\mathcal{L}_{\text{rate}}^{t,p} = \frac{1}{|\Omega^{t,p}|} \sum_{i\in\Omega^{t,p}} \Big[-\log_2\big(P(\hat{m}_i^{t,p})\big) \Big]. \tag{8}$$

$$P(\hat{m}_i^{t,p}) = P_{\text{cdf}}\big(\tilde{m}_i^{t,p} + \tfrac{1}{2}\big) - P_{\text{cdf}}\big(\tilde{m}_i^{t,p} - \tfrac{1}{2}\big). \tag{9}$$

Here, $\hat{m}_i^{t,p} = Q(m_i^{t,p})$ is a quantization operation. Since $Q(\cdot)$ is non-differentiable, during training we simulate quantization by adding uniform noise: $\tilde{m}_i^{t,p} = m_i^{t,p} + \delta$, where $\delta \sim \mathcal{U}[-\frac{1}{2}, \frac{1}{2}]$. We estimate rates for the six planes $\{\mathbf{M}^d_{XY}, \mathbf{M}^d_{YZ}, \mathbf{M}^d_{XZ}, \mathbf{M}^a_{XY}, \mathbf{M}^a_{YZ}, \mathbf{M}^a_{XZ}\}$ separately and take their mean:

$$\mathcal{L}_{\text{rate}} = \frac{1}{6}\Big(\mathcal{L}_{\text{rate}}^{d,XY} + \mathcal{L}_{\text{rate}}^{d,YZ} + \mathcal{L}_{\text{rate}}^{d,XZ} + \mathcal{L}_{\text{rate}}^{a,XY} + \mathcal{L}_{\text{rate}}^{a,YZ} + \mathcal{L}_{\text{rate}}^{a,XZ}\Big). \tag{10}$$

Overlapped-Sliding Window(OSW) and Anchor Refresh Period (ARP). TV-GS uses implicit parameters—triplanes and MLPs—to predict neural Gaussian positions and attributes.

Inaccuracies in these implicit parameters during training can hinder anchor-position optimization and prevent a proper anchor population. Thus, we use a two-stage strategy: in stage 1, we adopt explicit parameter–based training [2] to populate anchor positions; then in stage 2 we use these anchor positions to initialize our main training and optimize MLPs and triplanes. This approach allows us to bootstrap point clouds learned from explicit parameter-based methods, adapting them to the implicit representation and expediting optimization. To promote temporal consistency and reduce transmission overhead, we maintain stage 1 anchors across frame groups and periodically refresh them to accommodate scene changes.

The concepts of overlapped sliding window (OSW) and anchor refresh period (ARP), using a window size of $w = 16$, are illustrated in Fig. 2. As shown in the top row of the figure, stage-1 anchor positions are collected from three equidistant frames at $\{\tau - \frac{w}{2}, \tau, \tau + \frac{w}{2}\}$. The sampled points are then sorted using Morton ordering, and every third point is selected to form the fused anchor set. For each first frame in the ARP, indexed as $\tau = \{\, i(\frac{w}{2}) \mid i \in \mathbb{Z}_{\geq 0}, i = 0 \vee i \geq 2 \,\}$, the fused anchor points are used to jointly optimize the MLPs and triplanes to represent the scene. These fused anchors and trained MLPs are then shared across the remaining frames within the same ARP, during which only the triplanes are further optimized to represent the scene dynamics.

Once all frames within an ARP are trained, the window shifts forward by $\frac{w}{2}$ time steps, and training proceeds similarly for the frames in the next ARP, as illustrated in the bottom row of Fig. 2. By design, there is a certain percentage of overlap in anchor points from one ARP to another, which helps mitigate abrupt fluctuations in the triplane feature space. The parameter sharing enabled by OSW and ARP provides following key advantages: (1) sharing anchor points and MLPs across frames within an ARP significantly reduces the overall bitrate; (2) during decoding and playback, only a single set of anchor points and five MLPs per ARP need to be decoded, reducing computation and latency; and (3) using the same anchors and MLPs enhances temporal consistency across triplanes, improving 2D codec compression efficiency and reducing flickering in rendered views.

Parameter re-initialization and temporal stability. The triplanes and MLPs parameter at $\tau = 0$ are optimized from scratch. However, we allow triplanes to be re-initialized from a previous frame and fine-tuned to adapt it to the current frame. Within an ARP, the first frame is trained for full 30K iterations and for the rest of the frames, only the triplanes are finetuned for 10K iterations. At the ARP boundary, i.e., last frame of the previous ARP and first frame of current ARP, only the triplane re-initialization is allowed as anchor points are obtained from the first stage training while MLPs are learned from scratch. This parameter re-initialization approach presents two key benefits: (1) It reduces the overall training time (2) It also helps to maintain temporal consistency. We also impose a temporal consistency (tc) loss between consecutive frames:

$$\mathcal{L}_{\text{tc}} = \frac{1}{6} \sum_{t \in \{a,d\}} \sum_{p \in \{XY, YZ, XZ\}} \frac{1}{|\Omega^{t,p}|} \left\| \hat{\mathbf{M}}^{t}_{p,\tau-1} - \tilde{\mathbf{M}}^{t}_{p,\tau} \right\|_1, \quad (11)$$

Here, $\hat{\mathbf{M}}^{t}_{p,\tau}$ and $\tilde{\mathbf{M}}^{t}_{p,\tau}$ denote the triplanes of type $t \in \{a, d\}$ and plane $p \in \{XY, YZ, XZ\}$ at frame τ, quantized by rounding and noise addition, respectively. The term $|\Omega^{t,p}|$ denotes the number of elements in the triplane. This constrains triplane features to vary only at grid locations aligned with actual 3D motion, enabling a 2D codec to leverage inter-coding (e.g., low-delay P and random access) for compression efficiency. The final loss of the proposed TV-GS is given as,

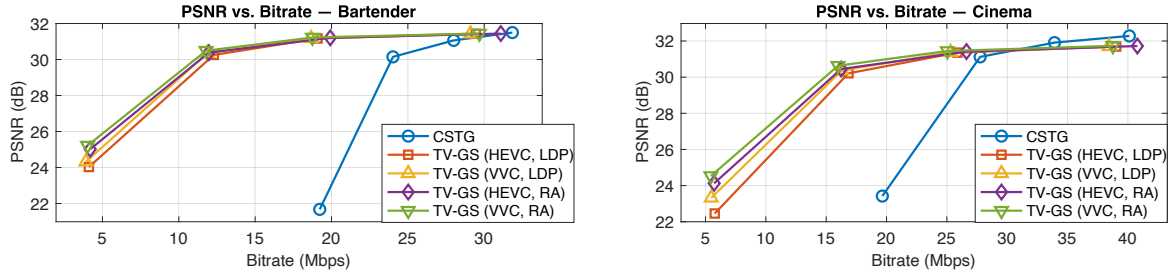
$$\mathcal{L}_{\text{TVGS}} = \mathcal{L}_1 + \lambda_{\text{SSIM}}\mathcal{L}_{\text{SSIM}} + \lambda_{\text{vol}}\mathcal{L}_{\text{vol}} + \lambda_{\text{rate}}\mathcal{L}_{\text{rate}} + \lambda_{\text{tc}}\mathcal{L}_{\text{tc}}. \quad (12)$$

Here, $\mathcal{L}_1$ and $\mathcal{L}_{\text{SSIM}}$ are the ℓ_1 and SSIM [16] losses on the rendered images. The volume regularizer is given by $\mathcal{L}_{\text{vol}} = \sum_{k=0}^{N_k-1} \sum_{v=0}^{N_v-1} \text{Prod}(\hat{\mathbf{s}}_{v,k})$, where $\text{Prod}(\cdot)$ denotes the product of a vector's components, encouraging each neural Gaussian to occupy small volumes with minimal overlap [2]. We set $\lambda_{\text{SSIM}} = 0.2$, $\lambda_{\text{vol}} = 0.01$, $\lambda_{\text{rate}} = 5 \times 10^{-4}$, and $\lambda_{\text{tc}} = 0.01$.

2D Codec based Compression. Once all the N frames are trained with TV-GS, they are subjected to compression using a 2D video codec. In the encoder, TV-GS model parameters: anchor points, MLPs and triplanes are first pre-processed to convert them into integer values. Anchor points are converted to integer values simply by dividing them by voxel-size, ϵ and shifted to positive quadrant by subtracting with minimum values in each axis. MLPs are quantized to 16-bit integer values following "*min-max*" quantization. Triplane features are first clipped between the $(-30, 30)$ range. Then utilizing the same range as "*min-max*" values, the triplane features are quantized to 10-bit integer values mapping -30 to 0, 0 to 512 and 30 to 1023.

The triplanes are packed into b numbers of separate YUV videos to form triplane-videos. In this work, we choose YUV:420 10-bit UHD-picture format. The triplane features are packed into Y-channels while U and V channels are filled with dummy 512 values. We pack features planes of same-type, i.e. "XZ-plane", "YZ-plane" and "XZ-plane", from both the triplanes together in "XZ", "YZ" and "XZ" order. We allow top-left packing location to be multiple of 32 to respect the coding tree unit (CTU) size employed in video-codecs. While packing into a YUV video, corresponding parameters from models at time, t, are packed into the t^{th} frame of a YUV video. These YUV videos are encoded at various quantization-parameters (QPs) to form b bitstreams. The anchor points and MLPs are coded per ARP. We utilize octree, more specifically G-PCC [17] inspired hybridtree, based method to compress anchor points losslessly. The occupancy maps generated by octree and integer MLPs parameters can be further compressed utilizing any dictionary-based or entropy-based coder. In this work, we utilize Lempel–Ziv–Markov chain algorithm (LZMA) [18] coding for the efficiency. The pre-processing

Fig. 3. Visual comparison of rendered test-view on *Bartender*. *Top row*: Visual comparison between CSTG and TV-GS. *Bottom row*: Visual comparison depicting the effectiveness of different video codecs and coding modes on rendering.



(a) Bartender (b) Cinema

Fig. 4. Rate–Distortion (RD) performance on the MPEG dataset: (a) Bartender; (b) Cinema.

steps also generate metadata which also constitute a part of the bitstream. The metadata carries shape & size information of anchors, MLPs and triplanes as well as information related to the quantization that are essential for reconstruction.

IV. Experimental Results

Datasets. We tested TV-GS on two dynamic scenes from MPEG: Bartender, and Cinema. Bartender and Cinema have total of 21 views with 1920×1080 resolution. Following MPEG-GSC CTC [19], we test on total of 65 frames by selecting 50 and 235 frame as the first-frame for *Bartender* and *Cinema* respectively. We select 9^{th} and 11^{th} views as test-views.

Implementation Details. The TV-GS hyperparameters that control the rendered quality include the voxel size, threshold ϵ, triplane resolution $\{R_x, R_y, R_z\}$, channel size C, window size w, and MLP hidden-node size.

We carried all experimentation by setting the voxel size $\epsilon = 0.01$, triplane subsampling factor $\eta = 4$ and channel-size $C = 10$. With these configurations, the triplane dimensions (R_x, R_y, R_z, C) were computed to be $(363, 269, 324, 10)$ for *Bartender* and $(458, 293, 291, 10)$ for *Cinema* using equation 2. Similarly, we set window size $w = 16$, and the hidden-node size of all MLPs to 32 in all of our experiments.

TABLE II
BD-Rate performance of TV-GS against CSTG.

2D Codec (mode)	Bartender	Cinema
HEVC (LDP)	-54.26	-28.56
HEVC (RA)	-54.97	-34.91
VVC (LDP)	-56.33	-33.88
VVC (RA)	-56.90	-40.16

TABLE III
Size comparison with Scaffold-GS at similar PSNR level.

	Bartender		Cinema	
Method	**PSNR [dB]**	**Size [Mbps]**	**PSNR [dB]**	**Size [Mbps]**
Scaffold-GS	31.62	257.27	31.75	400.56
TV-GS (HEVC, LDP)	31.43	29.45	31.70	39.04
TV-GS (VVC, LDP)	31.43	29.10	31.70	38.43
TV-GS (HEVC, RA)	31.45	29.68	31.73	38.76
TV-GS (VVC, RA)	31.44	31.09	31.72	40.79

We use the aforementioned hyperparameters and configurations to train the TV-GS model. To obtain different rate–distortion (RD) trade-offs, the triplane video is further compressed at various quantization parameters (QP) using standard 2D video codecs. In this work, we employ the HEVC reference software (HM-18.0) and the VVC reference software (VTM-23.0), encoding the triplane video at $QP = \{7, 12, 17, 22\}$. The above QP values are tested under both low-delay P (LDP) and random-access (RA) coding modes. For all experiments, we disable the temporal filters and in-loop filters, and set *qpoffset* = 0.

Comparison. We evaluate TV-GS against two methods: Scaffold-GS [2] and CSTG [9], which is state-of-the-art method for compact dynamic 3DGS representation. The RD comparison between TV-GS and CSTG is shown in Fig. 4, and the corresponding *BD-Rate* gains are summarized in Table II.

From Fig. 4, we observe that at higher bitrates, TV-GS yields slightly lower quality than CSTG, but it achieves significantly better performance at lower bitrates. Overall, TV-GS shows more than 50% gain for *Bartender* and $\sim 30\%$ gain for *Cinema*. In comparison across the tested configurations, VVC demonstrates superior compression efficiency to HEVC under same coding modes. Furthermore, the RA configuration yields better rate–distortion performance than the LDP. Overall, the VVC-RA combination provides the most efficient compression among all tested settings.

A visual comparison between CSTG and TV-GS on the rendered test view is shown in the top row of Fig. 3. At similar PSNR levels, TV-GS produces significantly better reconstructions, particularly for foreground objects such as the person's face and hands. The figure also presents rendered views where the triplane video is encoded using HEVC and VVC in both LDP and RA modes at *QP*=22. Perceptually, VVC yields higher visual quality than HEVC, and RA provides better results than LDP under the same *QP* setting.

We also compare the performance with Scaffold-GS, where its voxel size is adjusted to match the PSNR obtained at the highest bitrate among our test configurations. The optimal voxel sizes are found to be $\epsilon = 0.08$ for *Bartender* and $\epsilon = 0.07$ for *Cinema*. Table III summarizes the bitrate (in Mbps) of TV-GS—under various coding configurations—and Scaffold-GS at comparable quality levels. As shown in the table, Scaffold-GS requires substantially higher bandwidth, while TV-GS achieves similar quality below 50Mbps, providing an 8–10$\times$ reduction in data size.

V. Conclusion

In this work, we utilize a triplane-based 3DGS representation and propose triplane-video, a 2D codec-compatible solution for modeling dynamic 3D scenes, which we refer to as TV-GS. The TV-GS framework consists of two sets of triplanes and five sets of MLPs to generate 3DGS attributes. To leverage the benefits of inter-frame coding in 2D video codecs, we address feature consistency in triplane-video sequences by employing three key methods: (1) sparsifying the triplane features as much as possible through entropy-supervised loss, (2) reducing feature fluctuation among successive triplane-video frames through inter-consistency loss, and (3) sharing anchor points and MLPs among ARP frames to further reduce feature fluctuation. We evaluate TV-GS on two MPEG dynamic 3D scenes and demonstrated the effectiveness of our proposed method in terms of RD performance.

References

[1] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering." *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.

[2] T. Lu, M. Yu, L. Xu, Y. Xiangli, L. Wang, D. Lin, and B. Dai, "Scaffold-gs: Structured 3d gaussians for view-adaptive rendering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 20 654–20 664.

[3] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, 2012.

[4] B. Bross, Y.-K. Wang, Y. Ye, S. Liu, J. Chen, G. J. Sullivan, and J.-R. Ohm, "Overview of the versatile video coding (vvc) standard and its applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 10, pp. 3736–3764, 2021.

[5] B. Kathariya, D. Y. Lee, T.-W. Huang, T. Shao, P. Yin, and G.-M. Su, "Gaussian splatting: State-of-the-arts and future trends," in *2025 International Conference on Computing, Networking and Communications (ICNC)*, 2025, pp. 876–881.

[6] Y. Chen, Q. Wu, W. Lin, M. Harandi, and J. Cai, "Hac: Hash-grid assisted context for 3d gaussian splatting compression," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2025, pp. 422–438.

[7] X. Liu, X. Wu, P. Zhang, S. Wang, Z. Li, and S. Kwong, "Compgs: Efficient 3d scene representation via compressed gaussian splatting," in *Proc. 32nd ACM Int. Conf. Multimedia*, 2024, pp. 2936–2944.

[8] J. Cao, V. Goel, C. Wang, A. Kag, J. Hu, S. Korolev, C. Jiang, S. Tulyakov, and J. Ren, "Lightweight predictive 3d gaussian splats," *arXiv preprint arXiv:2406.19434*, 2024.

[9] J. C. Lee, D. Rho, X. Sun, J. H. Ko, and E. Park, "Compact 3d gaussian splatting for static and dynamic radiance fields," *CoRR*, vol. abs/2408.03822, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2408.03822

[10] P. Wang, Z. Zhang, L. Wang, K. Yao, S. Xie, J. Yu, M. Wu, and L. Xu, "V^ 3: Viewing volumetric videos on mobiles via streamable 2d dynamic gaussians," *ACM Trans. Graph.*, vol. 43, no. 6, pp. 1–13, 2024.

[11] W. Morgenstern, F. Barthel, A. Hilsmann, and P. Eisert, "Compact 3d scene representation via self-organizing gaussian grids," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2025, pp. 18–34.

[12] H. Li, S. Li, X. Gao, A. Batuer, L. Yu, and Y. Liao, "Gifstream: 4d gaussian-based immersive video with feature stream," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2025, pp. 21 761–21 770.

[13] S. Lee, F. Shu, Y. Sanchez, T. Schierl, and C. Hellge, "Compression of 3d gaussian splatting with optimized feature planes and standard video codecs," *arXiv preprint arXiv:2501.03399*, 2025.

[14] S. Fridovich-Keil, G. Meanti, F. R. Warburg, B. Recht, and A. Kanazawa, "K-planes: Explicit radiance fields in space, time, and appearance," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 12 479–12 488.

[15] Z.-X. Zou, Z. Yu, Y.-C. Guo, Y. Li, D. Liang, Y.-P. Cao, and S.-H. Zhang, "Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 10 324–10 335.

[16] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[17] D. B. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, S. Teruhiko, and A. J. Tabatabai, "An overview of ongoing point cloud compression standardization activities: video-based (v-pcc) and geometry-based (g-pcc)," *APSIPA Trans. Signal Inf. Process.*, vol. 9, 2020.

[18] D. Salomon, *Data Compression: The Complete Reference*, 2007, with contributions by Giovanni Motta and David Bryant.

[19] MPEG-GSC, "Draft ctc for gaussian splat coding," *ISO/IEC JTC 1/SC 29/WG 7*, no. N1292, July 2025.