Compact 3D Gaussian Representation for Radiance Field - Supplementary Materials

Joo Chan Lee[1] Daniel Rho[2] Xiangyu Sun[1] Jong Hwan Ko[1]✉ Eunbyung Park[1]✉

Sungkyunkwan University[1], KT[2]

Appendix

1. Implementation Detail

We retained all hyper-parameters of 3DGS and trained models during 30K iterations, and we set the codebook size C and the number of stages L of R-VQ to 64 and 6, respectively. The neural field for view-dependent color uses hash grids with 2-channel features across 16 different resolutions (16 to 4096) and a following 2-layer 64-channel MLP. Due to the different characteristics between the real and synthetic scenes, we adjusted the maximum hash map size and the hyper-parameters for learning the neural field and the mask. For the real scenes, we set the max size of hash maps to 2^{19}, the control factor for the number of Gaussians λ_m to $5e^{-4}$, and the learning rate of the mask parameter and the neural fields to $1e^{-2}$. The learning rate of the neural fields is decreased at 5K, 15K, and 25K iterations by multiplying a factor of 0.33. For the synthetic scenes, the maximum hash map size and the control factor λ_m were set to 2^{16} and $4e^{-3}$, respectively. The learning rate of the mask parameter and the neural fields were set to $1e^{-3}$, where the learning rate of the neural fields was reduced at 25K iterations with a factor of 0.33.

2. Fast inference pipeline

The main paper's analysis shows that our method extends the overall training time slightly more than 3DGS, owing to the time for iterating neural fields and for R-VQ search. However, our approach effectively reduces rendering time for several reasons. First, the proposed masking strategy significantly effectively reduces the number of Gaussians, as demonstrated in our results, leading to reduced training and rendering times. Second, by precomputing grid features at the testing phase, we minimize the operational time of the neural field. Since these grid features, which precede the subsequent MLP input, are not dependent on the view direction itself, they can be prepared in advance of testing. This allows our method to simply process a small MLP for generating view-dependent colors during testing. Third, in the testing phase, the time spent searching for suitable geometry is eliminated. In a manner similar to precomputing grid features, we can index the closest codes from multi-stage codebooks before testing. These strategies collectively enable us to achieve a notably faster rendering speed.

Precomputing time. The precomputing involves a single iteration of hash grid sampling and indexing using the trained R-VQ codebooks. This process requires only negligible runtime, 7.1 ms and 17.8 ms for *bonsai* (indoor) and *bicycle* (outdoor) scenes, respectively.

3. Additional ablation study

3.1. Rate-distortion curve based on each proposal

In addition to the ablation study in the main paper, we conduct an in-depth analysis on the impact of our contributions: volume-based masking, compact color representation, and geometry codebook. As illustrated in Fig. 1, we start with the standard configuration of our approach and adjust the compactness level of the three proposals. We control λ_m, max hashmap size, and the number of R-VQ stages, respectively, by doubling each hyper-parameter. As the performance for the different scenes varies due to their diverse characteristics, we choose three distinct scenes where our approach increases, retains, and decreases visual quality while achieving over $10\times$ compression.

Although the proposed geometry codebook effectively reduces the storage as validated in the main paper, it shows poor performance when the diversity of codebooks is extremely limited. In contrast, our neural field-based color representation demonstrates remarkable robustness even in a low-rate condition across the various scenes, indicating that scaling down the neural field is the best option in environments with severe resource limitations.

These results show that the storage need of our method can be halved with minimal performance loss, and our default configuration is a well-rounded choice for a wide range of scenes.

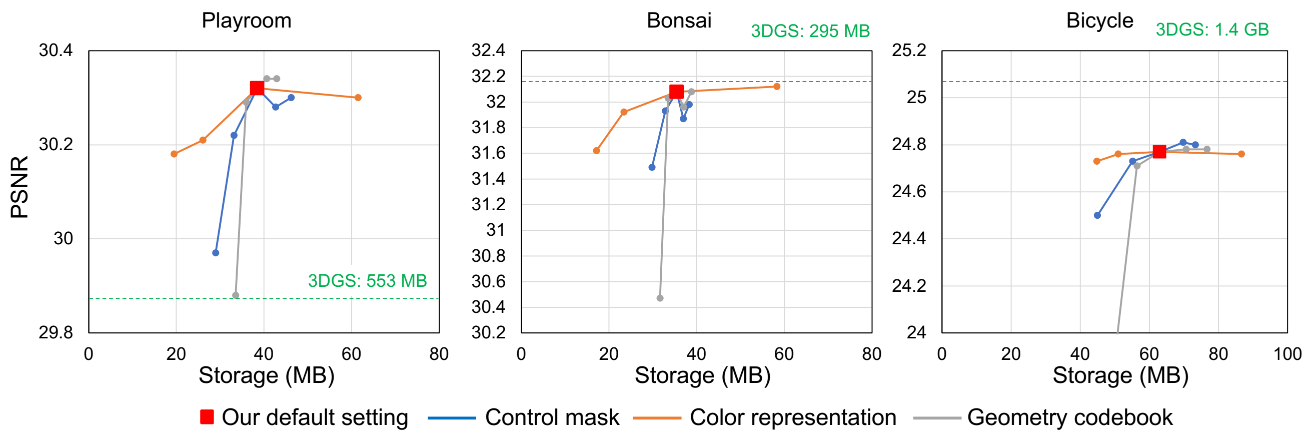

Figure 1. Rate-distortion curves evaluated on diverse scenes. Starting with our default model, we methodically adjust the compactness level of each proposal to evaluate their individual contributions.

Table 1. Performance evaluation of the proposed masking strategy on the *bonsai* scene. We apply only masking without other proposals.

Method	PSNR	SSIM	LPIPS	#Gauss
3DGS	**32.2**	**0.946**	**0.181**	1245 K
Keep densification	31.7	0.941	0.181	1167 K
Threshold-based mask	31.5	0.937	0.192	956 K
Opacity threshold	32.1	0.944	0.186	845 K
	31.9	0.941	0.191	661 K
Proposed masking	**32.2**	0.944	**0.181**	**643 K**

Table 2. Ablation study on the proposed volume-based masking (*bonsai* scene). The performance is evaluated after applying geometry R-VQ and color representation.

Opacity	Scale	λ_m	#Gaussian	PSNR
✓		0.0005	311786	31.50
		0.0001	629837	31.86
	✓	0.0005	508762	31.89
		0.0003	596449	31.97
✓	✓	**0.0005**	**601048**	**32.08**

3.2. Effectiveness of the proposed masking

More baselines of masking. Tab. 1 shows the performance of the proposed masking strategy, compared to straightforward solutions for reducing Gaussians: 1) keep densification during the whole training, 2) prune based on threshold of opacity and scale, 3) increase the opacity threshold for pruning. The proposed learnable masking effectively eliminates non-essential Gaussians, while other baselines show poor performance. It is worth noting that the masking parameter does not require additional storage and effectively

Table 3. Ablation study on represented components by I-NGP (*bonsai* scene).

I-NGP representation				PSNR	Train time	#Gaussians
Opa.	Sca.	Rot.	Col.			
3DGS				32.2	19:21	1245 K
			✓	32.3	24:42	1178 K
✓			✓	9.3	29:55	666 K
	✓		✓	9.3	29:55	559 K
		✓	✓	25.9	27.37	1692 K

Table 4. Evaluation of R-VQ training strategy using the *bonsai* scene.

Train R-VQ	PSNR	SSIM	LPIPS	Train	#Gauss
1K iter with K-means	32.1	0.939	0.193	24:16	601 K
From initial training	32.0	0.936	0.194	**50:38**	508 K

reduced Gaussians accelerate the training process and reduce both inference memory and storage.

Volume-based masking. We have proposed the learnable masking of Gaussians based on their volume as well as transparency. As shown in Tab. 2, masking based on only Gaussian volume outperforms the method that only considers Gaussian opacity. Moreover, the best results are achieved when both volume and transparency are taken into account for masking.

3.3. Usage of I-NGP and R-VQ

Neural fields can represent continuous signals efficiently, whereas VQ works well for repetitive components. In a 3D scene, near Gaussians can be expected to share similar colors but are not guaranteed to have a similar shape; rather, similar shapes can be frequently found in the entire scene. Therefore, I-NGP is not successfully trained to represent other attributes except for color, as demonstrated in Tab. 3.

Table 5. Evaluation of GPU memory requirements for our method compared to 3DGS. 'Mem' indicates the GPU memory requirements.

Scene	bicycle			bonsai			drjohnson			playroom		
Method	PSNR	Storage	Mem.	PSNR	Storage	Mem.	PSNR	Storage	Mem.	PSNR	Storage	Mem.
3DGS	25.08	1.4 GB	9.4 GB	32.16	295 MB	8.7 GB	29.06	774 MB	7.5 GB	29.87	553 MB	6.4 GB
Ours	24.77	63 MB	7.6 GB	32.08	35 MB	8.3 GB	29.26	48 MB	6.5 GB	30.32	39 MB	5.6 GB

Given this intuition, we have proposed an effective representation for each component.

3.4. R-VQ with initial training.

We evaluate the performance when applying R-VQ from the beginning of the training, as shown in Tab. 4. It shows similar performance with the reduced number of Gaussians but it requires over 2× training time, compared to the proposed training strategy.

4. Inference memory

Throughout the main paper, we analyze the effectiveness of our method in terms of the balance between the visual quality, storage requirement, and rendering speed. Our approach effectively reduces not only storage needs but also memory requirements, both of which are key aspects of efficiency. Tab. 5 depicts the GPU memory requirement for inference with the visual quality and storage need, evaluated on diverse scenes. Our method consistently reduces the GPU memory requirements by a safe margin, demonstrating its efficient usage of computing resources.

5. Per-Scene Results

We evaluated the performance on various novel view synthesis datasets. We provide per-scene results for Mip-NeRF 360 (Tab. 6), Tanks&Temples (Tab. 7), Deep Blending (Tab. 7), and NeRF synthetic (Tab. 8) datasets.

Table 6. Per-scene results evaluated on Mip-NeRF 360 dataset.

Scene		bicycle	flowers	garden	stump	tree hill	room	counter	kitchen	bonsai	Avg.
3DGS	PSNR	25.10	21.33	27.25	26.66	22.53	31.50	29.11	31.53	32.16	27.46
	SSIM	0.747	0.588	0.856	0.769	0.635	0.925	0.914	0.932	0.946	0.812
	LPIPS	0.244	0.361	0.122	0.243	0.346	0.198	0.184	0.117	0.181	0.222
	Train (mm:ss)	34:04	25:33	33:46	27:05	24:51	22:55	22:42	26:08	19:18	24:07
	#Gaussians	5,723,640	3,414,994	5,641,235	4,549,202	3,470,681	1,483,653	1,171,684	1,744,761	1,250,329	3,161,131
	Storage (MB)	1350.78	805.94	1331.33	1073.60	819.08	350.14	276.52	411.76	295.08	746.03
	FPS	63.81	132.03	77.19	108.81	100.92	132.51	146.40	122.07	199.86	120.40
Ours	PSNR	24.77	20.89	26.81	26.46	22.65	30.88	28.71	30.48	32.08	27.08
	SSIM	0.723	0.556	0.832	0.757	0.638	0.919	0.902	0.919	0.939	0.798
	LPIPS	0.286	0.399	0.161	0.278	0.363	0.209	0.205	0.131	0.193	0.247
	Train (mm:ss)	42:36	32:37	45:36	33:43	34:08	24:18	27:41	32:59	24:16	33:06
	#Gaussians	2,221,689	1,525,598	2,209,609	1,732,089	2,006,446	529,136	536,672	1,131,168	601,048	1,388,162
	Storage (MB)	62.99	51.15	62.78	54.66	59.33	34.21	34.34	44.45	35.44	**48.82**
	FPS	76.41	142.41	89.49	120.96	110.28	183.03	119.52	114.24	196.08	**128.05**
Ours+PP	PSNR	24.73	20.89	26.72	26.31	22.67	30.88	28.63	30.48	31.98	27.03
	SSIM	0.722	0.554	0.831	0.754	0.637	0.918	0.901	0.919	0.937	0.797
	LPIPS	0.284	0.399	0.158	0.280	0.363	0.209	0.206	0.130	0.193	0.247
	Storage (MB)	42.42	32.05	43.26	33.83	39.08	15.01	15.22	24.39	16.40	**29.07**

Table 7. Per-scene results evaluated on Tank&Temples and Deep Blending.

Dataset		Tanks&Temples			Deep Blending		
Scene		train	truck	Avg.	drjohnson	playroom	Avg.
3DGS	PSNR	22.07	25.35	23.71	29.06	29.87	29.46
	SSIM	0.812	0.878	0.845	0.899	0.901	0.900
	LPIPS	0.208	0.148	0.178	0.247	0.247	0.247
	Train (mm:ss)	12:18	15:24	13:51	24:22	19:22	21:52
	#Gaussians	1,084,001	2,579,252	1,831,627	3,278,027	2,343,368	2,810,698
	Storage (MB)	255.82	608.70	432.26	773.61	553.03	663.32
	FPS	174.42	145.14	159.78	110.46	154.47	132.47
Ours	PSNR	21.56	25.07	23.32	29.26	30.32	**29.79**
	SSIM	0.792	0.871	0.831	0.900	0.902	0.901
	LPIPS	0.240	0.163	0.201	0.258	0.258	0.258
	Train (mm:ss)	16:03	20:36	18:20	30:31	24:35	27:33
	#Gaussians	710,434	962,158	836,296	1,339,005	778,353	1,058,679
	Storage (MB)	37.29	41.57	**39.43**	47.98	38.45	**43.21**
	FPS	185.91	184.83	**185.37**	155.37	205.83	**180.60**
Ours+PP	PSNR	21.62	25.02	23.32	29.16	30.30	**29.73**
	SSIM	0.792	0.870	0.831	0.899	0.900	0.900
	LPIPS	0.240	0.163	0.202	0.257	0.259	0.258
	Storage (MB)	19.07	22.64	**20.86**	28.43	19.21	**23.82**

Table 8. Per-scene results evaluated on NeRF-synthetic dataset.

Scene		chair	drums	ficus	hotdog	lego	materials	mic	ship	Avg.
Ours	PSNR	34.91	26.18	35.44	37.38	35.48	29.97	35.81	31.51	33.33
	SSIM	0.986	0.953	0.987	0.984	0.981	0.958	0.991	0.905	0.968
	LPIPS	0.013	0.041	0.013	0.023	0.018	0.042	0.008	0.113	0.034
	Train (m:ss)	9:19	8:55	6:41	8:20	8:23	6:53	7:12	8:46	8:04
	#Gaussians	153,570	178,615	83,910	64,194	171,826	107,188	56,015	148,442	120,470
	Storage (MB)	6.11	6.54	4.93	4.59	6.42	5.32	4.44	6.02	**5.55**
	FPS	512.10	427.56	706.71	719.85	402.15	638.01	674.34	282.72	545.43
Ours+PP	PSNR	34.58	26.01	35.06	36.71	34.96	29.04	35.57	31.06	32.88
	SSIM	0.985	0.951	0.987	0.983	0.979	0.954	0.991	0.903	0.968
	LPIPS	0.013	0.042	0.013	0.023	0.020	0.043	0.008	0.115	0.034
	Storage (MB)	3.09	3.56	2.11	1.75	3.45	2.51	1.59	3.28	**2.67**