Layered 4D-Rotor Gaussian Splatting: A Compressed Representation for Long Dynamic Scenes — Supplementary Material

Contents

A. Details of RCQ

Starting from layer 1, we perform RCQ for each block of buckets. We carry out global weighted vector quantization for each layer, specifically on the normalized scale, rotor spatial, and rotor temporal components. The codeword $\mathbf{c}_i^g$ assigned by global vector quantization to each Gaussian $\mathbf{g}_i$ in this layer is given by:

$$\mathbf{c}_i^g = \arg\min_{\mathbf{c}\in\mathcal{C}^g} \|\mathbf{g}_i - \mathbf{c}\|_2. \tag{1}$$

Here, $\mathcal{C}^g$ is the global codebook and $\mathbf{c}$ represents each codeword in $\mathcal{C}^g$. Next, we construct a local codebook $\mathcal{C}^l$ for each block of buckets:

$$\mathcal{C}^l = \text{weighted-k-means}\left(\{\mathbf{g}_j \mid \mathbf{g}_j \in B_k\}, K\right), \tag{2}$$

where $\mathbf{g}_j$ denotes a Gaussian in block B_k, and K is the codebook size, which is set to the final size of $\mathcal{C}^g$. Next, we calculate the residual $\mathbf{r}_m$ for each codeword $\mathbf{l}_m$ in $\mathcal{C}^l$ as follows:

$$\forall \mathbf{l}_m \in \mathcal{C}^l, \quad \mathbf{c}_m = \arg\min_{\mathbf{c}\in\mathcal{C}^g} \|\mathbf{l}_m - \mathbf{c}\|_2, \quad \mathbf{r}_m = \mathbf{l}_m - \mathbf{c}_m. \tag{3}$$

Subsequently, we perform residual quantization to obtain the residual codebook $\mathcal{R}^l$:

$$\mathcal{R}^l = \text{weighted-k-means}\left(\{\mathbf{r}_m \mid \mathbf{l}_m \in \mathcal{C}^l\}, M\right), \tag{4}$$

where M is the residual codebook size. The weight w_m for the residual $\mathbf{r}_m$ is defined as the sum of weights of the Gaussians assigned to $\mathbf{l}_m$:

$$w_m = \sum_{\mathbf{g}_j\in\mathcal{S}_m} W_j, \quad \mathcal{S}_m = \left\{\mathbf{g}_j \mid \arg\min_{\mathbf{l}\in\mathcal{C}^l} \|\mathbf{g}_j - \mathbf{l}\|_2 = \mathbf{l}_m\right\}. \tag{5}$$

Here, W_j represents the weight of the Gaussian $\mathbf{g}_j$, and $\mathcal{S}_m$ is the set of Gaussians in the block whose nearest local codeword is $\mathbf{l}_m$.

B. Additional Experiments

B.1. Additional Implementation Details

Initialization. We initialize our model using a colored sparse point cloud generated by COLMAP [2] from the initial frames across all camera views.

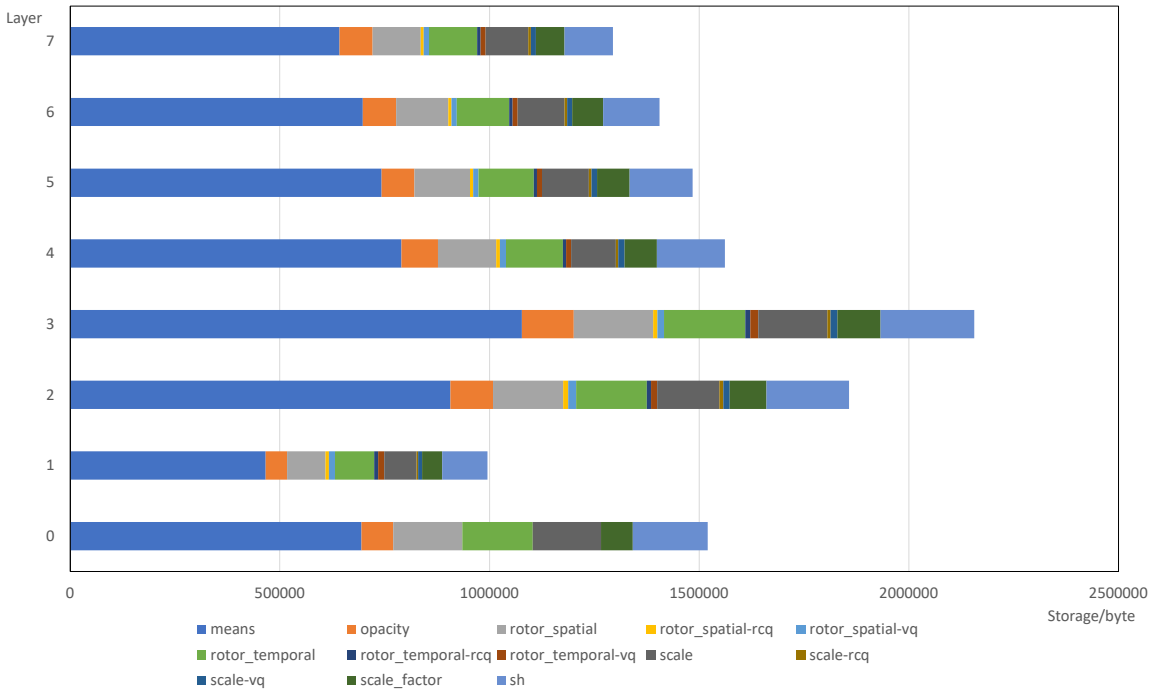


Figure 10. **Compressed Indices Storage per Layer.** We visualize the size of each layer's indices after entropy encoding. The Gaussian means are stored in FP16 precision, which accounts for over 40% of the total index storage.

Training. We optimize our model using Adam for 40K steps on the *10s* scenes, 200K steps on the *40s* scenes from the N3DV dataset, and 2M steps on the *10min Bike* scene from the SelfCap dataset. The settings for our loss functions, learning rates, densification, pruning, and opacity reset follow those of [1, 3].

Compression. Following C3DGS [6], we adopt a compact model configuration for our *Small* setting. Specifically, we use a codebook size of 4096 for all vector-quantized components, including SH, normalized scale, and the spatial and temporal components of the rotor. The quantization thresholds are set to 2×10^{-7} for normalized scale, 1×10^{-6} for the spatial rotor, 8×10^{-6} for the temporal rotor, and 6×10^{-7} for SH. The residual codebook size is fixed at 512 across all layers. In the *Large* model setting, we increase the SH codebook size to 8192 and reduce the SH threshold to 1×10^{-7}, keeping all other settings unchanged.

B.2. Additional Compression Details

B.2.1. Storage Breakdown

Fig. 10 shows the storage consumption of Gaussian indices and means. We store the means using FP16 precision without applying any quantization, as quantizing the means would cause spatial collisions among Gaussian centers, severely degrading the reconstruction accuracy.

Fig. 11 presents the storage breakdown across all codebooks, with SH dominating the storage footprint. This substantial storage requirement aligns with our experimental findings, which demonstrate that SH compression has the most significant impact on rendering quality.

Fig. 12 further illustrates the change in the number of Gaussians across layers before and after compression. Our pruning strategy significantly reduces the total number of Gaussians, directly contributing to the observed FPS improvement. Furthermore, our layer-wise merging strategy effectively reduces the total number of layers by combining those with fewer Gaussians, thereby improving the overall compression ratio.

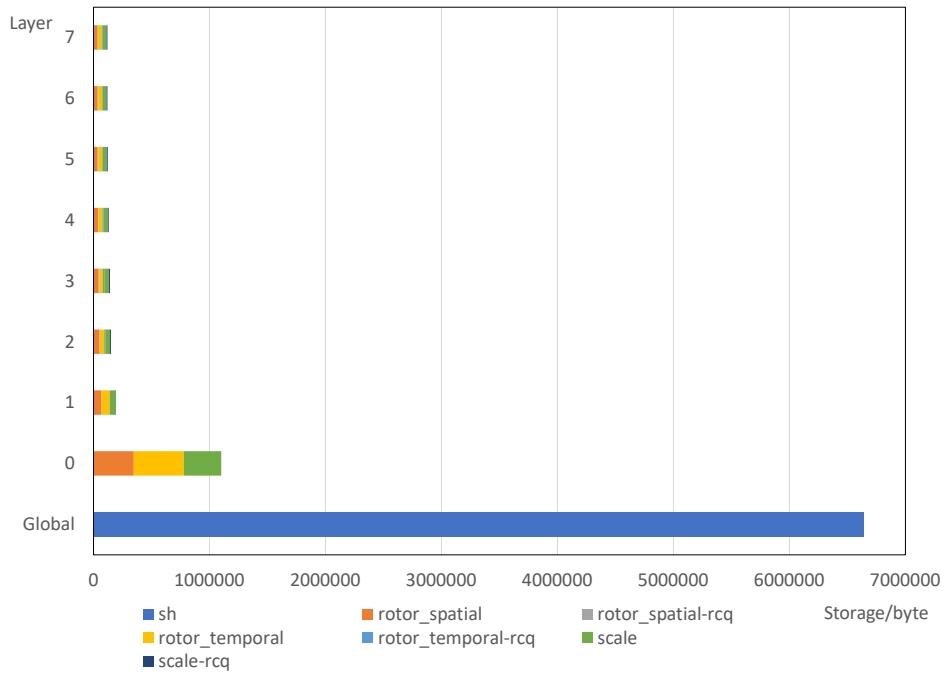


Figure 11. **Codebook Storage per Layer.** We visualize the sizes of the global and layer-wise codebooks. The SH codebooks account for approximately 75% of the total codebook storage, whereas the RCQ codebooks incur minimal storage overhead.

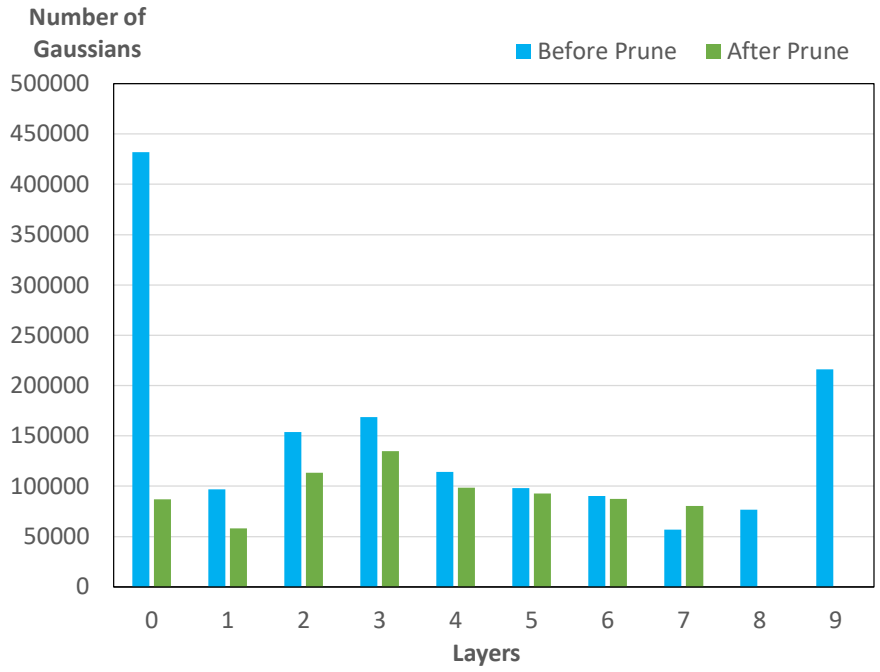


Figure 12. **Number of Gaussians per Layer.** Our compression method prunes Gaussians that are assigned zero weights during vector quantization. This significantly reduces the total number of 4D Gaussians while maintaining rendering quality. Furthermore, the final two layers are merged into layer 7 due to their low Gaussian count after pruning.

B.2.2. Timing Statistics

Tab. 6 details the time cost of our compression pipeline. Notably, the sensitivity calculation stage accounts for 97% of the total processing time.

Table 6. **Time Consumption per Compression Step.** We report the runtime for each step on the 40s *Flame Salmon* scene. Unlike C3DGS [6], our method does not require additional fine-tuning. However, computing sensitivity dominates the time cost due to the massive number of frames (over 20,000). Conversely, the remaining steps are highly efficient thanks to our optimized C++/CUDA implementation. Timings are measured on an NVIDIA RTX 5090 GPU.

Compression Step	Time (s)
Sensitivity Calculation	439.1
Pruning & Merging	0.5
VQ, RCQ, & SQ	12.4

B.2.3. Analysis of Perceptual Fidelity

To assess the temporal stability of our compressed representation, we measure the Just-Objectionable-Difference (JOD) [5] between the rendered videos of our full model ("Ours") and the heavily compressed variant ("Ours Small"). The JOD metric evaluates perceptual video quality on a scale of 0 to 10, where 10 indicates perfect perceptual equivalence. As shown in Table 7, our compressed model achieves high JOD scores on both the *60s Bike* and *40s Flame Salmon* sequences. This confirms that our pipeline effectively preserves temporal coherence despite a significant reduction in storage.

The slight deviation from a perfect JOD score stems primarily from the aggressive quantization of SH coefficients in the "Small" variant. Unlike geometric parameters (scale and rotor) that maintain structural integrity, SH coefficients govern view-dependent color radiance. Compressing them leads to a loss of high-frequency color information, manifesting as minor luminance shifts and reduced contrast. While this degradation is largely imperceptible in static backgrounds, complex motion in dynamic regions perceptually amplifies these color inaccuracies, accounting for the minor temporal inconsistencies.

Table 7. **JOD Evaluation.** We compare the temporal stability between our full model and the heavily compressed variant.

Scene	JOD Score ↑
60s Bike	8.20
40s Flame Salmon	8.86

B.3. Additional Results

B.3.1. Per-Scene Results

Tables 8 and 9 detail the per-scene quantitative results on the N3DV and SelfCap datasets, respectively. For the SelfCap scenes, we adopt the official testing viewpoints where available: `0009` for *Bike*, `0015` for *Hair*, and `0007` for *Corgi*. For the remaining three scenes without officially designated testing views, we explicitly select `09` for *Bar*, `0015` for *Dance*, and `0011` for *Yoga*.

Furthermore, as noted in the original SelfCap dataset, the baseline TGH [9] neither withheld testing views during training nor released quantitative metrics or source code for this dataset. Consequently, we primarily benchmark the compressed variants against our own uncompressed full model.

B.3.2. Comparison on SelfCap Dataset

As noted in the main text, we provide an additional comparison against recent short-sequence baselines on a 1-second clip from the SelfCap dataset [9]. While baselines like FreeTimeGS [8] are heavily optimized for peak fidelity on brief snippets, our representation is inherently designed to scale efficiently across thousands of frames. Despite this architectural difference, Table 10 demonstrates that our method achieves highly competitive visual quality and state-of-the-art rendering speeds, significantly outperforming other approaches in efficiency.

B.3.3. Quantitative Results on HyperNeRF Dataset

To evaluate the capability of our method on casually captured videos, we conduct additional experiments on four scenes (*3D Printer*, *Broom*, *Chicken*, and *Banana*) from the HyperNeRF dataset [7]. As presented in Table 11, our full model achieves

Table 8. **Per-Scene Results on N3DV Dataset.** Quantitative results of our method and its compressed variants across all 6 N3DV scenes. For *Flame Salmon*, we additionally provide a complete 40-second version as a separate comparison.

Method	Ours		Ours Large		Ours Small	
	PSNR	Storage	PSNR	Storage	PSNR	Storage
Flame Steak	34.30	163.9 MB	34.09	13.0 MB	33.86	8.3 MB
Coffee Martini	28.74	206.1 MB	28.60	14.6 MB	28.48	9.5 MB
Sear Steak	34.25	158.3 MB	34.09	12.3 MB	33.79	7.7 MB
Cut Roasted Beef	33.75	170.0 MB	33.55	13.9 MB	33.25	8.5 MB
Cook Spinach	33.39	168.8 MB	33.19	13.8 MB	32.95	8.6 MB
Flame Salmon	28.94	216.9 MB	28.82	14.9 MB	28.72	10.1 MB
*Flame Salmon (40s)**	29.27	372.8 MB	29.07	20.8 MB	28.92	16.7 MB
Average (except *)	32.23	180.7 MB	32.06	13.8 MB	31.84	8.8 MB

Table 9. **Per-Scene Results on SelfCap Dataset.** Quantitative results of our method and its compressed variants across all 6 SelfCap scenes.

Method	Ours		Ours Large		Ours Small	
	PSNR	Storage	PSNR	Storage	PSNR	Storage
Bike (10min)	24.90	1787.1 MB	24.73	92.9 MB	24.70	87.4 MB
Bar	27.56	491.7 MB	27.35	27.2 MB	27.25	23.7 MB
Corgi	23.73	228.1 MB	23.59	16.2 MB	23.56	12.7 MB
Dance	23.91	1380.7 MB	23.70	70.6 MB	23.46	62.3 MB
Yoga	25.00	490.3 MB	24.94	28.8 MB	24.89	22.6 MB
Hair	22.72	1194.7 MB	22.61	54.6 MB	22.60	41.9 MB
Average	24.64	928.8 MB	24.49	48.4 MB	24.41	41.8 MB

Table 10. **Quantitative comparison on the SelfCap Dataset.** All methods are evaluated on a 1-second clip to enable direct comparison with recent short-sequence baselines.

Method	PSNR ↑	FPS ↑
STG [4]	24.97	142
Real-Time 4DGS [10]	25.98	65
FreeTimeGS [8]	**27.41**	467
Ours	26.12	**977**

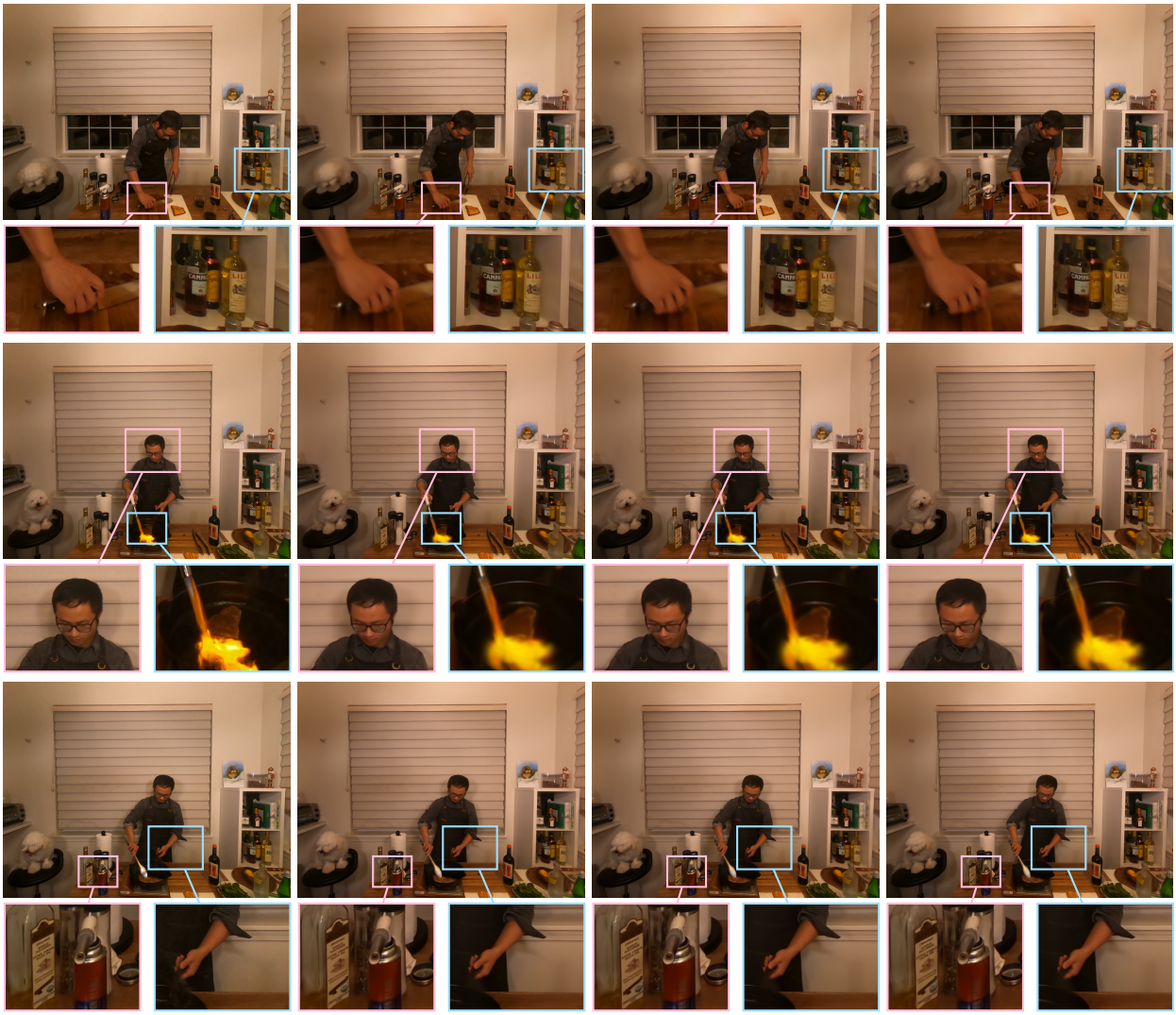
rendering quality comparable to the 4D-rotor GS baseline. More importantly, our compressed variants successfully maintain this competitive visual fidelity while achieving a drastic reduction in storage footprint.

Table 11. **Quantitative results on the HyperNeRF Dataset.** We evaluate our method against the 4D-rotor GS baseline on four casually captured scenes. Our compressed variants achieve massive storage reductions with negligible PSNR drops.

Method	PSNR↑	Storage↓
4D-Rotor GS [1]	25.71	240.8 MB
Ours	**25.84**	219.3 MB
Ours Large	25.69	20.9 MB
Ours Small	25.59	**16.7 MB**

B.3.4. Additional Qualitative Results

We provide further results on the N3DV and SelfCap datasets, illustrated in Fig. 13 and Fig. 14, respectively.



Ground Truth Ours Ours Large Ours Small

Figure 13. **Qualitative Comparison on N3DV Dataset.** Both compressed variants, **Ours Large** and **Ours Small**, closely match the uncompressed model, indicating high visual fidelity under compression.

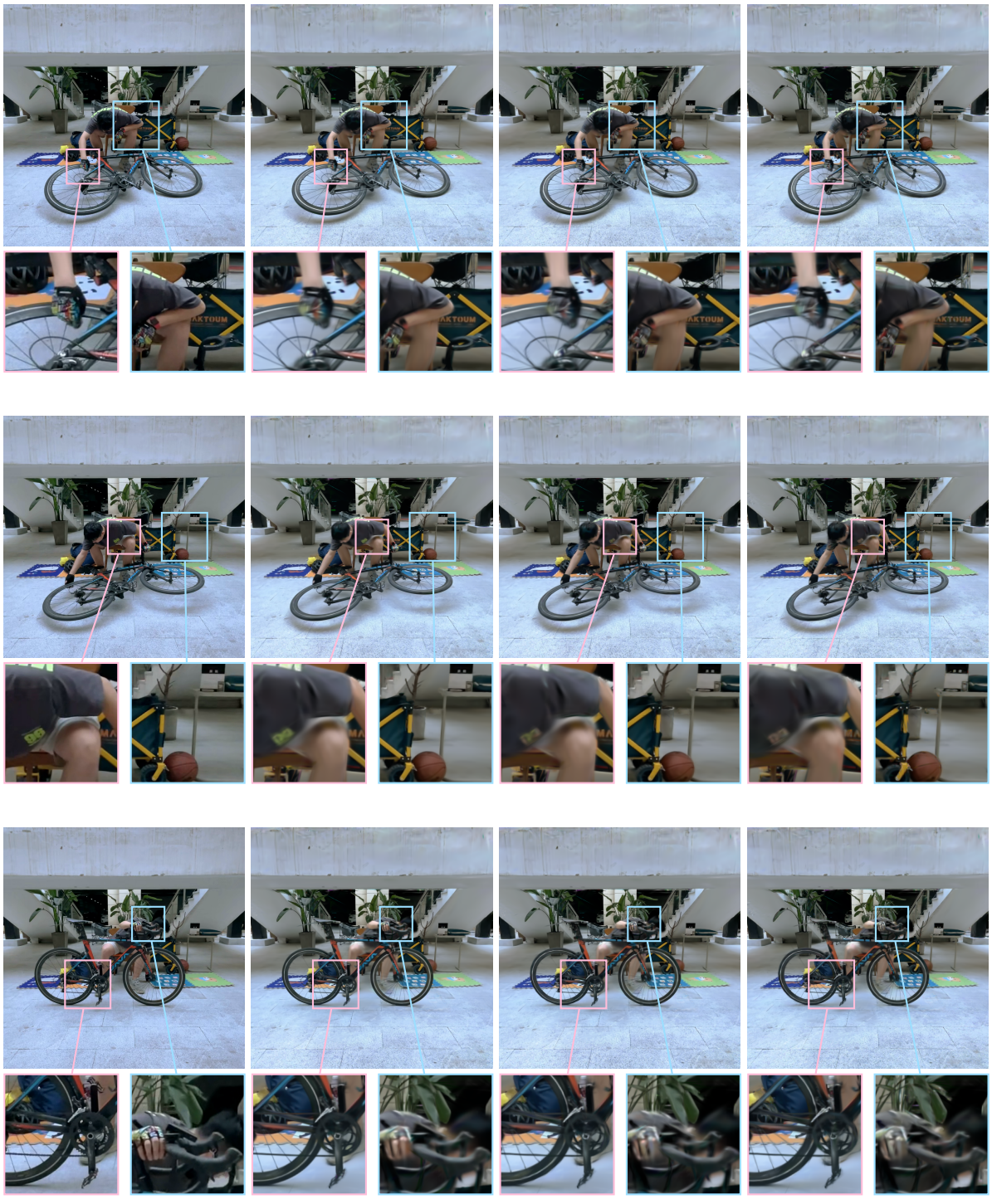


Figure 14. **Qualitative Comparison on SelfCap Dataset.** For these long dynamic sequences, **Ours Large** is visually indistinguishable from the uncompressed model, while **Ours Small** shows minor artifacts in highly dynamic regions due to the tighter storage budget.

References

[1] Yuanxing Duan, Fangyin Wei, Qiyu Dai, Yuhang He, Wenzheng Chen, and Baoquan Chen. 4d-rotor gaussian splatting: towards efficient novel view synthesis for dynamic scenes. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 2, 5

[2] Alex Fisher, Ricardo Cannizzaro, Madeleine Cochrane, Chatura Nagahawatte, and Jennifer L Palmer. Colmap: A memory-efficient occupancy grid mapping framework. *Robotics and Autonomous Systems*, 142:103755, 2021. 1

[3] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14, 2023. 2

[4] 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. 5

[5] Rafał K Mantiuk, Gyorgy Denes, Alexandre Chapiro, Anton Kaplanyan, Gizem Rufo, Romain Bachy, Trisha Lian, and Anjul Patney. Fovvideovdp: A visible difference predictor for wide field-of-view video. *ACM Transactions on Graphics (TOG)*, 40(4):1–19, 2021. 4

[6] 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. 2, 4

[7] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Transactions on Graphics (TOG)*, 40(6):1–12, 2021. 4

[8] Yifan Wang, Peishan Yang, Zhen Xu, Jiaming Sun, Zhanhua Zhang, Yong Chen, Hujun Bao, Sida Peng, and Xiaowei Zhou. Freetimegs: Free gaussian primitives at anytime anywhere for dynamic scene reconstruction. In *CVPR*, 2025. 4, 5

[9] Zhen Xu, Yinghao Xu, Zhiyuan Yu, Sida Peng, Jiaming Sun, Hujun Bao, and Xiaowei Zhou. Representing long volumetric video with temporal gaussian hierarchy. *ACM Transactions on Graphics (TOG)*, 43(6):1–18, 2024. 4

[10] Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In *International Conference on Learning Representations (ICLR)*, 2024. 5