GIFStream: 4D Gaussian-based Immersive Video with Feature Stream

Supplementary Material

1. Overview

In this supplementary material, we present three sections: (I) additional details on the methods, (II) supplementary information on the experiments, and (III) a discussion of limitations and future work.

2. Supplementary for Methods

Details on 3D-to-2D Sorting: We adopt the sorting strategy introduced in PLAS [7] to reorganize the Gaussian primitives from 3D space to 2D space based on the similarity of their attributes. While [7] sorts static 3D Gaussian primitives using their positions, zero-degree spherical harmonic (SH) coefficients, and scaling, we sort our canonical anchors based on their positions and the three principal components of the time-independent features $\mathbf{f}$ obtained through Principal Component Analysis (PCA). All the parameters used for sorting are normalized to the range $[0, 1]$ along the channel dimension through min-max normalization.

Compression for Time-Independent Video: Our time-independent video $\mathbf{V}_{\text{TI}}$, comprises two components: time-independent features frames and other attributes frames, including $\mathbf{x}, \mathbf{S}_1, \mathbf{S}_2, \{\mathbf{o}^i\}, M$. For compressing this video, there are many options, including existing 3DGS compression methods [1–4, 7–9] and our feature stream compression methods. We perform compression following the same principle as compressing the time-dependent video to keep our overall method simple and effectively exploit intra-channel and spatial dependencies.

We compress the features frames in an auto-regressive manner, grouping the frames into n fragments, $\{\mathbf{F}_i\}_{i=1}^n$, and predicting their distribution as described in Eq. (1). This compression method effectively leverages both the channel and spatial dependencies of the anchors, as convolutional networks h_ϕ^{ent2} are employed to predict the conditional distribution.

$$h_\theta^{\text{ent2}} : [\mathbf{F}_{i-k}; \cdots; \mathbf{F}_{i-1}] \mapsto \{\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i\} \tag{1}$$

We compress the $\mathbf{S}_1, \mathbf{S}_2, \{\mathbf{o}^i\}, M$ with the per-parameter distribution and adaptive quantization steps $\mathbf{Q}$ predicted from time-independent feature $\bar{\mathbf{f}}$ using neural network h_ϕ^{ent3}:

$$h_\theta^{\text{ent3}} : [\bar{\mathbf{f}}] \mapsto \{\boldsymbol{\mu}, \boldsymbol{\sigma}, \mathbf{Q}\} \tag{2}$$

The quantization of $\mathbf{S}_1, \mathbf{S}_2, \{\mathbf{o}^i\}, M$ is performed using the Straight-Through Estimator (STE):

$$\bar{\boldsymbol{A}} = \text{SG}(\text{round}(\boldsymbol{A}/Q) - \boldsymbol{A}/Q) \times Q + \boldsymbol{A} \tag{3}$$

Methods	Rate	Scenes	PSNR ↑	SSIM ↑	LPIPS (VGG) ↓	Storage (MB) ↓
4DGS	rate0	Bartender	31.58	0.865	0.221	126.2
		CBA	29.43	0.911	0.161	101.6
		AVG	30.50	0.888	0.191	113.9
4DGaussian	rate0	Bartender	31.06	0.858	0.249	108.0
		CBA	27.2	0.875	0.259	107.0
		AVG	29.13	0.867	0.254	107.5
STG	rate0	Bartender	31.4	0.875	0.207	49.1
		CBA	27.85	0.896	0.183	111.3
		AVG	29.63	0.886	0.195	80.2
CSTG	rate0	Bartender	31.12	0.876	0.218	10.7
		CBA	27.85	0.895	0.199	18.8
		AVG	29.48	0.885	0.208	14.7
	rate1	Bartender	31.06	0.874	0.221	8.2
		CBA	27.5	0.892	0.202	16.2
		AVG	29.28	0.883	0.212	12.2
Ours	rate0	Bartender	31.94	0.879	0.190	5.3
		CBA	29.5	0.906	0.187	8.5
		AVG	30.72	0.893	0.189	6.9
	rate1	Bartender	31.69	0.876	0.195	3.3
		CBA	29.46	0.906	0.185	7.1
		AVG	30.57	0.891	0.190	5.2
	rate2	Bartender	31.48	0.873	0.201	2.6
		CBA	29.39	0.905	0.190	6.1
		AVG	30.43	0.889	0.195	4.3
	rate3	Bartender	31.35	0.872	0.207	2.3
		CBA	28.91	0.897	0.207	6.1
		AVG	30.13	0.885	0.207	4.2

Table 1. **Per-Scene Results on The MPEG dataset**. We present the PSNR, SSIM, LPIPS (VGG), and storage result of each methods.

Here, $\boldsymbol{A}$ represents the attributes in $\mathbf{S}_1, \mathbf{S}_2, \{\mathbf{o}^i\}, M$ and SG denotes the stop gradient operation. The entropy loss for $\boldsymbol{A}$ is similar to that of $\mathbf{V}_{GF}$, but the half quantization step is modified from 0.5 to $Q/2$.

For the position $\mathbf{x}$ of anchors, we simply apply 16-bit quantization and store the results as a PNG image.

3. Supplementary for Experiments

Additional Implementation Details: We implement 4DGaussian, 4DGS, STG and CSTG [5, 6, 10, 11] using their official code bases. For the 4DGS, we only keep zero degrees of SH coefficients, therefore the storage will be much smaller than the original version. For the Neur3D dataset, we directly cite reported results from CSTG [5] in our paper. For the MPEG dataset, as no predefined hyper-parameters are available, we conduct multiple experiments with various hyper-parameter combinations, adjusted from the Neur3D configurations, and select the best-performing ones as our comparison benchmarks.

To obtain multi-rate results for CSTG on the MPEG dataset, we adjust the weight of the loss function associated with the pruning mask.

Per-Scene Quantitative Comparison: Detailed per-scene

Experiment	Scenes	Time-independent Feature $\mathbf{f}$ (MB)	Attributes (MB)	Time-dependent Feature $\mathbf{f}_t$ (MB)	Neural networks (MB)
GIFStream w/ Compression	Bartender	0.79	2.85	1.46	0.1
	CBA	1.47	5.62	2.88	0.1
	AVG	1.13	4.23	2.17	0.1
GIFStream w/o Compression	Bartender	18.68		27.42	0.1
GIFStream w/o Sparse Mask M_{de}	Bartender	0.88	2.89	2.68	0.1

Table 2. **Memory Breakdown**. For a more comprehensive evaluation, we present the memory breakdown for each experiment. In the GIFStream w/o Compression experiment, we provide the total memory usage of the time-independent features and attributes.

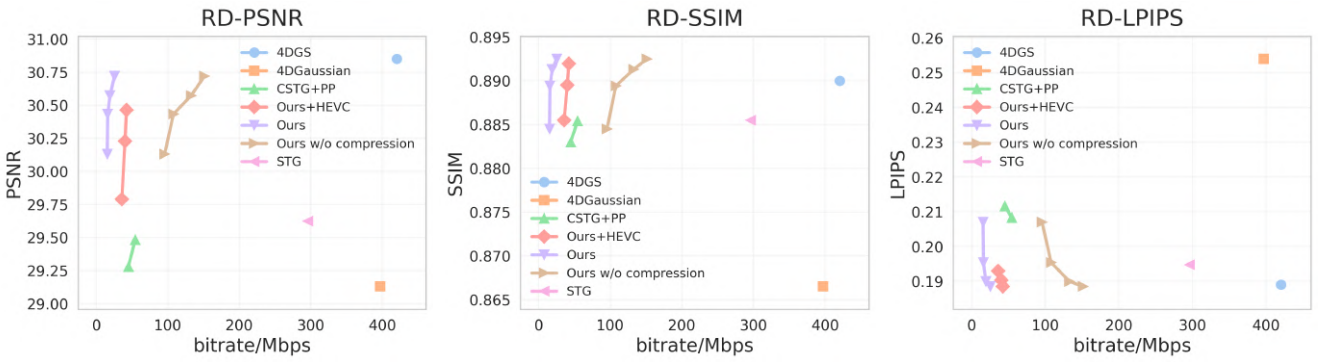

Figure 1. **RD Curve Comparision on MPEG dataset.** We visualize the RD Curve results in the **GOP 65** setting.

results for the Neur3D dataset are provided in Table 3, and detailed per-scene results for the MPEG dataset are provided in Table 1.

More Qualitative Evaluation: We provide additional qualitative results in Fig. 3 and Fig. 2 for better evaluation. To accurately segment the video into complete GOPs, We train GIFStream with a GOP of 60 for the Neur3D dataset and with a GOP of 65 for the MPEG dataset. Additionally, we observe flicker across different GOPs in the static background of the scenes like "flame salmon 1", since the reconstruction of content out of windows is challenging and artifacts in these areas look different in different GOPs. To improve the temporal consistency of these scenes, we utilize the previous GOP as initialization for the next GOP. Specifically, we load the checkpoint from the previous GOP and start training from the 10000th iteration for 10 minutes. During this process, we add and remove points to accommodate new content.

Memory Break Down: We provide the memory breakdown for parts of our experiments in Table 2. This includes the memory usage of the compressed GIFStream using our end-to-end compression (GIFStream w/ Compression) and the uncompressed GIFStream, which is trained without compression techniques (GIFStream w/o Compression). Additionally, we present the increase in memory requirements when sparse feature masks are not applied (GIFStream w/o Sparse Mask M_{de}).

Scenes	PSNR ↑	SSIM ↑	LPIPS ↓	Storage (MB) ↓
coffee martini	28.14	0.905	0.163	11.1
cook spinach	33.03	0.950	0.138	12.0
cut roasted beef	33.19	0.947	0.141	8.8
flame salmon 1	28.51	0.916	0.157	8.2
flame steak	33.76	0.957	0.134	8.2
sear steak	33.83	0.958	0.134	10.2

Table 3. **Per-Scene Results on The Neur3D dataset**. We present the specific results of each scene on the Neur3D dataset.

Compression Utilizing HEVC: We perform simple compression utilizing image and video compression codecs on the MPEG dataset and present the RD-Curve in Fig. 1. Specifically, we apply 16-bit quantization to the attributes $\mathbf{x}, \mathbf{S}_1, \mathbf{S}_2, \{o^i\}_{i=1}^K, M$ and 8-bit quantization for the feature $\mathbf{f}$ and $\mathbf{f}_t$. Subsequently, we use PNG compression for $\mathbf{x}, \mathbf{S}_1, \mathbf{S}_2, \{o^i\}_{i=1}^K, M, \mathbf{f}$ and employ HEVC to compress $\mathbf{f}_t$. While being inferior to our end-to-end compression solution, the HEVC-based compression also yields promising performance compared with other baselines. This demonstrates that our proposed feature streams are also compatible with existing video codecs.

4. Limitation

Our representation may exhibit inconsistencies in the background area between different GOPs, particularly in the distant background, where there are insufficient points for ini-

Figure 2. **More Qualitative Evaluation**. We present complete frames from the MPEG dataset along with a comparison of local details in this figure.

tialization. This is due to the instability of densification. The adaptive sampling strategy from STG [6] or the continual training approach can help alleviate this issue. Additionally, since our representation and compression methods require neural networks for inference, the computational demands may be unacceptable for some mobile devices.

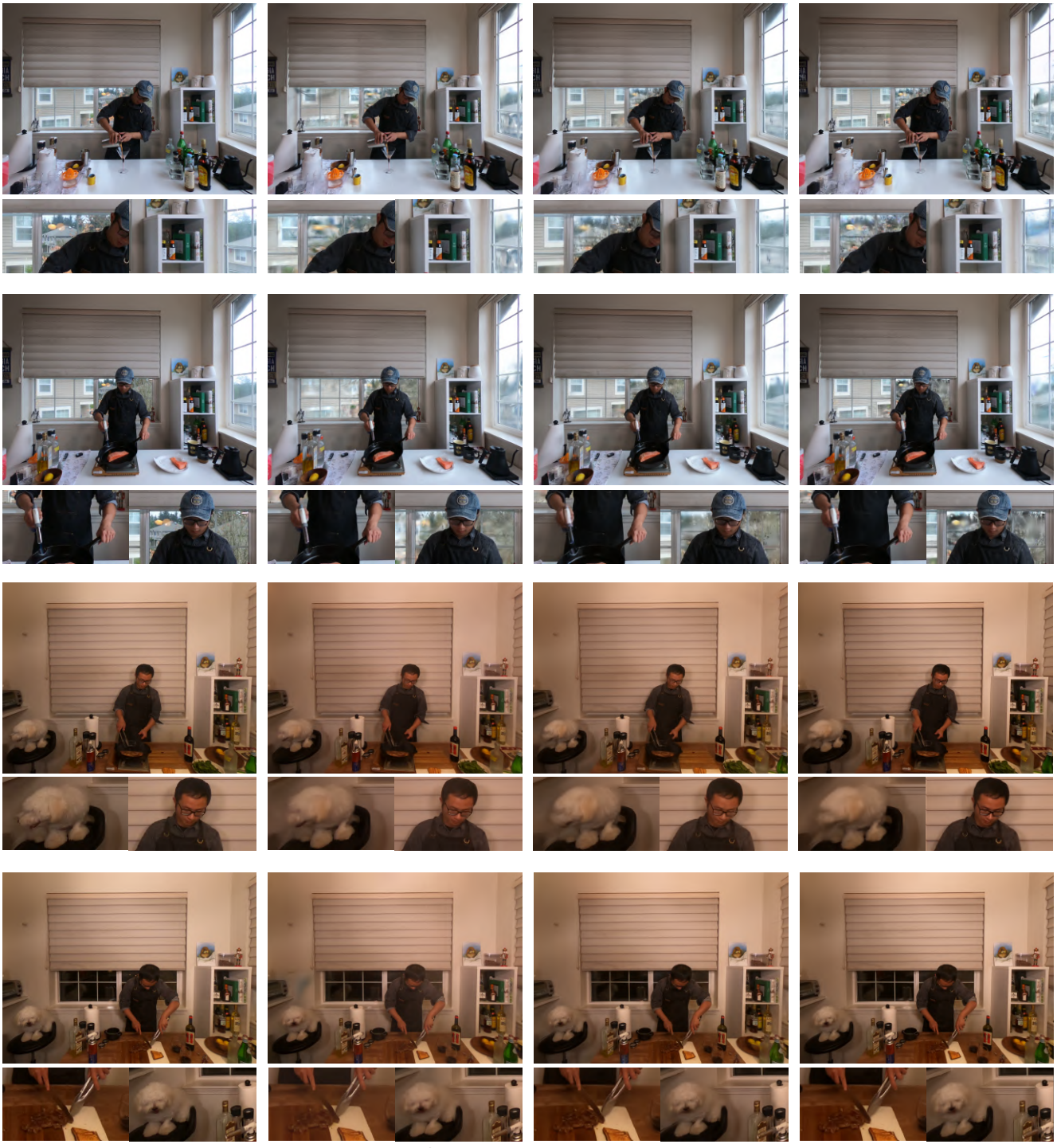

Figure 3. **More Qualitative Evaluation**. We present complete frames from the Neur3D dataset along with a comparison of local details in this figure.

References

[1] Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Hac: Hash-grid assisted context for 3d gaussian splatting compression. In *European Conference on Computer Vision*, pages 422–438. Springer, 2025. 1

[2] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *arXiv preprint arXiv:2311.17245*, 2023.

[3] Sharath Girish, Kamal Gupta, and Abhinav Shrivastava. Eagles: Efficient accelerated 3d gaussians with lightweight encodings. *arXiv preprint arXiv:2312.04564*, 2023.

[4] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21719–21728, 2024. 1

[5] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian splatting for static and dynamic radiance fields. *arXiv preprint arXiv:2408.03822*, 2024. 1

[6] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8508–8520, 2024. 1, 3

[7] Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. Compact 3d scene representation via self-organizing gaussian grids. *arXiv preprint arXiv:2312.13299*, 2023. 1

[8] KL Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Compact3d: Compressing gaussian splat radiance field models with vector quantization. *arXiv preprint arXiv:2311.18159*, 2023.

[9] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10349–10358, 2024. 1

[10] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20310–20320, 2024. 1

[11] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv:2310.10642*, 2023. 1