

StreamSTGS: Streaming Spatial and Temporal Gaussian Grids for Real-Time Free-Viewpoint Video

Zhihui Ke, Yuyang Liu, Xiaobo Zhou*, Tie Qiu

School of Computer Science and Technology, Tianjin University, China
kezhihui@tju.edu.cn, yvyang_liu@tju.edu.cn, xiaobo.zhou@tju.edu.cn, qitue@tju.edu.cn

Abstract

Streaming free-viewpoint video (FVV) in real-time still faces significant challenges, particularly in training, rendering, and transmission efficiency. Harnessing superior performance of 3D Gaussian Splatting (3DGS), recent 3DGS-based FVV methods have achieved notable breakthroughs in both training and rendering. However, the storage requirements of these methods can reach up to 10MB per frame, making stream FVV in real-time impossible. To address this problem, we propose a novel FVV representation, dubbed StreamSTGS, designed for real-time streaming. StreamSTGS represents a dynamic scene using canonical 3D Gaussians, temporal features, and a deformation field. For high compression efficiency, we encode canonical Gaussian attributes as 2D images and temporal features as a video. This design not only enables real-time streaming, but also inherently supports adaptive bitrate control based on network condition without any extra training. Moreover, we propose a sliding window scheme to aggregate adjacent temporal features to learn local motions, and then introduce a transformer-guided auxiliary training module to learn global motions. On diverse FVV benchmarks, StreamSTGS demonstrates competitive performance on all metrics compared to state-of-the-art methods. Notably, StreamSTGS increases the PSNR by an average of 1dB while reducing the average frame size to just 170KB.

Introduction

Free-viewpoint video (FVV) is a crucial application in Virtual Reality (VR) with significant potential in education, industry, and entertainment, as it provides immersive user experiences. However, the 4D representation of FVV requires substantial storage, creating challenges for real-time transmission and hindering its practical application.

Recent advances in Neural Radiance Field (NeRF) (Mildenhall et al. 2021) have significantly facilitated the development of FVV. These NeRF-based works (Li et al. 2022a; Wang et al. 2023b; Zheng et al. 2024) reconstruct a NeRF model for each timestamp and store the residuals of these model, thereby allowing the streaming of these residuals. Though enabling real-time transmission, the volume rendering used in NeRF prevents real-time rendering. The most recent 3D Gaussian Splatting (3DGS) (Kerbl et al. 2023) has achieved real-time

rendering by directly projecting 3D Gaussians onto the 2D image plane and has thus attracted much researches.

Leveraging the high performance of 3DGS, many efforts have been made to extend 3DGS to dynamic scene reconstruction. Dynamic 3DGS methods (Yang et al. 2024b; Duan et al. 2024) incorporate a time dimension directly into Gaussian attributes, while deformable 3DGS methods (Yang et al. 2024a; Wu et al. 2024a; Li et al. 2024) utilize canonical 3D Gaussians to represent the geometry in canonical space and model motion through deformation fields (e.g., MLPs, HexPlanes, and Hash Grid). In these works, Gaussian attributes and temporal features are tightly coupled, resulting in them unsuitable for streaming. Consequently, some studies (Sun et al. 2024; Girish et al. 2024; Gao et al. 2024) focus on reconstructing streamable dynamic 3DGS for FVV. These methods typically use the first frame to reconstruct a 3D Gaussian representation and then predict attribute offsets for subsequent frames using a neural network, employing a per-frame training strategy to model the scene changes. However, the frame-by-frame training method brings significant cumulative errors and hard to deal with new objects. Furthermore, these methods require training multiple Level-of-detail (LoD) models to accommodate varying network conditions. Importantly, attribute offsets prediction inevitably becomes a part of decoding latency. If a user wants to view frame i , they have to wait for all previous $i - 1$ frames to be inferred to obtain accumulated Gaussian attribute offsets.

In this paper, we propose the Streamable Spatial and Temporal Gaussian Grids (StreamSTGS) representation, designed to achieve real-time FVV while adapting to dynamic network conditions. We decouple dynamic 3DGS into canonical 3D Gaussians, a series of temporal features, and a deformation field. Then, we use temporal features and the deformation field to predict the deformation of canonical 3D Gaussian. Moreover, we apply a sliding window to aggregate adjacent temporal features to learn local object motions. To reduce model size, inspired by (Morgenstern et al. 2025), we represent canonical 3D Gaussians and temporal features as spatial grids and temporal grids, respectively. Ultimately, spatial grids are stored as images while temporal grids are saved as a video. Notably, image and video formats naturally support adaptive streaming without any extra training. However, utilizing temporal features rather than HexPlanes or Hash Grid makes StreamSTGS hard to learn

*Corresponding author.

global motions, leading to blurring of dynamic areas. To address this, we propose a transformer-guided auxiliary training strategy, which employs a transformer to learn global motions and distill the learned features into StreamSTGS. This allows us to discard transformer during rendering to maintain high FPS. Furthermore, different from the frame-by-frame training strategy used in existing GS-based FVV frameworks, we train our StreamSTGS using a Group of Pictures (GOP), enabling it to capture large motions and sudden changes. As a result, StreamSTGS achieves superior performance in reconstruction quality, storage size, rendering and training speed.

The contributions are summarized as follows:

- We propose StreamSTGS, a streamable spatial and temporal Gaussian grids representation for real-time FVV, where a sliding window is applied to temporal Gaussian grids to capture adjacent temporal motions.
- We introduce spatial and temporal smoothness loss to regularize spatial and temporal Gaussian grids, enabling their compression into images and videos, which significantly reduces model size and inherently adapts to dynamic network conditions.
- We develop a transformer-guided auxiliary training strategy, which helps StreamSTGS learn global motions without impact the rendering speed.

Related Work

Dynamic Scene Reconstruction

Given the success of NeRFs (Mildenhall et al. 2021; Yu et al. 2021; Müller et al. 2022; Chen et al. 2022), there are many methods have extended NeRF to 4D space-time radiance fields to reconstruct dynamic scenes from multi-view or monocular videos (Xian et al. 2021; Gao et al. 2021; Li et al. 2021; Park et al. 2021a; Fang et al. 2022; Park et al. 2021b; Du et al. 2021; Li et al. 2022b; Yan, Li, and Lee 2023; Liu et al. 2023; Tian, Du, and Duan 2023; Zhan et al. 2024). However, NeRF-based dynamic scene reconstruction methods are constrained by volume rendering, resulting in slow rendering speed. Consequently, recent works have focused on accelerating NeRF-based methods from different perspectives. Specifically, Mixvoxels (Wang et al. 2023a), and (Guo et al. 2023) use voxels to represent the geometry and appearance of 3D space. K-Planes (Fridovich-Keil et al. 2023), HexPlane (Cao and Johnson 2023), Tensor4D (Shao et al. 2023), and DMiT (Yang et al. 2025) decompose 4D space into multiple planes to enhance computational and storage efficiency. MSTH (Wang et al. 2024a), F^2 NeRF (Wang et al. 2023c), and (Park et al. 2023) utilize hash tables (Müller et al. 2022) to store features in voxels or planes. NeRFPlayer (Song et al. 2023) and HyperReel (Attal et al. 2023) adopt decomposed vector-matrix (Chen et al. 2022), while DaReNeRF (Lou et al. 2024) propose a discrete wavelet transform (DWT)-based representation to model the 4D space. Though these methods significantly accelerate the rendering speed, a considerable gap remains in achieving real-time performance.

Recently, following the revolutionary 3DGS (Kerbl et al. 2023), dynamic Gaussian Splatting has quickly gained at-

tention for dynamic scene reconstruction. For instance, deformable 3DGS methods (Luiten et al. 2024; Bae et al. 2025; Lu et al. 2024b; Shaw et al. 2024; Yang et al. 2024a; Huang et al. 2024; Zhao et al. 2024; Wan, Lu, and Zeng 2024; Wu et al. 2024a; Lu et al. 2024a; Duisterhof et al. 2023; Li et al. 2024; Katsumata, Vo, and Nakayama 2025; Lin et al. 2024; Xu et al. 2024; Yan et al. 2024; Zhu et al. 2024; Kwak et al. 2025; Lei et al. 2025; Fan et al. 2025) construct canonical 3D Gaussians and utilize a deformation field to deform each Gaussians to specific timestamps. Other works (Yang et al. 2024b; Duan et al. 2024) incorporate a time dimension into 3D Gaussian attributes to form 4D Gaussian primitives. Meanwhile, some works directly learn 3D Gaussian motions through spline function (Lee et al. 2024; Park et al. 2025). Kim et al. (Kim, Lim, and Han 2024) introduce uncertainty-aware regularization to improve reconstruction performance. These methods train a dynamic scene as a single model, which limits their streamability.

Free-viewpoint Video Reconstruction

The emergence of implicit neural representations offers a new paradigm for capturing FVV from multi-view videos. StreamRF (Li et al. 2022a), ReRF (Wang et al. 2023b), HPC (Zheng et al. 2024), (Zhang et al. 2024), FSVFG (Yin et al. 2024), and VRVVC (Hu et al. 2025b) employ a per-frame training strategy, where a NeRF model is initially trained on the first frames and then incrementally refined with field residuals to adapt to the current timestamp. This approach allows streaming only the field residuals rather than the entire NeRF model. Similarly, 3DGStream (Sun et al. 2024), HiCoM (Gao et al. 2024), QUEEN (Girish et al. 2024), 4DGC (Hu et al. 2025a) apply per-frame training strategy to 3DGS for FVV reconstruction. 3DGStream (Sun et al. 2024) utilizes InstantNGP (Müller et al. 2022) to predict Gaussian attribute offsets and streams InstantNGP directly, which have a large model size. HiCoM (Gao et al. 2024) prefer to store and stream Gaussian attribute offsets, but it still exhibits high spatial and temporal redundancy. QUEEN (Girish et al. 2024), VideoGS (Wang et al. 2024c), GIFStream (Li et al. 2025), and 4DGC (Hu et al. 2025a) further employ entropy encoding to compress Gaussian attribute offsets but suffer from high decoding latency. Moreover, frame-by-frame training strategy brings significant cumulative errors. VideoRF (Wang et al. 2024b) and TeTriRF (Wu et al. 2024b) use 2D grids to store temporal 3D voxel features and compress these grids as videos. Nevertheless, their NeRF-based representation and aggressive quantization limit rendering speed and reconstruction quality.

Preliminary

3D Gaussian Splatting (3DGS) (Kerbl et al. 2023) explicitly reconstructs a scene using anisotropic 3D Gaussians, which have five attributes: 1) Position $X \in \mathbb{R}^3$ (i.e., the mean of the Gaussian function), 2) Scale $S \in \mathbb{R}^3$, represented as a diagonal matrix, 3) Rotation matrix R , parameterized by a quaternion vector $Q \in \mathbb{R}^4$, 4) Opacity $O \in [0, 1]$, and 5) Color C , represented by spherical harmonic coefficients. The covariance matrix is derived from scale and rotation matrices

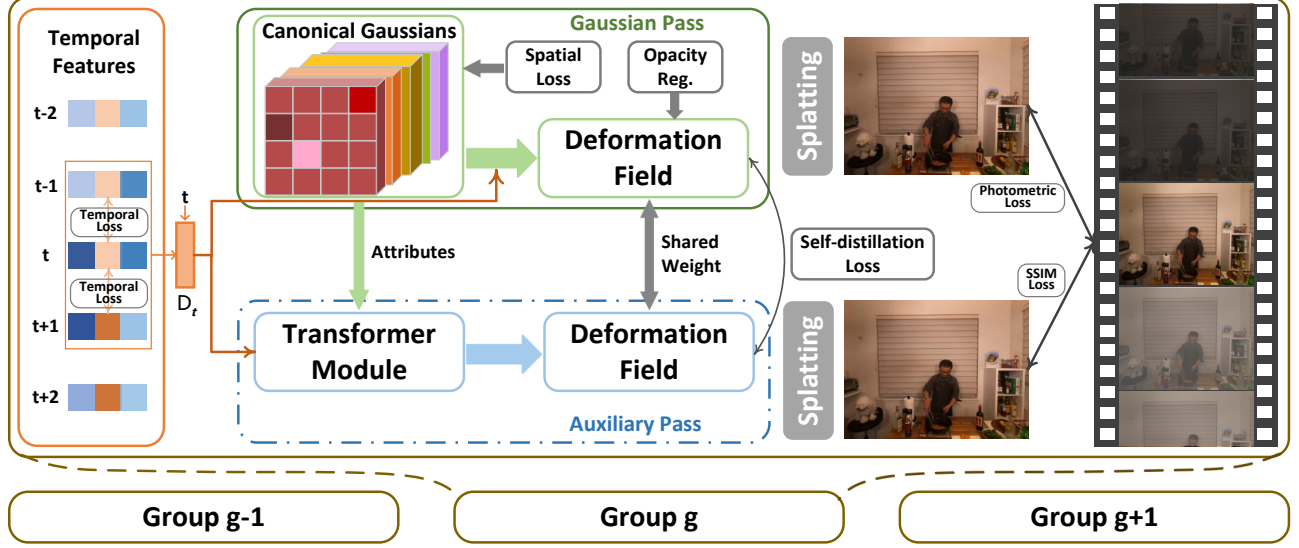


Figure 1: Overview of the StreamSTGS framework. First, the long video sequence is split into multiple groups. Within each group, a sliding window is employed to extract multiple temporal features, which serve as inputs to the deformation field for predicting the deformation of canonical 3D Gaussians. For real-time streaming, the canonical Gaussians are represented as images, while the temporal features are encoded as a video. To improve global motion learning, we introduce a Transformer-guided auxiliary training strategy, which can be removed during inference to achieve higher FPS.

as $\Sigma = RSS^T R^T$. For rendering, these 3D Gaussians are then projected into 2D Gaussians according to a given camera intrinsic matrix K and a viewing transformation matrix W through differential splatting (Zwicker et al. 2001):

$$X' = K((WX)/(WX)_z), \Sigma' = JW\Sigma W^T J^T \quad (1)$$

where J denotes the Jacobian matrix. Following depth sorting, the color of a pixel x is computed as:

$$color(x) = \sum_{m \in N} c_m \alpha_m \prod_{j=1}^{m-1} (1 - \alpha_j) \quad (2)$$

where c_m and α_m represent the color and opacity of Gaussian m .

Method

To realize real-time dynamic 3DGS streaming, our proposed StreamSTGS framework incorporates four key innovations, as illustrated in Fig. 1. First, we introduce a stream-friendly dynamic 3DGS representation, StreamSTGS, which facilitates real-time streaming through traditional video codecs. Second, we develop Gaussian attribute compression, which significantly reduces the model size of our StreamSTGS representation. Additionally, we propose a Transformer-guided auxiliary training strategy to help the global motion learning in StreamSTGS without compromising FPS performance. Finally, we introduce dynamic-aware density and Gaussian relocate schemes to improve the reconstruction quality under limited Gaussian quantity.

StreamSTGS Representation

We adopt the group of pictures (GOP) structure widely used in video compression, where each GOP comprises G frames, denoted as $\mathcal{T} = \{t_0, t_1, \dots, t_G\}$. The dynamic scene within each GOP is independently reconstructed using our proposed StreamSTGS representation. As shown in Fig. 1, StreamSTGS within a GOP consists of a canonical 3D Gaussians set $\mathcal{G}$, temporal features, and a deformation field. Unlike existing deformable 3DGS methods that typically rely on HexPlanes (Cao and Johnson 2023) to model motion in dynamic scenes, we employ temporal features $\mathcal{E} = \{e_0, e_1, \dots, e_E\}$ to directly learn dynamic features, enabling the streaming of temporal features in a manner akin to traditional video.

Considering the temporal correlation of motion in real-world dynamic scenarios, we employ a sliding window of length W to capture the motion relationship between adjacent frames, extracting W temporal features, as depicted in Fig. 1. For example, when $W = 3$, we concatenate the temporal features e_{i-1} , e_i , and e_{i+1} at timestamp t_i . For the next timestamp t_{i+1} , we concatenate e_i , e_{i+1} , and e_{i+2} . These concatenated features are then processed by a temporal MLP D_t to predict temporal feature f_i :

$$f_e = \text{concat}(e_{i-1}, e_i, e_{i+1}) \quad (3)$$

$$f_i = D_t(f_e, \gamma(t_i)), \quad (4)$$

$$\gamma(t_i) = (\sin(2^l \pi t_i), \cos(2^l \pi t_i))_{l=0}^{L-1}, \quad (5)$$

where position encoding (Mildenhall et al. 2021) is applied to transform timestamp t_i to a high frequency representation

of dimension L . This approach not only captures the motion relationships between adjacent frames but also reduces the number of temporal features to $E = G + W - 1$.

The original 3DGS uses spherical harmonic to represent view-dependent color, which requires large storage and is not well-suited for real-time streaming. Therefore, we use a three-channel tensor C to learn view-independent color and employ a color decoder D_c and an opacity decoder D_o to model view-dependent and time-varying color as follows:

$$\Delta O = \tanh(D_o(f_i, \text{view})), \Delta C = D_c(f_i, \text{view}), \quad (6)$$

where view denotes the camera direction. Since dynamic objects may appear or disappear over time, we apply the tanh activation function to the output of the opacity decoder to model its variations. Moreover, we use a velocity decoder D_v and a covariance decoder D_{cov} to predict the deformation of position $\Delta X = D_v(f_i)$ as well as scale and rotation changes $[\Delta S, \Delta Q] = D_{cov}(f_i)$. The final deformed 3D Gaussians $\mathcal{G}^i$ at timestamp t_i are given by:

$$\mathcal{G}^i = (X^i, S^i, R^i, O^i, C^i) \quad (7)$$

$$= (X + \Delta X, S + \Delta S, Q + \Delta Q, O + \Delta O, C' + \Delta C'),$$

where $C' = \text{relu}(C)$ ensures the canonical color remains non-negative. Finally, we employ the splatting algorithm (Zwicker et al. 2001) to render the target image $\mathcal{I}_i$.

Gaussian Attribute Compression

In achieve a compact representation for real-time streaming, we draw inspiration from work (Morgenstern et al. 2025) that sorts 3D Gaussians into 2D grids through the PLAS algorithm. We organize position, scale, rotation, opacity, and color into five attribute images and further organize E temporal features into E feature images. These feature images are then compressed into a feature video using traditional video codecs (e.g., H.264 and HEVC), thereby significantly reducing redundancy among temporal features, as shown in Fig. 2. Furthermore, our proposed representation inherently supports adaptive bit-rate transmission without requiring extra training, as feature video can be encoded and decoded using different bit-rates according to network conditions.

Consistency regularization For high compression efficiency, we adopt a method (Morgenstern et al. 2025) that applies a 2D Gaussian filter to the sorted 2D attribute images to smooth adjacent Gaussian attributes. The spatial smooth regularization, $\mathcal{L}_{\text{spatial}}$, is defined as the difference between the smoothed 2D images and the original 2D images. As for feature images, we introduce temporal consistency regularization. Specifically, we compute the difference between e_{i-1} and e_i , as well as between e_i and e_{i+1} . Then, we calculate the mean of these two differences as the temporal consistency regularization $\mathcal{L}_{\text{temporal}}$:

$$L_i = \text{huber}(e_{i-1} - e_i), \quad (8)$$

$$L_{i+1} = \text{huber}(e_i - e_{i+1}), \quad (9)$$

$$\mathcal{L}_{\text{temp}} = \text{mean}(L_i, L_{i+1}), \quad (10)$$

where huber is the Huber loss, which applies MSE to deviations when the delta below a threshold and MAE to deviations when the delta exceeds the threshold. This loss is well-suited for temporal consistency regularization. Since static

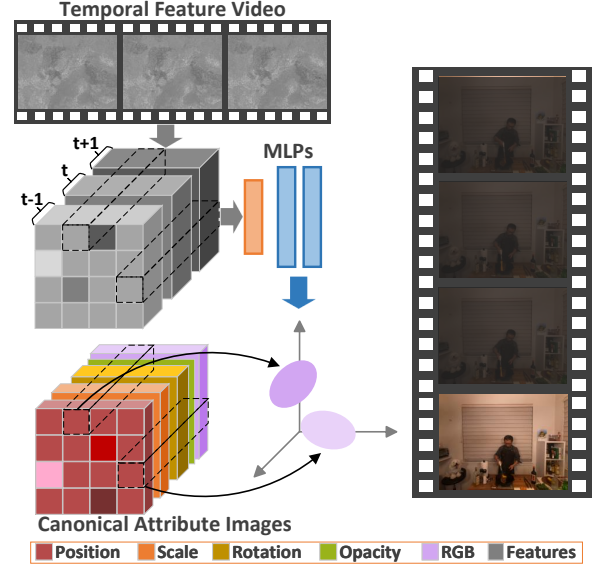


Figure 2: The 2D grid representation of our StreamSTGS. Canonical Gaussian attributes are compressed as 2D images and temporal features are encoded as a video for real-time streaming.

Gaussians exhibit similarity across different temporal features, MSE is used to preserve these feature values consistency to reduce the data size of feature video. In contrast, dynamic Gaussians show significant differences across different temporal features, and MAE is used to smooth these values while tolerating outliers. Applying MSE to such outliers would result in over-smoothing and degrade dynamic motion reconstruction.

Transformer-Guided Auxiliary Training

Though we design a sliding window scheme to learn local motion, learning global motions remains challenging due to the limited length of the sliding window. Inspired by TimeFormer (Jiang et al. 2024), which treats all timestamps as a sequence and utilizes Batch Attention (Hou, Yu, and Tao 2022) to learn robust global motions. However, they concatenate the position of canonical 3D Gaussians X and different timestamps t_i as input, which primarily enhances the learning of canonical 3D Gaussians rather than temporal features. In this paper, we use the output of the temporal MLP f_i as the input:

$$f'_i = \mathcal{F}(f_i, \gamma(t_i), \gamma(X)), \quad (11)$$

where $\mathcal{F}$ is the Transformer module. Note that $\gamma(t_i)$ and $\gamma(X)$ provide position encoding for the Transformer. $\gamma(t_i)$ encodes the position of the sliding window in temporal features, allowing the model to establish relationships among different sliding windows for global motion learning. $\gamma(X)$ provides the position of 3D Gaussians and helps transformer jointly learn the spatial-temporal features, especially in our fully decoupled spatial and temporal representation.

To tolerate with the loss of feature encoding, we add a small noise ε to the sliding window during training:

$$fe_i = \text{concat}(e_{i-1} + \varepsilon_1, e_i + \varepsilon_2, e_{i+1} + \varepsilon_3), \quad (12)$$

where ε_1 , ε_2 , and ε_3 are random sampled from a uniform distribution $\lambda \cdot \mathcal{N}(-0.5, 0.5)$.

Integrating the Transformer module into the StreamSTGS results in low FPS. Therefore, we employ a two-pass design, as illustrated in Fig. 1, where the deformation field is shared between the Gaussian Pass and the Auxiliary Pass. This design allows the global motion knowledge learned by Transformer to be transferred to our StreamSTGS representation. During inference, we can remove the Transformer module to maintain a high FPS. Additionally, we introduce a self-distillation loss $\mathcal{L}_{sd}$ to facilitate knowledge transfer:

$$\mathcal{L}_{sd} = \|f_i - f'_i\|_1. \quad (13)$$

Moreover, we find that the SSIM loss helps our StreamSTGS representation learn finer details. Thus, we use SSIM loss to supervise the prediction of the Auxiliary Pass $\mathcal{I}'_i$:

$$\mathcal{L}_t = SSIM(\mathcal{I}'_i - \mathcal{I}_i^{gt}). \quad (14)$$

Optimization

dynamic-aware density Existing Gaussian-based dynamic scene reconstruction methods typically use views from the first timestamp to generate a SFM point cloud via COLMAP (Schönerberger and Frahm 2016). However, subsequent timestamps lack enough prior points in dynamic areas. This limitation causes some 3D Gaussians in static areas attempt to clone and split to fit dynamic objects, resulting in an uneven distribution of 3D Gaussians overly dense in static areas and insufficient in dynamic areas. Consequently, the reconstruction quality in dynamic areas is poor. Thus, we propose a simple yet efficient method, namely dynamic-aware density, which applies an L1 loss to the entire predicted image but restricts the SSIM loss to dynamic pixels:

$$\begin{aligned} \mathcal{L}_c = & (1 - \beta) \cdot \|\mathcal{I}_i - \mathcal{I}_i^{gt}\|_1 \\ & + \beta \cdot SSIM(\mathcal{I}_i \cdot \mathcal{I}^{mask} - \mathcal{I}_i^{gt} \cdot \mathcal{I}^{mask}), \end{aligned} \quad (15)$$

where $\mathcal{I}^{mask}$ is the dynamic mask. We calculate the standard deviation across 30 frames per camera, identifying pixels as dynamic if their standard deviation exceeds a predefined threshold θ . Alternatively, dynamic masks can be obtained using SAM (Kirillov et al. 2023) or other models.

Pruning To limit the number of 3D Gaussians, we collect the predicted opacity O^i during training and compute its average value. If the average value falls below a given threshold, we prune these 3D Gaussians. This pruning operation is safe because we have collected the opacity of Gaussians across all timestamps within a GOP. To encourage low opacity for pruning, we introduce opacity regularization $\mathcal{L}_o$:

$$\mathcal{L}_o = \|O^i\|_1. \quad (16)$$

In summary, the total loss is:

$$\begin{aligned} \mathcal{L} = & \mathcal{L}_c + 0.2 \cdot \mathcal{L}_t + \mathcal{L}_{spatial} + \alpha_{temp} \cdot \mathcal{L}_{temp} \\ & + \alpha_o \cdot \mathcal{L}_o + \alpha_{sd} \cdot \mathcal{L}_{sd}. \end{aligned} \quad (17)$$

Gaussian relocate We limit the number of 3D Gaussians to approximately 150k to reduce data size. However, this upper bound may be reached during training. In such cases, all density operations, including prune, clone, and split, have to be halted, which inevitably leads to suboptimal model performance. Therefore, we introduce a relocate operation inspired by work (Kheradmand et al. 2024), which does not prune unnecessary Gaussians but instead moves them to more optimal position. This mechanism is particularly well-suited to our method, as it allows continuous optimization of the 3D Gaussians distribution without changing the total number of Gaussians.

Experiments and Results

Experimental Setup

We utilize two real-world multi-view dynamic scene datasets: 1) N3DV (Li et al. 2022b) dataset, which comprises six scenes captured by 18 to 21 cameras at a resolution of 2704×2028 and a frame rate of 30 FPS. Traditionally, we downsample the resolution to 1352×1014 . 2) MeetRoom (Li et al. 2022a) dataset, which uses 12 cameras to capture three scenes at a resolution of 1280×720 and a frame rate of 30 FPS. Following the approach of 4DGaussians (Wu et al. 2024a), we use COLMAP (Schönerberger and Frahm 2016) to generate initial point clouds.

Implementation. All experiments are conducted on a RTX A6000 GPU. We first train a coarse 3DGS model using all multi-view frames about 3000 iterations with a batch size of 2. Then, a refined model is trained for each GOP with a size of 60. Each GOP is trained for 12000 and 7000 iterations for N3DV and MeetRoom datasets, respectively. The noise parameter λ is set to 0.001, as we observe that the loss of temporal features after compression is about 0.0002. Additionally, α_{temp} , α_o , and α_{sd} are set to 1.0, 0.01, and 0.005, respectively. Dynamic-aware density is applied after 5000 iteration for N3DV dataset and 3000 iteration for MeetRoom dataset. More implementation details are provided in the **Appendix**.

Quantitative Comparisons

We conduct a comprehensive quantitative comparison with state-of-the-art streamable FVV methods, including TeTriRF (Wu et al. 2024b), 3DGStream (Sun et al. 2024), VideoGS (Wang et al. 2024c), HiCoM (Gao et al. 2024), and 4DGC (Hu et al. 2025a). First, we evaluate reconstruction quality by PSNR, SSIM, and LPIPS(VGG). As presented results in Table 1 and Table 2, our StreamSTGS achieves the best reconstruction quality on both N3DV and MeetRoom datasets. In the meanwhile, the average frame size of StreamSTGS is only 170KB, which is 4X smaller than that of GS-based methods. Though TeTriRF achieve the smallest frame size, its reconstruction quality and FPS are inferior to StreamSTGS. Furthermore, since users may not start watching FVV from the beginning, they must wait for previous n frames to be decoded or for keyframes with large amounts of data to be transmitted. Therefore, we also compare Key-Frame (K.F.) size and decoding delay. As shown in

Method	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Storage (KB) $\downarrow$	K.F. Size (MB) $\downarrow$	Decoding (ms) $\downarrow$	Render (ms) $\downarrow$	FPS $\uparrow$	Train (s) $\downarrow$	VBR
TeTriRF (Wu et al. 2024b)	30.07	0.900	0.299	65.89	2.03	149	652	1.53	32	✓
3DGStream (Sun et al. 2024)	30.73	0.935	0.147	8204	42.22	7×n	14	72	17	×
VideoGS (Wang et al. 2024c)	27.45	0.871	0.213	932.9	3.24	48	7	21	143	✓
HiCoM (Gao et al. 2024)	31.32	0.939	0.147	10704	83.35	0	6	163	10	×
4DGC (Hu et al. 2025a)	31.52	0.941	0.143	784	21.94	2.5×n	12	78.6	62	×
Ours	32.30	0.943	0.147	173.6	3.86	8	10	100	67	✓

Table 1: Quantitative comparison on N3DV dataset. ‘VBR’ indicates variable bitrate transmission.

Method	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Storage (KB) $\downarrow$	K.F. Size (MB) $\downarrow$	Decoding (ms) $\downarrow$	Render (ms) $\downarrow$	FPS $\uparrow$	Train (s) $\downarrow$	VBR
3DGStream (Sun et al. 2024)	26.41	0.90	0.24	4108	18	3×n	8.23	121	11	×
HiCoM (Gao et al. 2024)	26.69	0.90	0.23	5535	42	0	3.64	275	6	×
4DGC (Hu et al. 2025a)	27.11	0.91	0.23	1196	11	2×n	9.06	110	60	×
Ours	27.41	0.92	0.21	142	2.8	6	7.93	126	29	✓

Table 2: Quantitative comparison on MeetRoom dataset. ‘VBR’ indicates variable bitrate transmission.

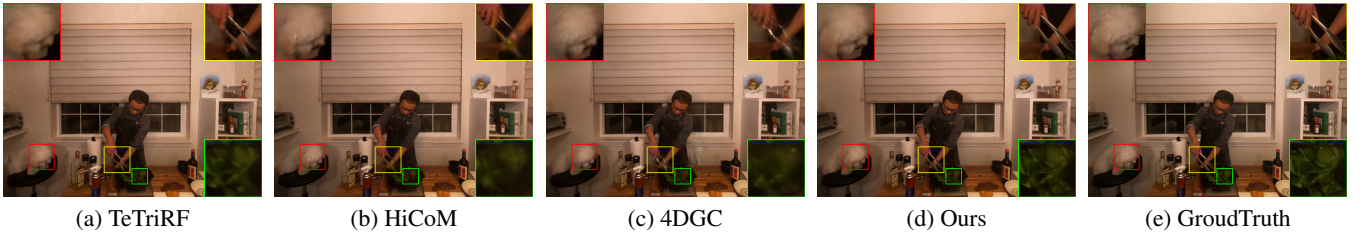


Figure 3: Qualitative comparison of ours with the benchmark methods on *Cook Spinach* scene of N3DV dataset.

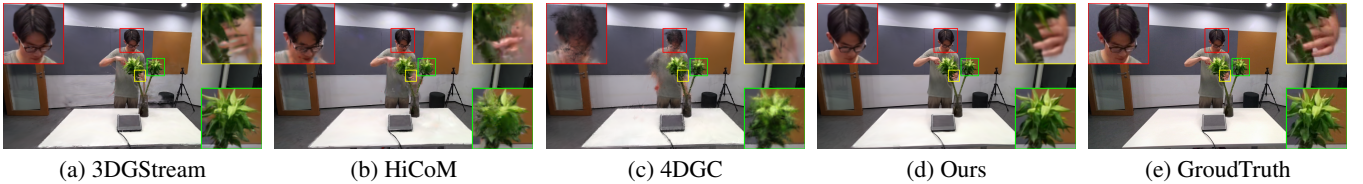


Figure 4: Qualitative comparison of ours with the benchmark methods on *Trimming* scene of MeetRoom dataset.

Table 1 and Table 2, our StreamSTGS demonstrates competitive performance in these metrics. More quantitative results are provided in the **Appendix**.

Qualitative Comparisons

Fig. 3 and Fig. 4 present the qualitative results of our StreamSTGS compared to benchmark methods. It is evident that StreamSTGS achieves significant improvements in dynamic areas, owing to the proposed Transformer-guided auxiliary module that enhances the learning of motion. Notably, in the area of complex human interaction, such as *yellow* region in both Fig. 3 and Fig. 4, our StreamSTGS accurately reconstructs hands and objects, whereas comparative methods yield blurred results. This improvement can be attributed to our dynamic-aware density strategy, which

directs a higher concentration of 3D Gaussians toward dynamic areas. These Gaussians are subsequently optimized to model motion patterns more effectively through Transformer module. Besides, our StreamSTGS successfully reconstructs high-light effect, demonstrating that our simplified color model not only reduces storage but also preserves reconstruction quality.

Ablations

Compression QP. We evaluate the performance of our StreamSTGS under different compression QP parameters using libx265, with results presented in Table. 3. It can be observed that setting QP to 20 achieves an optimal trade-off between reconstruction quality and storage. Even when the QP is set to 28 or 32, our StreamSTGS still outperforms

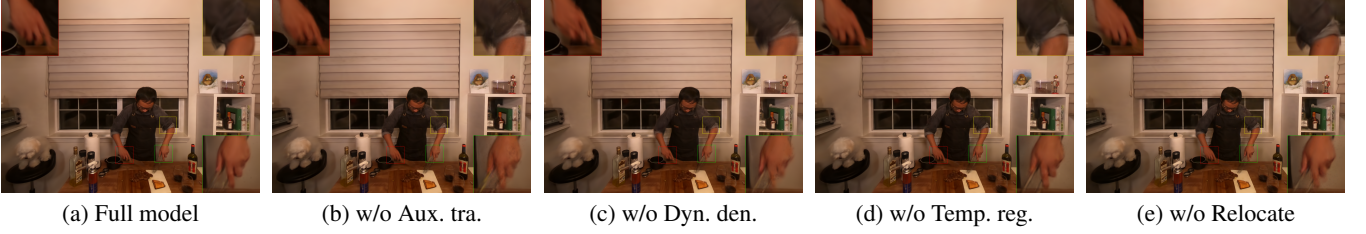


Figure 5: Ablation experiments on *Cut Roasted Beef* scene of N3DV dataset.

QP	N3DV		MeetRoom	
	PSNR $\uparrow$	Storage $\downarrow$	PSNR $\uparrow$	Storage $\downarrow$
16	32.36	247.52	27.46	208.71
20	32.30	173.59	27.41	142.53
24	32.15	121.97	27.32	97.06
28	31.68	87.95	27.02	68.25
32	30.76	68.35	26.46	52.34

Table 3: Performance under different compression QP.

Sliding Window Size	PSNR	SSIM	Storage	K.F. Size
W = 1	32.01	0.941	298.06	3.92
W = 3	32.30	0.944	173.59	3.86
W = 5	32.26	0.944	176.05	3.86

Table 4: Performance under different sizes of sliding window on N3DV dataset.

	PSNR	SSIM	Storage	Train
GOP-30	32.32	0.944	228.7	133
GOP-60(Ours)	32.30	0.943	173.6	67
GOP-100	32.06	0.942	161.9	40

Table 5: Ablation of GOP length.

benchmark methods in reconstruction quality while significantly reducing storage size.

Sliding Window. As shown in Table 4, we evaluate the performance of our StreamSTGS under different sliding window sizes W . When the sliding window scheme is removed (i.e., $W = 1$), both the reconstruction quality and the storage efficiency degrade, as the relationships between adjacent temporal features cannot be captured and optimized during training. Note that increasing the size of sliding window to 5 does not provide significant benefits, as learning motions across five frames is challenging.

GOP Length. We also conduct ablation experiments on the GOP length. We set the GOP length as 30, 60, and 100, and the results are shown in the Table 5. When GOP is 60, a good tradeoff is achieved in terms of quality, storage size,

Components	PSNR	Storage	Train
Full model	32.30	173.59	67
w/o Auxiliary training	31.99	174.51	29
w/o Dynamic density	32.07	114.15	69
w/o Temporal reg.	32.23	319.48	64
w/o Gaussian relocate	32.11	169.70	68

Table 6: Effect of various components ablated on N3DV.

and training time. Since our method does not require training multiple models for different bitrates, sacrificing training time for higher performance is worthwhile. Existing methods exceed 60s to train 3-6 models for variable bitrates.

Key Components. We ablate four key components of our StreamSTGS as shown in Table. 6. In the second row, we do not apply Transformer-guided auxiliary training, which significantly reduced training costs. However, this leads to decreased reconstruction quality and motion blur, as illustrated in Fig. 5(b), due to the difficulty in learning global motions. In the third row, we disable the dynamic-aware density strategy, which causes blurry motions in dynamic areas as shown in Fig. 5(c). Correspondingly, the reduced storage in this case is because static Gaussians exhibit more temporal consistency, which improves compression efficiency. In the fourth row, we omit temporal regularization, which substantially increase storage size without any improvement in reconstruction quality as shown in Fig. 5(d). In the fifth row, we remove the Gaussian relocate strategy as demonstrated in Fig. 5(e), which prevents unnecessary Gaussians from being repositioned effectively, resulting in local optima.

Conclusion and Limitation

This paper introduces a novel streamable FVV representation, StreamSTGS, which represents canonical 3D Gaussians as 2D images and temporal features as a video, thereby resulting in a compact enough representation capable of meeting real-time requirements. Moreover, a Transformer-guided auxiliary training strategy is proposed to improve the learning of global motions. Additionally, several key improvements, such as sliding window-based temporal feature aggregation, dynamic-aware density, and Gaussian relocate strategy, collectively contribute to the superior performance of StreamSTGS. However, each 3D Gaussians in StreamSTGS needs temporal features, but some 3D Gaus-

sians maintain static within a GOP. Therefore, by categorizing 3D Gaussians into static and dynamic sets and assigning temporal features exclusively to dynamic Gaussians, we can further reduce storage and improve FPS.

References

- Attal, B.; Huang, J.-B.; Richardt, C.; Zollhoefer, M.; Kopf, J.; O’Toole, M.; and Kim, C. 2023. Hyperreel: High-fidelity 6-dof video with ray-conditioned sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16610–16620.
- Bae, J.; Kim, S.; Yun, Y.; Lee, H.; Bang, G.; and Uh, Y. 2025. Per-gaussian embedding-based deformation for deformable 3d gaussian splatting. In *European Conference on Computer Vision*, 321–335. Springer.
- Cao, A.; and Johnson, J. 2023. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 130–141.
- Chen, A.; Xu, Z.; Geiger, A.; Yu, J.; and Su, H. 2022. Tensorf: Tensorial radiance fields. In *European conference on computer vision*, 333–350. Springer.
- Du, Y.; Zhang, Y.; Yu, H.-X.; Tenenbaum, J. B.; and Wu, J. 2021. Neural radiance flow for 4d view synthesis and video processing. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 14304–14314. IEEE Computer Society.
- Duan, Y.; Wei, F.; Dai, Q.; He, Y.; Chen, W.; and Chen, B. 2024. 4d-rotor gaussian splatting: towards efficient novel view synthesis for dynamic scenes. In *ACM SIGGRAPH 2024 Conference Papers*, 1–11.
- Duisterhof, B. P.; Mandi, Z.; Yao, Y.; Liu, J.-W.; Shou, M. Z.; Song, S.; and Ichnowski, J. 2023. Md-splatting: Learning metric deformation from 4d gaussians in highly deformable scenes. *arXiv preprint arXiv:2312.00583*.
- Fan, C.-D.; Chang, C.-W.; Liu, Y.-R.; Lee, J.-Y.; Huang, J.-L.; Tseng, Y.-C.; and Liu, Y.-L. 2025. Spectromotion: Dynamic 3d reconstruction of specular scenes. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 21328–21338.
- Fang, J.; Yi, T.; Wang, X.; Xie, L.; Zhang, X.; Liu, W.; Nießner, M.; and Tian, Q. 2022. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, 1–9.
- Fridovich-Keil, S.; Meanti, G.; Warburg, F. R.; Recht, B.; and Kanazawa, A. 2023. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12479–12488.
- Gao, C.; Saraf, A.; Kopf, J.; and Huang, J.-B. 2021. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5712–5721.
- Gao, Q.; Meng, J.; Wen, C.; Chen, J.; and Zhang, J. 2024. HiCoM: Hierarchical Coherent Motion for Streamable Dynamic Scene with 3D Gaussian Splatting. *Advances in Neural Information Processing Systems*.
- Girish, S.; Li, T.; Mazumdar, A.; Shrivastava, A.; Luebke, D.; and De Mello, S. 2024. QUEEN: QUAntized Efficient ENcoding of Dynamic Gaussians for Streaming Free-viewpoint Videos. *Advances in Neural Information Processing Systems*.
- Guo, X.; Sun, J.; Dai, Y.; Chen, G.; Ye, X.; Tan, X.; Ding, E.; Zhang, Y.; and Wang, J. 2023. Forward flow for novel view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 16022–16033.
- Hou, Z.; Yu, B.; and Tao, D. 2022. Batchformer: Learning to explore sample relationships for robust representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7256–7266.
- Hu, Q.; Zheng, Z.; Zhong, H.; Fu, S.; Song, L.; Zhai, G.; Wang, Y.; et al. 2025a. 4DGC: Rate-Aware 4D Gaussian Compression for Efficient Streamable Free-Viewpoint Video. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- Hu, Q.; Zhong, H.; Zheng, Z.; Zhang, X.; Cheng, Z.; Song, L.; Zhai, G.; and Wang, Y. 2025b. VRVVC: Variable-Rate NeRF-Based Volumetric Video Compression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 3563–3571.
- Huang, Y.-H.; Sun, Y.-T.; Yang, Z.; Lyu, X.; Cao, Y.-P.; and Qi, X. 2024. SC-GS: Sparse-controlled gaussian splatting for editable dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4220–4230.
- Jiang, D.; Ke, Z.; Zhou, X.; Hou, Z.; Yang, X.; Hu, W.; Qiu, T.; and Guo, C. 2024. TimeFormer: Capturing Temporal Relationships of Deformable 3D Gaussians for Robust Reconstruction. *arXiv preprint arXiv:2411.11941*.
- Katsumata, K.; Vo, D. M.; and Nakayama, H. 2025. A compact dynamic 3d gaussian representation for real-time dynamic view synthesis. In *European Conference on Computer Vision*, 394–412. Springer.
- Kerbl, B.; Kopanas, G.; Leimkühler, T.; and Drettakis, G. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM transactions on graphics (TOG)*, 42(4): 139–1.
- Kheradmand, S.; Rebain, D.; Sharma, G.; Sun, W.; Tseng, J.; Isack, H.; Kar, A.; Tagliasacchi, A.; and Yi, K. M. 2024. 3D Gaussian Splatting as Markov Chain Monte Carlo. *arXiv preprint arXiv:2404.09591*.
- Kim, M.; Lim, J.; and Han, B. 2024. 4D Gaussian Splatting in the Wild with Uncertainty-Aware Regularization. *Advances in Neural Information Processing Systems*.
- Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A. C.; Lo, W.-Y.; et al. 2023. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4015–4026.

- Kwak, S.; Kim, J.; Jeong, J. Y.; Cheong, W.-S.; Oh, J.; and Kim, M. 2025. MoDec-GS: Global-to-Local Motion Decomposition and Temporal Interval Adjustment for Compact Dynamic 3D Gaussian Splatting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*.
- Lee, J.; Won, C.-Y.; Jung, H.; Bae, I.; and Jeon, H.-G. 2024. Fully Explicit Dynamic Gaussian Splatting. *Advances in Neural Information Processing Systems*.
- Lei, J.; Weng, Y.; Harley, A. W.; Guibas, L.; and Danilidis, K. 2025. Mosca: Dynamic gaussian fusion from casual videos via 4d motion scaffolds. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 6165–6177.
- Li, H.; Li, S.; Gao, X.; Batuer, A.; Yu, L.; and Liao, Y. 2025. GIFStream: 4D Gaussian-based Immersive Video with Feature Stream. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 21761–21770.
- Li, L.; Shen, Z.; Wang, Z.; Shen, L.; and Tan, P. 2022a. Streaming radiance fields for 3d video synthesis. *Advances in Neural Information Processing Systems*, 35: 13485–13498.
- Li, T.; Slavcheva, M.; Zollhoefer, M.; Green, S.; Lassner, C.; Kim, C.; Schmidt, T.; Lovegrove, S.; Goesele, M.; Newcombe, R.; et al. 2022b. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5521–5531.
- Li, Z.; Chen, Z.; Li, Z.; and Xu, Y. 2024. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8508–8520.
- Li, Z.; Niklaus, S.; Snavely, N.; and Wang, O. 2021. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6498–6508.
- Lin, Y.; Dai, Z.; Zhu, S.; and Yao, Y. 2024. Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 21136–21145.
- Liu, Y.-L.; Gao, C.; Meuleman, A.; Tseng, H.-Y.; Saraf, A.; Kim, C.; Chuang, Y.-Y.; Kopf, J.; and Huang, J.-B. 2023. Robust dynamic radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13–23.
- Lou, A.; Planche, B.; Gao, Z.; Li, Y.; Luan, T.; Ding, H.; Chen, T.; Noble, J.; and Wu, Z. 2024. DaReNeRF: Direction-aware Representation for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5031–5042.
- Lu, J.; Deng, J.; Zhu, R.; Liang, Y.; Yang, W.; Zhang, T.; and Zhou, X. 2024a. DN-4DGS: Denoised Deformable Network with Temporal-Spatial Aggregation for Dynamic Scene Rendering. *Advances in Neural Information Processing Systems*.
- Lu, Z.; Guo, X.; Hui, L.; Chen, T.; Yang, M.; Tang, X.; Zhu, F.; and Dai, Y. 2024b. 3d geometry-aware deformable gaussian splatting for dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8900–8910.
- Luiten, J.; Kopanas, G.; Leibe, B.; and Ramanan, D. 2024. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *2024 International Conference on 3D Vision (3DV)*, 800–809. IEEE.
- Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1): 99–106.
- Morgenstern, W.; Barthel, F.; Hilsmann, A.; and Eisert, P. 2025. Compact 3d scene representation via self-organizing gaussian grids. In *European Conference on Computer Vision*, 18–34. Springer.
- Müller, T.; Evans, A.; Schied, C.; and Keller, A. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4): 1–15.
- Park, J.; Bui, M.-Q. V.; Bello, J. L. G.; Moon, J.; Oh, J.; and Kim, M. 2025. Splines: Robust motion-adaptive spline for real-time dynamic 3d gaussians from monocular video. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 26866–26875.
- Park, K.; Sinha, U.; Barron, J. T.; Bouaziz, S.; Goldman, D. B.; Seitz, S. M.; and Martin-Brualla, R. 2021a. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5865–5874.
- Park, K.; Sinha, U.; Hedman, P.; Barron, J. T.; Bouaziz, S.; Goldman, D. B.; Martin-Brualla, R.; and Seitz, S. M. 2021b. HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields. *ACM transactions on graphics (TOG)*, 40(6).
- Park, S.; Son, M.; Jang, S.; Ahn, Y. C.; Kim, J.-Y.; and Kang, N. 2023. Temporal interpolation is all you need for dynamic neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4212–4221.
- Schönberger, J. L.; and Frahm, J.-M. 2016. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Shao, R.; Zheng, Z.; Tu, H.; Liu, B.; Zhang, H.; and Liu, Y. 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.
- Shaw, R.; Nazarczuk, M.; Song, J.; Moreau, A.; Catley-Chandar, S.; Dharmo, H.; and Pérez-Pellitero, E. 2024. Swings: sliding windows for dynamic 3D gaussian splatting. In *European Conference on Computer Vision*. Springer.
- Song, L.; Chen, A.; Li, Z.; Chen, Z.; Chen, L.; Yuan, J.; Xu, Y.; and Geiger, A. 2023. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 29(5): 2732–2742.

- Sun, J.; Jiao, H.; Li, G.; Zhang, Z.; Zhao, L.; and Xing, W. 2024. 3dstream: On-the-fly training of 3d gaussians for efficient streaming of photo-realistic free-viewpoint videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 20675–20685.
- Tian, F.; Du, S.; and Duan, Y. 2023. Mononerf: Learning a generalizable dynamic radiance field from monocular videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 17903–17913.
- Wan, D.; Lu, R.; and Zeng, G. 2024. Superpoint Gaussian Splatting for Real-Time High-Fidelity Dynamic Scene Reconstruction. In *Forty-first International Conference on Machine Learning*.
- Wang, F.; Chen, Z.; Wang, G.; Song, Y.; and Liu, H. 2024a. Masked space-time hash encoding for efficient dynamic scene reconstruction. *Advances in Neural Information Processing Systems*, 36.
- Wang, F.; Tan, S.; Li, X.; Tian, Z.; Song, Y.; and Liu, H. 2023a. Mixed neural voxels for fast multi-view video synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 19706–19716.
- Wang, L.; Hu, Q.; He, Q.; Wang, Z.; Yu, J.; Tuytelaars, T.; Xu, L.; and Wu, M. 2023b. Neural residual radiance fields for streamably free-viewpoint videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 76–87.
- Wang, L.; Yao, K.; Guo, C.; Zhang, Z.; Hu, Q.; Yu, J.; Xu, L.; and Wu, M. 2024b. VideoRF: Rendering Dynamic Radiance Fields as 2D Feature Video Streams. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 470–481.
- Wang, P.; Liu, Y.; Chen, Z.; Liu, L.; Liu, Z.; Komura, T.; Theobalt, C.; and Wang, W. 2023c. F2-nerf: Fast neural radiance field training with free camera trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4150–4159.
- Wang, P.; Zhang, Z.; Wang, L.; Yao, K.; Xie, S.; Yu, J.; Wu, M.; and Xu, L. 2024c. V³: Viewing Volumetric Videos on Mobiles via Streamable 2D Dynamic Gaussians. *ACM Transactions on Graphics (TOG)*, 43(6): 1–13.
- Wu, G.; Yi, T.; Fang, J.; Xie, L.; Zhang, X.; Wei, W.; Liu, W.; Tian, Q.; and Wang, X. 2024a. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 20310–20320.
- Wu, M.; Wang, Z.; Kourois, G.; and Tuytelaars, T. 2024b. TeTriRF: Temporal Tri-Plane Radiance Fields for Efficient Free-Viewpoint Video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6487–6496.
- Xian, W.; Huang, J.-B.; Kopf, J.; and Kim, C. 2021. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9421–9431.
- Xu, J.; Fan, Z.; Yang, J.; and Xie, J. 2024. Grid4D: 4D Decomposed Hash Encoding for High-fidelity Dynamic Gaussian Splatting. *Advances in Neural Information Processing Systems*.
- Yan, J.; Peng, R.; Tang, L.; and Wang, R. 2024. 4D Gaussian Splatting with Scale-aware Residual Field and Adaptive Optimization for Real-time rendering of temporally complex dynamic scenes. In *Proceedings of the 32nd ACM International Conference on Multimedia*, 7871–7880.
- Yan, Z.; Li, C.; and Lee, G. H. 2023. Nerf-ds: Neural radiance fields for dynamic specular objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8285–8295.
- Yang, J.-W.; Sun, J.-M.; Yang, Y.-L.; Yang, J.; Shan, Y.; Cao, Y.-P.; and Gao, L. 2025. DMiT: Deformable Mipmapped Tri-Plane Representation for Dynamic Scenes. In *European Conference on Computer Vision*, 436–453. Springer.
- Yang, Z.; Gao, X.; Zhou, W.; Jiao, S.; Zhang, Y.; and Jin, X. 2024a. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 20331–20341.
- Yang, Z.; Yang, H.; Pan, Z.; and Zhang, L. 2024b. Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting. In *International Conference on Learning Representations (ICLR)*.
- Yin, D.; Shi, J.; Zhang, M.; Huang, Z.; Liu, J.; and Dong, F. 2024. FSVFG: Towards Immersive Full-Scene Volumetric Video Streaming with Adaptive Feature Grid. In *Proceedings of the 32nd ACM International Conference on Multimedia*, 11089–11098.
- Yu, A.; Li, R.; Tancik, M.; Li, H.; Ng, R.; and Kanazawa, A. 2021. Plenotrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5752–5761.
- Zhan, Y.; Li, Z.; Niu, M.; Zhong, Z.; Nobuhara, S.; Nishino, K.; and Zheng, Y. 2024. KFD-NeRF: Rethinking Dynamic NeRF with Kalman Filter. In *European Conference on Computer Vision*. Springer.
- Zhang, Z.; Lu, G.; Liang, H.; Cheng, Z.; Tang, A.; and Song, L. 2024. Rate-aware Compression for NeRF-based Volumetric Video. In *Proceedings of the 32nd ACM International Conference on Multimedia*, 3974–3983.
- Zhao, B.; Li, Y.; Sun, Z.; Zeng, L.; Shen, Y.; Ma, R.; Zhang, Y.; Bao, H.; and Cui, Z. 2024. Gaussianprediction: Dynamic 3d gaussian prediction for motion extrapolation and free view synthesis. In *ACM SIGGRAPH 2024 Conference Papers*, 1–12.
- Zheng, Z.; Zhong, H.; Hu, Q.; Zhang, X.; Song, L.; Zhang, Y.; and Wang, Y. 2024. HPC: Hierarchical Progressive Coding Framework for Volumetric Video. In *Proceedings of the 32nd ACM International Conference on Multimedia*, 7937–7946.
- Zhu, R.; Liang, Y.; Chang, H.; Deng, J.; Lu, J.; Yang, W.; Zhang, T.; and Zhang, Y. 2024. Motiongs: Exploring explicit motion guidance for deformable 3d gaussian splatting. *Advances in Neural Information Processing Systems*.

Zwicker, M.; Pfister, H.; Van Baar, J.; and Gross, M. 2001. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 371–378.

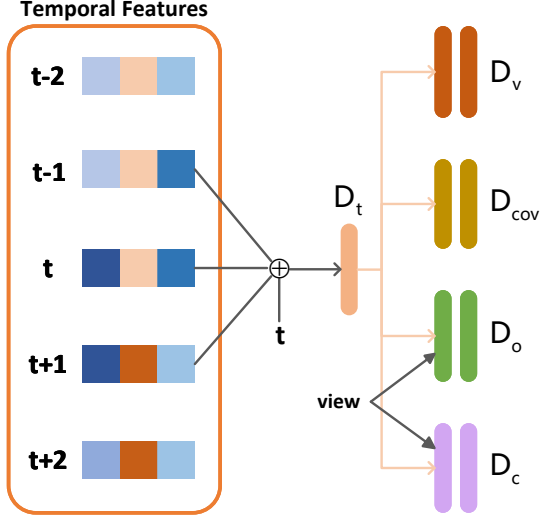


Figure 6: The deformation field of our StreamSTGS representation.

Appendix

More Implementation Details

Compression. Gaussian attribute images are stored in lossless JPEG XL format, while temporal feature images are encoded as a video using libx265 with the default quantization parameter $qp = 20$. The channel of temporal features $\mathcal{E}$ is set to 16, allowing the reshaping of its dimensions from $W \times H \times 16$ to $4W \times 4H$ to ensure compatibility with existing video codecs. Rotation R is clipped to the range $[-1, 2]$, opacity O and color C are clipped to the range $[-4, 4]$ and $[0, 4]$, respectively. These attributes, along with position X and temporal features $\mathcal{E}$, are then normalized to the range $[0, 1]$. For smaller image size, rotation is quantized with $q_r = 2^7$, and scale and opacity are quantized with $q_o = q_s = 2^6$. Since position X and scale S are highly dependent on scene size and temporal features are content-related, no quantization is applied to these attributes.

Deformation field. The network architecture of our deformation field is presented as Fig. 6. The temporal MLP comprises a single linear layer with 64 neurons, followed by a $Tanh$ activation function. The resulting output serves as the input for four Gaussian attribute prediction MLPs, i.e., D_v , D_{cov} , D_o , and D_c . Each of these MLPs consists of two linear layers, each containing 64 neurons. Bias are disabled for all MLPs. The learning rate for the canonical Gaussian attributes is same to that used in the original 3DGS. The temporal features are trained with a learning rate of 0.0025. Furthermore, the learning rates for the deformation field networks are as follows: D_t from 0.0025 to 0.000025, D_v from 0.005 to 0.00005, D_{cov} at 0.04, D_o from 0.002 to 0.00002, and D_c from 0.008 to 0.00005.

Auxiliary training module. Our transformer module consists of two *nn.TransformerEncoderLayer* and a single linear

Scene	GOP1	GOP2	GOP3	GOP4	GOP5	Average
PSNR						
Coffee Martini	28.70	29.10	28.83	28.73	28.77	28.83
Cook Spinach	33.70	33.78	33.49	33.14	33.28	33.48
Cut Roasted Beef	33.82	33.92	33.97	33.72	33.38	33.76
Flame Salmon	29.47	29.37	29.34	29.43	29.37	29.40
Flame Steak	34.14	33.95	33.97	33.69	33.55	33.86
Sear Steak	34.47	34.65	34.31	34.29	34.64	34.47
SSIM						
Coffee Martini	0.91	0.92	0.91	0.91	0.91	0.91
Cook Spinach	0.96	0.96	0.96	0.95	0.95	0.95
Cut Roasted Beef	0.96	0.96	0.96	0.96	0.95	0.96
Flame Salmon	0.92	0.92	0.92	0.92	0.92	0.92
Flame Steak	0.96	0.96	0.96	0.96	0.96	0.96
Sear Steak	0.96	0.96	0.96	0.96	0.96	0.96
LPIPS						
Coffee Martini	0.160	0.160	0.162	0.162	0.160	0.161
Cook Spinach	0.145	0.146	0.148	0.153	0.149	0.148
Cut Roasted Beef	0.146	0.143	0.148	0.146	0.154	0.148
Flame Salmon	0.160	0.155	0.157	0.156	0.153	0.156
Flame Steak	0.134	0.135	0.139	0.140	0.138	0.137
Sear Steak	0.134	0.133	0.133	0.135	0.134	0.134

Table 7: Per-GOP Metrics for the N3DV dataset.

layer. Each *nn.TransformerEncoderLayer* have two heads, hidden layers with 64 neurons, and $Tanh$ activation function. A $Tanh$ activation function is also applied to the output of the linear layer. The learning rate for the transformer is from 0.002 to 0.00001. To avoid the impact of timestamp length on positional encoding, we normalize it by the GOP length, e.g., timestamps within $[0, 59]$ or $[60, 119]$ are normalized into $[0, 1]$.

Additional Quantitative Results

Table 8 presents a comprehensive comparisons against additional dynamic scene reconstruction baselines on the N3DV dataset, representing a superset of the results reported in Table 1 of the main paper. Methods marked with * in Table 8 denote our re-implementation, using our dataset preparation and sparse point cloud generation based on their official public codes. Owing to a bit performance variations stem from differences in dataset preparation and sparse cloud point generation methods. For example, image undistortion, adding depth information from pre-trained depth prediction model, or merging of per-frame point clouds into a unified initial point cloud, instead of relying solely on the first frame’s point cloud, can lead to performance improvements.

Per-scene Quantitative Results

In addition to the average quantitative results over the full datasets of N3DV and MeetRoom in Table 8, we show results for each scene in both the N3DV and MeetRoom datasets of various metrics, including PSNR, SSIM, LPIPS(VGG), storage(frame size), FPS, decoding time, training time in Table 9 and Table 10.

Per-scene Qualitative Results

In addition to the qualitative results in the paper, we show all scenes results in Fig. 7 through Fig. 13. We have incorporated results of 3DGStream (Sun et al.

Scene	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Storage (MB) $\downarrow$	FPS $\uparrow$	Training (s) $\downarrow$	Streamable/ Variable bitrate
HyperReel (Attal et al. 2023)	31.10	0.928	-	1.2	2.0	104	×/×
HexPlanes (Cao and Johnson 2023)	31.70	-	-	0.8	0.21	144	×/×
KPlanes (Fridovich-Keil et al. 2023)	31.63	-	-	1.0	0.15	48	×/×
Mix Voxels (Wang et al. 2023a)	30.81	-	-	1.7	38	16	×/×
NeRFPlayer (Song et al. 2023)	30.69	0.932	0.209	17.1	0.05	72	✓/×
StreamRF (Li et al. 2022a)	30.68	-	-	31.4	8.3	15	✓/×
TeTriRF* (Wu et al. 2024b)	30.07	0.90	0.299	0.06	1.5	32	✓/✓
4DGS (Yang et al. 2024b)	32.01	-	-	29	30	76	×/×
4DGaussians* (Wu et al. 2024a)	31.11	0.938	0.141	0.11	30	9	×/×
STGS (Li et al. 2024)	31.62	0.946	-	0.60	140	30	×/×
4D-Rotor (Duan et al. 2024)	31.62	0.94	-	-	277	12	×/×
DN-4DGS (Lu et al. 2024a)	32.02	0.944	-	0.37	15	10	×/×
Ex4DGS (Lee et al. 2024)	32.11	0.94	-	0.38	-	-	×/×
3DGStream* (Sun et al. 2024)	30.73	0.935	0.147	8.0	72	17	✓/×
VideoGS* (Wang et al. 2024c)	27.45	0.871	0.214	0.91	21	143	✓/✓
HiCoM* (Gao et al. 2024)	31.32	0.939	0.147	10.5	163	10	✓/×
QUEEN (Girish et al. 2024)	32.19	0.946	-	0.75	-	-	✓/×
4DGC* (Hu et al. 2025a)	31.52	0.941	0.143	0.77	79	62	✓/×
Ours	32.30	0.943	0.147	0.17	100	67	✓/✓

Table 8: Quantitative Comparisons on the N3DV Datasets. * refers to our re-implementation with our dataset preparation and sparse point cloud generation based on their official open-sourced codes.

Scene	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Storage (KB) $\downarrow$	FPS $\uparrow$	Decoding (ms) $\downarrow$	Training (s) $\downarrow$
Coffee Martini	28.83	0.91	0.161	164	105	8.7	69
Cook Spinach	33.48	0.95	0.148	183	98	8.7	64
Cut Roasted Beef	33.76	0.96	0.148	188	102	9.0	65
Flame Salmon	29.4	0.92	0.156	148	98	8.0	68
Flame Steak	33.86	0.96	0.137	179	97	8.6	68
Sear Steak	34.47	0.96	0.134	180	99	8.9	66

Table 9: Per-scene metrics for the N3DV dataset.

Scene	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Storage (KB) $\downarrow$	FPS $\uparrow$	Decoding (ms) $\downarrow$	Training (s) $\downarrow$
Discussion	27.74	0.91	0.211	161	125	7.2	29
Trimming	27.65	0.92	0.213	122	139	5.8	29
Vrheadset	26.83	0.91	0.223	144	114	6.7	30

Table 10: Per-scene metrics for the MeetRoom dataset.

2024) and VideoGS (Wang et al. 2024c) for comparison. VideoGS (Wang et al. 2024c) also uses video codecs to compress each channel of all Gaussian attributes, generating 23 feature videos and incurring long decoding latency. Furthermore, direct compression of Gaussian attributes leads to significant performance degradation in both dynamic areas

and static backgrounds. Instead, our method encodes Gaussian spatial attributes using lossless JPEG-XL and temporal features as a video. To mitigate decoding latency, five spatial images of the subsequent GOP are pre-decoded. Consequently, only the temporal feature video requires real-time decoding. Although compression of temporal features intro-

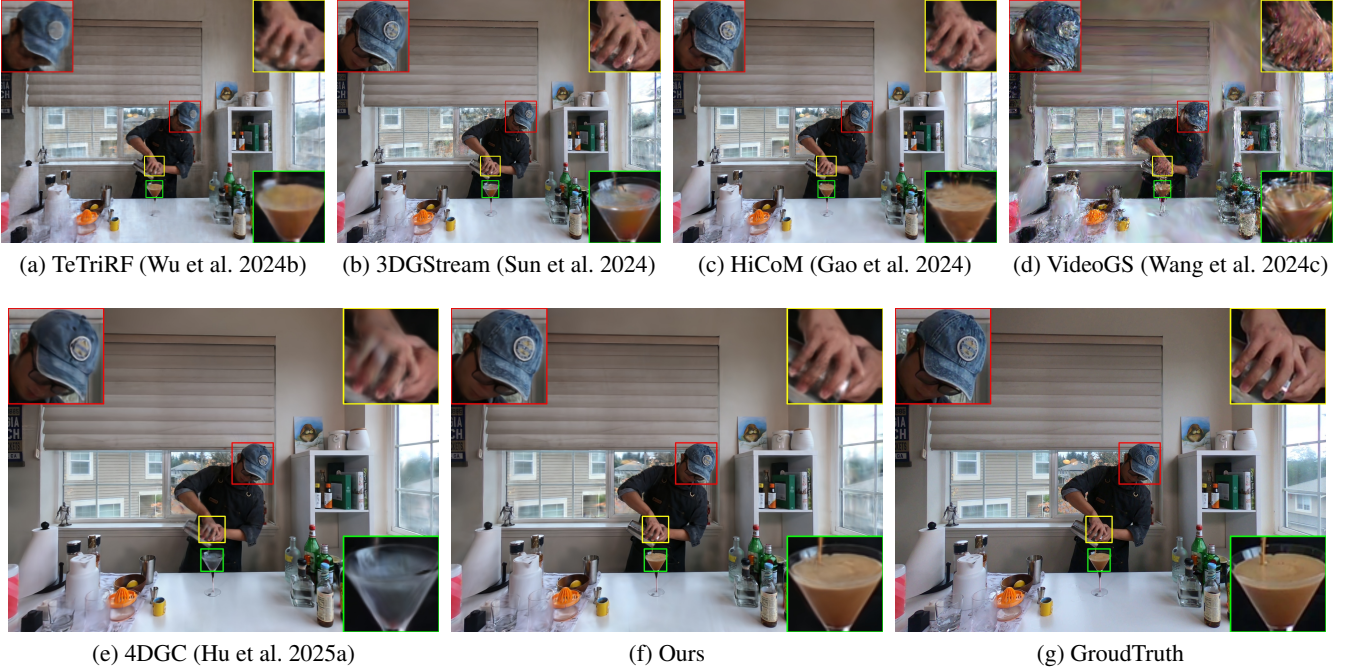


Figure 7: Qualitative comparison of ours with the benchmark methods on *Coffee Martini* scene of N3DV dataset.

duces a minor performance loss, several improvements minimize its impact. For example, we add a noise to the temporal features during training to emulate compression loss. Then, the deformation field serves as a denoiser, further improving performance. Nevertheless, the temporal regularization encourages static Gaussians in the temporal features to converge to 0, thereby minimizing the effect of temporal feature compression on static areas.

Per-GOP Quantitative Results

We show results for each GOP in the N3DV dataset of various metrics, including PSNR, SSIM, LPIPS(VGG) in Table 7. StreamSTGS trains in GOP units, effectively solving the cumulative error generated by frame-by-frame training methods. As the GOP grows, there is no decrease in performance.

Compression QP

To supplement the quantitative analysis presented in this paper, we provide a comprehensive visual assessment of all scenes under varying compression parameters, as illustrated in Fig. 14 through Fig. 18. When $QP = 32$, the N3DV dataset achieves an average PSNR of 30.76, while maintaining relatively high rendering quality. Static regions exhibit satisfactory quality within a QP range of 16 to 32, with only minor artifacts appearing in dynamic regions.

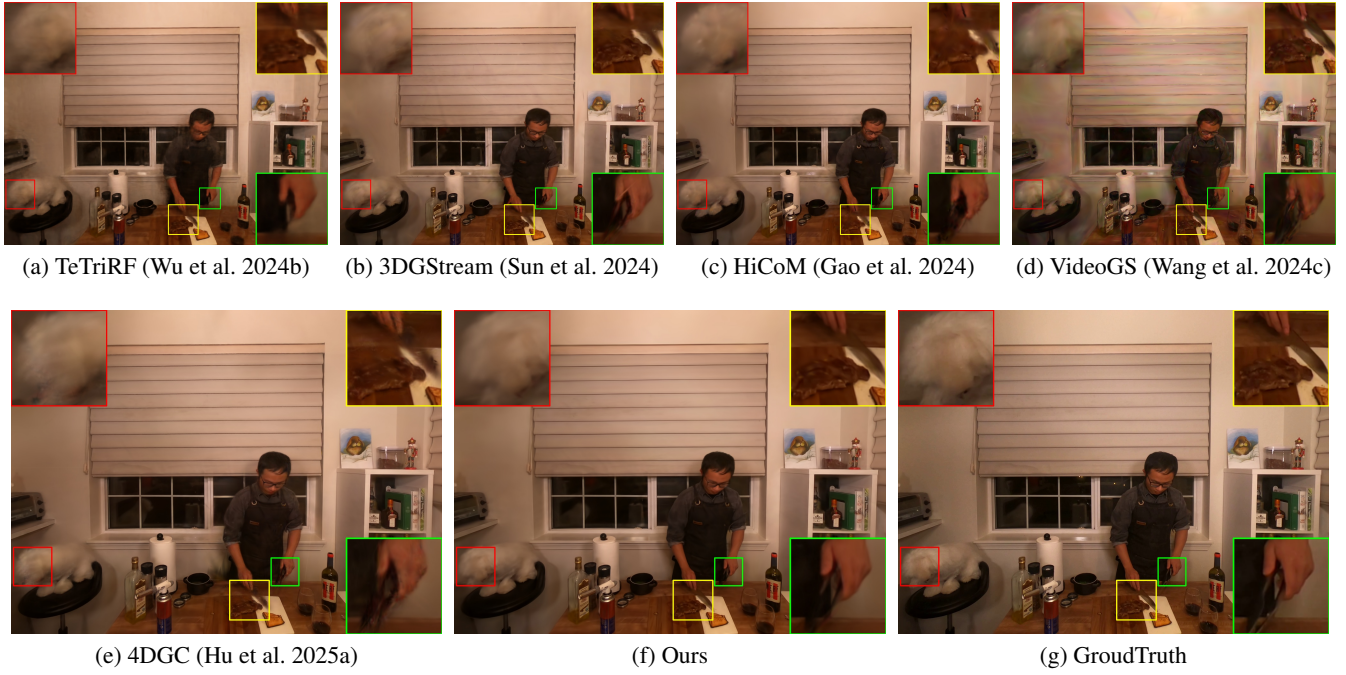


Figure 8: Qualitative comparison of ours with the benchmark methods on *Cut Roasted Beef* scene of N3DV dataset.

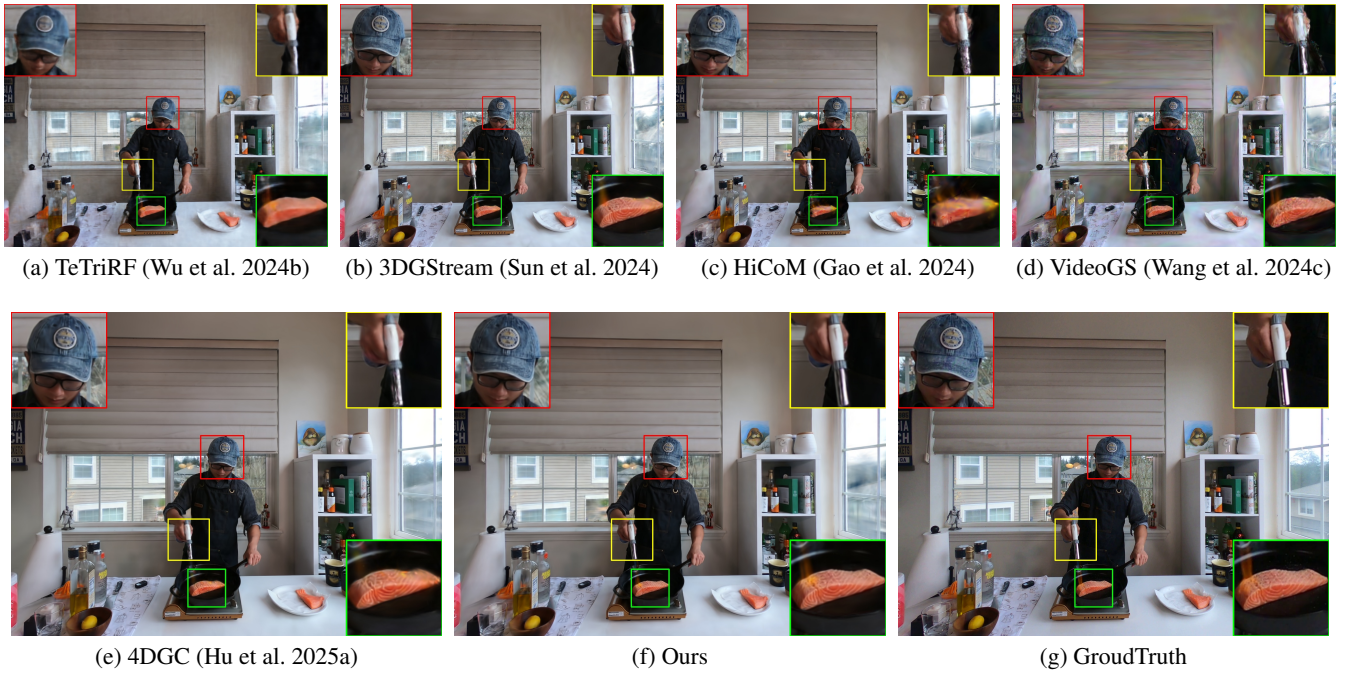


Figure 9: Qualitative comparison of ours with the benchmark methods on *Flame Salmon* scene of N3DV dataset.

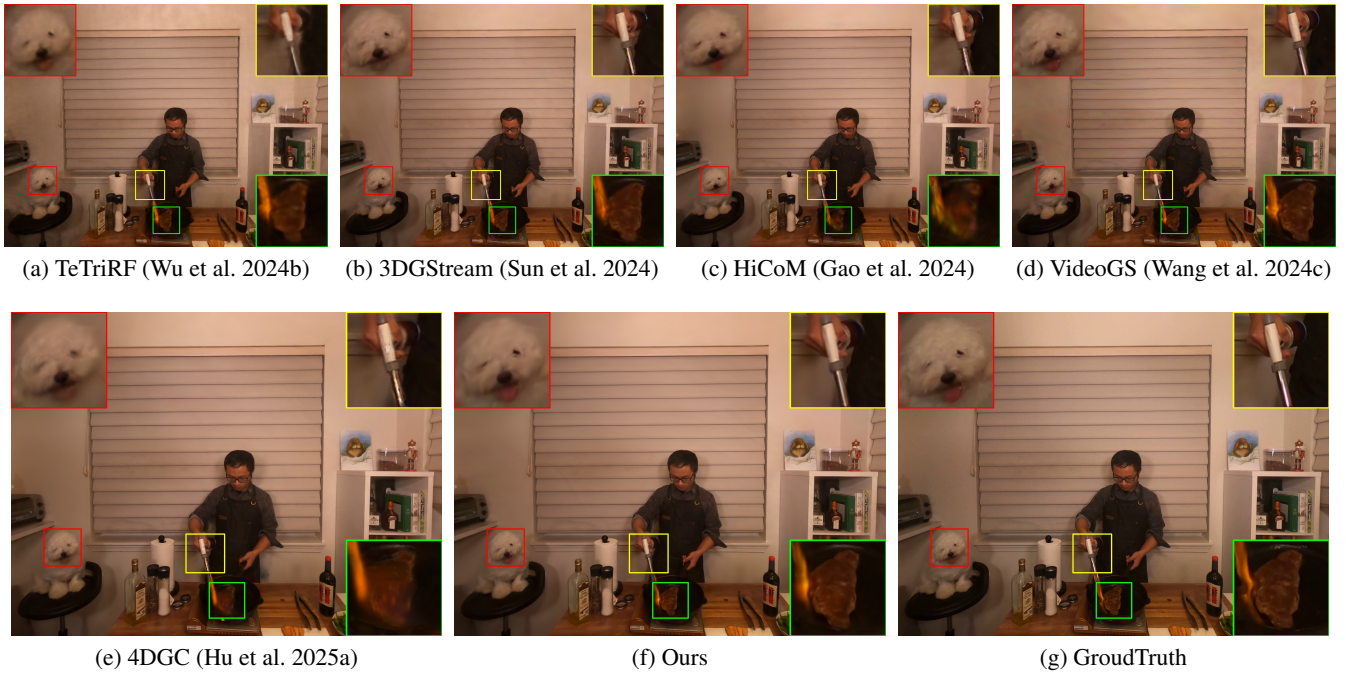


Figure 10: Qualitative comparison of ours with the benchmark methods on *Flame Steak* scene of N3DV dataset.

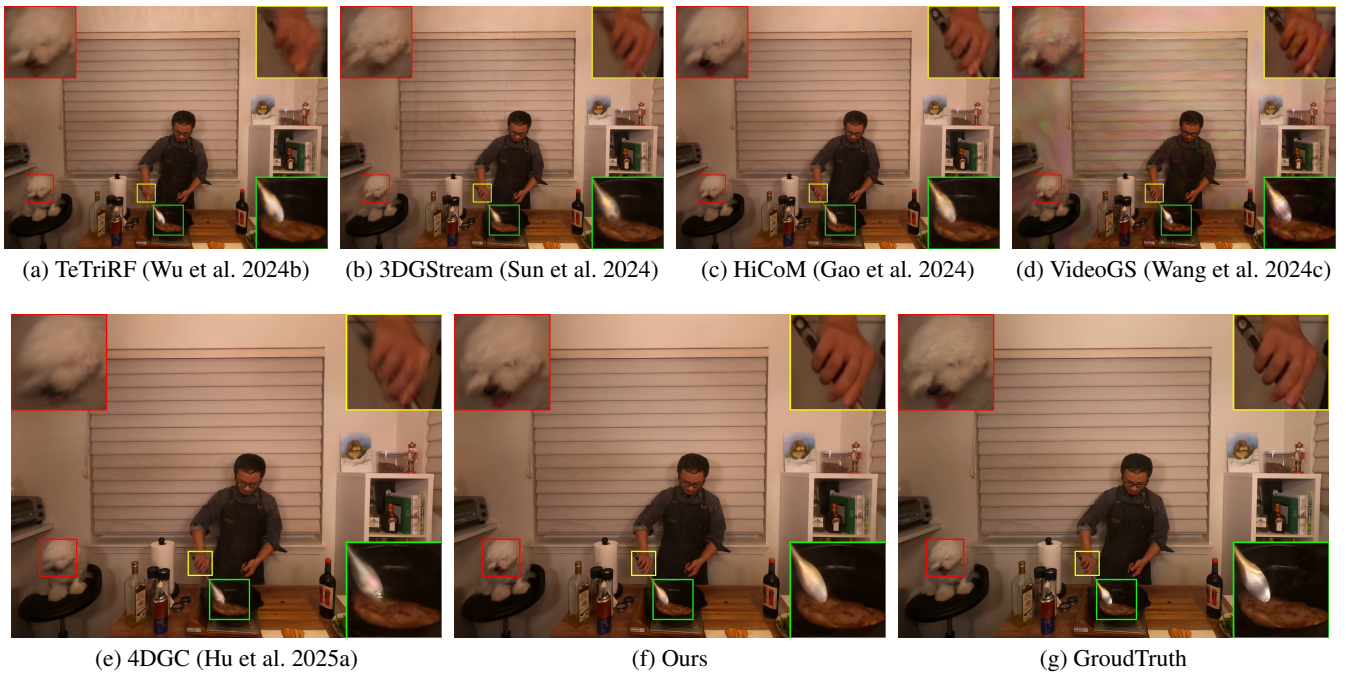


Figure 11: Qualitative comparison of ours with the benchmark methods on *Sear Steak* scene of N3DV dataset.



Figure 12: Qualitative comparison of ours with the benchmark methods on *Discussion* scene of MeetRoom dataset.

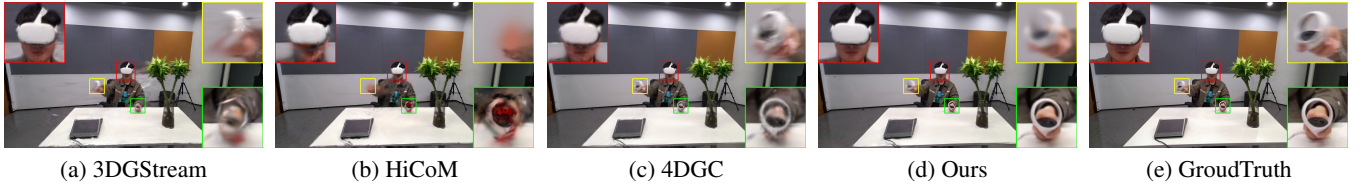


Figure 13: Qualitative comparison of ours with the benchmark methods on *Vrheadset* scene of MeetRoom dataset.

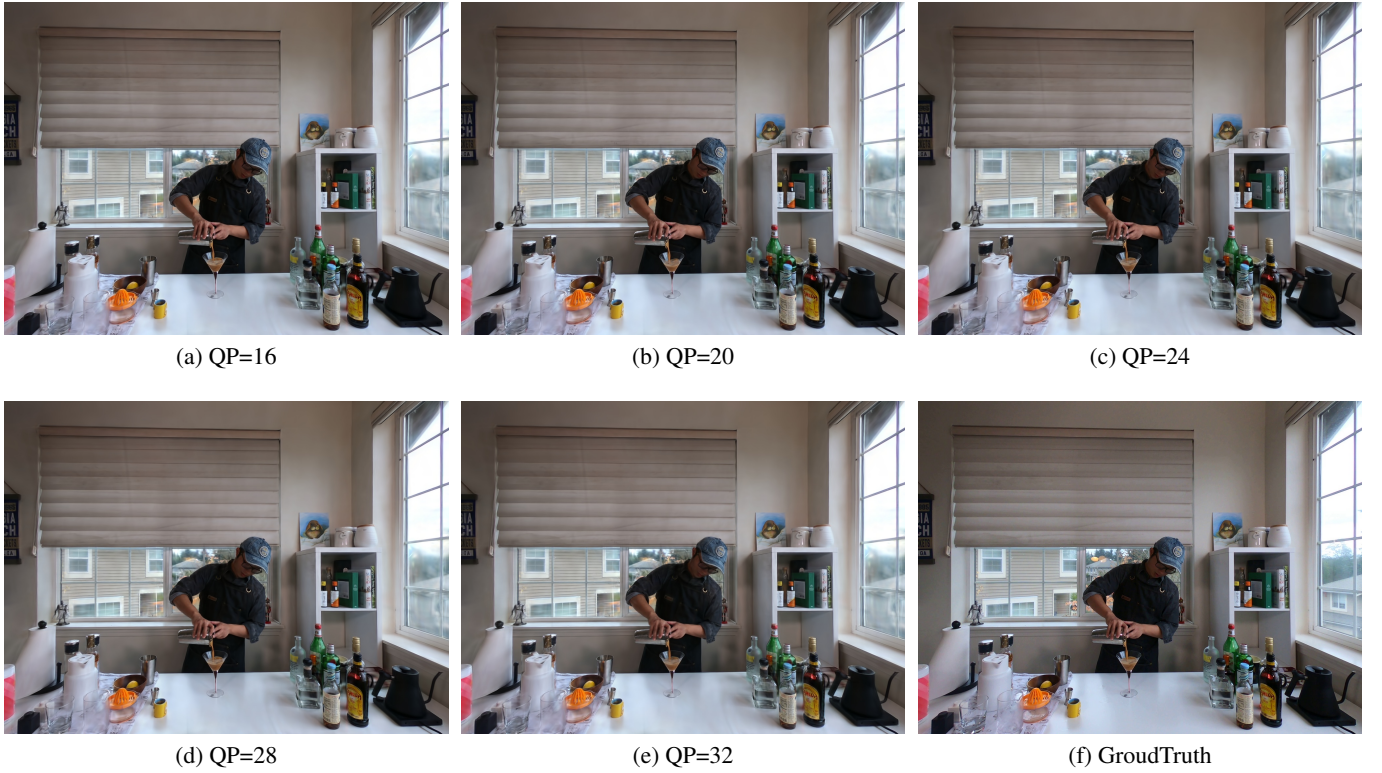


Figure 14: Qualitative comparison under different compression qp on *Coffee Martini* scene of N3DV dataset.



(a) QP=16



(b) QP=20



(c) QP=24



(d) QP=28



(e) QP=32



(f) GroudTruth

Figure 15: Qualitative comparison under different compression qp on *Cook Spinach* scene of N3DV dataset.



(a) QP=16



(b) QP=20



(c) QP=24



(d) QP=28



(e) QP=32



(f) GroudTruth

Figure 16: Qualitative comparison under different compression qp on *Cut Roasted Beef* scene of N3DV dataset.



(a) QP=16



(b) QP=20



(c) QP=24



(d) QP=28



(e) QP=32



(f) GroudTruth

Figure 17: Qualitative comparison under different compression qp on *Flame Salmon* scene of N3DV dataset.



(a) QP=16



(b) QP=20



(c) QP=24



(d) QP=28



(e) QP=32



(f) GroudTruth

Figure 18: Qualitative comparison under different compression qp on *Sear Steak* scene of N3DV dataset.