SymGS : Leveraging Reflective Symmetries for 3SDGS Compression SUPPLEMENTARY

Anonymous submission

Supplementary Video

We recommend the readers to refer to the supplementary video for an overview of our proposed framework and qualitative video results.

SymGS Algorithm

Please refer to Alg. 1 for the description of the SymGS algorithm.

Extended Analysis

Visualizing Symmetries in the Scene

We show different symmetries across scenes in Figure 4. The visualizations highlight how our method captures both obvious and subtle symmetry patterns, such as mirrored furniture arrangements, repeated textures, or symmetric structural elements.

To further illustrate the variety of symmetry complexity, we include multiple examples in Figure 5, demonstrating scenes with varying degrees and types of symmetry.

Analysis of View-dependent Appearance

We compare the advantage of HAC-based integration of our framework over 3DGS-based integration in Figure 1. Due to reflections, lighting effects sometimes get reflected to other views. However, in the case of HAC, the usage of MLP helps model view-dependent appearance much better than 3DGS. Notably, while optimizing HAC anchors attributes and MLPs, the MLP responsible for predicting colors is also fine-tuned, allowing the same anchor feature vector, when reflected, to produce different colors under different viewing directions. This occurs because the color-MLP takes in both the feature vector and viewing direction as input to output the color of a point. The original and reflected anchors have distinct viewing directions, despite sharing identical feature vectors. Hence, fine-tuning the color MLP allows for lighting changes over the reflected anchors - a feature not possible in standard Gaussian Splatting. We illustrate this effect in Figure 1.

Mirror Level vs Compression

We analyze the trend of adding mirrors on the compression rate and PSNR by observing the change in the percentage of redundant anchors at each iteration. These redundant

Algorithm 1 SymGS: Iterative Compression of 3DGS

Input: Initial Gaussians $\mathcal{G}^{(0)} = \{G_{init}\}$
Output: Compressed Model Ω, Mirrors $\{M_t\}_{t=0}^{L}$
$\mathcal{X}_{\text{ret}} \leftarrow \emptyset$, $M \leftarrow \emptyset$
Define $X =$ Positions of Gaussians G
for $t = 0$ to L **do**
  Cluster $\mathcal{G}^{M_t}$ into $\{C_k\}_{k=1}^{N_{\text{clstr}}}$
  Initialize accumulator grid $\mathcal{A} \leftarrow 0$
  for all C_k **do**
    for all $(G_i, G_j) \in C_k$ **do**
      $n \leftarrow \frac{x_i - x_j}{\|x_i - x_j\|}, \quad c \leftarrow \frac{x_i + x_j}{2}$
      $(\alpha, \beta) \leftarrow \text{spherical}(n), \quad \gamma \leftarrow n^\top c$
      $v^\alpha = \left\lfloor \frac{\alpha}{\alpha_{res}} \right\rfloor$
      $v^\beta = \left\lfloor \frac{\beta}{\beta_{res}} \right\rfloor$
      $v^\gamma = \left\lfloor \frac{\gamma}{\gamma_{res}} \right\rfloor$
      $\mathcal{A}[v^\alpha][v^\beta][v^\gamma] \mathrel{+}= 1$
    end for
  end for
  $M_t \leftarrow \arg\max \mathcal{A}, \quad M \leftarrow M \cup \{M_t\}$
  Split $\mathcal{G}^{M_t} \rightarrow G^{M_t}_{\text{left}}, G^{M_t}_{\text{right}}, G^{M_t}_{\text{out}}$
  Reflect $G^{M_t}_{\text{left}} \rightarrow \widehat{G}^{M_t}_{\text{right}}$ via M_t
  $G^{M_{t-1}}_{left+out} \leftarrow \{G^{M_L}_{left}, \widehat{G}^{M_L}_{right}, G^{M_L}_{out}\}$
  for $s = t - 1$ to 0 **do**
    $G^{M_s}_{\text{left}} \leftarrow \text{NearestNeighbour}(G^{M_s}_{left+out}, X^{M_s})$
    $G^{M_s}_{\text{out}} \leftarrow G^{M_s}_{left+out} - G^{M_s}_{\text{left}}$
    $\widehat{G}^{M_s}_{\text{right}} \leftarrow \text{Reflect}(G^{M_s}_{\text{left}}, M_s)$
    $G^{M_s}_{left+out} \leftarrow \{G^{M_{s+1}}_{left}, \widehat{G}^{M_{s+1}}_{right}, G^{M_{s+1}}_{out}\}$
  end for
  $\hat{I} \leftarrow \text{Render}(G^{M_0}_{left+out})$
  Compute loss $\mathcal{L}(\hat{I}, I_{\text{GT}})$; update M_t, $G^{M_t}_{\text{left}}$, $G^{M_t}_{\text{out}}$
  $\mathcal{X}_{\text{ret}} \leftarrow \mathcal{X}_{\text{ret}} \cup X^{M_t}_{\text{left}}$
  $\mathcal{G}^{M_{t+1}} \leftarrow G_{\text{left}} \cup G_{\text{out}}$
end for
$\Omega = \mathcal{G}^{M_L}_{\text{left}} \cup \mathcal{G}^{M_L}_{\text{out}} \cup \mathcal{X}_{\text{ret}}$
return Ω, M

anchors do not need to be stored, as they can be regenerated through reflection across the symmetry line. Mathematically, the percentage of redundant anchors at any iteration step is calculated as the cumulative sum of anchors that belonged to G_{right} up to that iteration, represented by $\sum_{k=0}^{k=\text{step}-1} G_{\text{right}}^{M_k}$, in relation to the total number of anchors in the scene. Figure 3 depicts how this percentage varies across 6 scenes for all 5 datasets on which we report our results.

Size Breakdown of Compressed Model

We show a complete breakdown of the size taken by different components being stored in the scene. Please refer to Table 3. The Total Anchors column reflects the cumulative size of anchor position encodings from G_{left} (i.e., X_{ret}) combined with those of the remaining anchors at the final level. This total corresponds to the overall size of anchor positions within Ω.

Timing Analysis

Comparison of execution times for different implementations of the mirror detection algorithm in Figure 2 reveals that the CUDA implementation outperforms the non-CUDA versions. For alternatives, we fully vectorized the PyTorch code (eliminating loops), but this approach leads to heavy memory usage as the number of Gaussians increases. To mitigate this, we also implemented a partially vectorized PyTorch version with a single loop. Among these, the CUDA implementation achieves significantly higher efficiency.

Qualitative and Quantitative Results

We first present a detailed per-scene comparison with HAC across all five datasets in Table 1. A comprehensive quantitative summary comparing different methods (including LPIPS and SSIM metrics) is provided in Table 2. Additional qualitative results can be found in Figure 6.

RCF Calculation

To evaluate the relative compression factor (RCF) improvement of our method over 3DGS and HAC, we follow the same approach used by HAC: computing the total storage size required across all scenes and datasets for each method. The total sizes obtained are: 3DGS – 3524.06 MB, HAC (our run) – 54.19 MB, and SymGS (ours) – 32.71 MB. This corresponds to a compression factor of approximately $108\times$ over 3DGS and $1.66\times$ over HAC. However, this metric overrepresents improvements on large scenes. For instance, a reduction from 1 MB to 0.95 MB gets overshadowed by much larger scenes where relative size reduction might be minimal - say 500 MB to 250 MB. The RCF in such a case would be around 2, even though for the smaller scene, the method did not lead any significant improvement. To address this, we also compute the RCF separately for each dataset and then average the results. We feel this per-dataset averaging gives a more accurate reflection of small and large scale scene improvements, yielding an average RCF of $121\times$ for SymGS, compared to $62.5\times$ for HAC.

Extended Related Works

3DGS Compression

3D Gaussian Splatting (3DGS) has recently emerged as a powerful technique for real-time radiance field rendering due to its explicit representation and high fidelity (Kerbl et al. 2023). However, its high memory and storage requirements have led to an explosion of work on compression and compaction techniques. Attribute compression aims to reduce per-Gaussian storage, especially of spherical harmonics (SH), which dominate the representation. LightGaussian (Fan et al. 2024) applies vector quantization (VQ) to selectively compress SH coefficients guided by a significance score, while RDO-Gaussian (Xie et al. 2024) employs rate-distortion optimized quantization and entropy coding. Compact3D (Zhang et al. 2024b) proposes sensitivity-aware VQ and entropy modeling, whereas MesonGS (Niedermayr et al. 2024) leverages octrees and lossless codecs like LZ77. SOG (Morgenstern et al. 2024) shows that third-degree SH can be removed with minimal visual degradation, and CodecGS (Mao et al. 2024) encodes attributes in tri-planes and compresses them using blockwise DCT and video codecs. Compact3DGS (Smith et al. 2024) replaces SH colors with hash-grid-MLP decoding, while Self-Organizing Gaussians arrange primitives on 2D grids to exploit local smoothness for image-like compression. EAGLES (Lee et al. 2024) encodes attributes as latent vectors decoded by MLPs, maintaining compactness while preserving quality. ContextGS (Zhou et al. 2024) builds on Scaffold-GS (Lu et al. 2024) by hierarchically predicting anchors across levels. Other structure-aware methods like SUNDAE (Yang et al. 2024) and Reduced3DGS (Papantonakis et al. 2024) prune Gaussians using spatial graphs and overlap filters, while methods such as Octree-GS (Ren et al. 2024) and Mini-Splatting (Fang and Wang 2024) utilize depth maps and level-of-detail octrees for compaction. Densification strategies include RDGS (Placeholder et al. 2024), which uses perceptual loss on individual Gaussians to trigger splits, and Taming3DGS (Mallick et al. 2024), which scores candidates globally based on structural and visual cues. Pixel-GS (Zhang et al. 2024c) and FreGS (Zhang et al. 2024a) exploit image-space and frequency-space signals, respectively. Among these, HAC (Chen et al. 2024) stands out as the current state-of-the-art by integrating anchor-based representation, hash-grid assisted context modeling, and adaptive quantization to achieve superior compression ratios and rendered quality across diverse datasets. Recently, HAC++ (Chen et al. 2025) extends HAC by capturing intra-anchor contextual relationships to further enhance compression performance.

Symmetry Detection

Symmetry detection in 3D point clouds is a fundamental problem with applications in geometry processing, segmentation, reconstruction, and compression. A seminal work by Mitra et al. (Mitra, Guibas, and Pauly 2006) introduced partial symmetry detection in point clouds via a Hough-transform-like voting scheme, where points with similar local structure contribute to a 7D accumulator representing

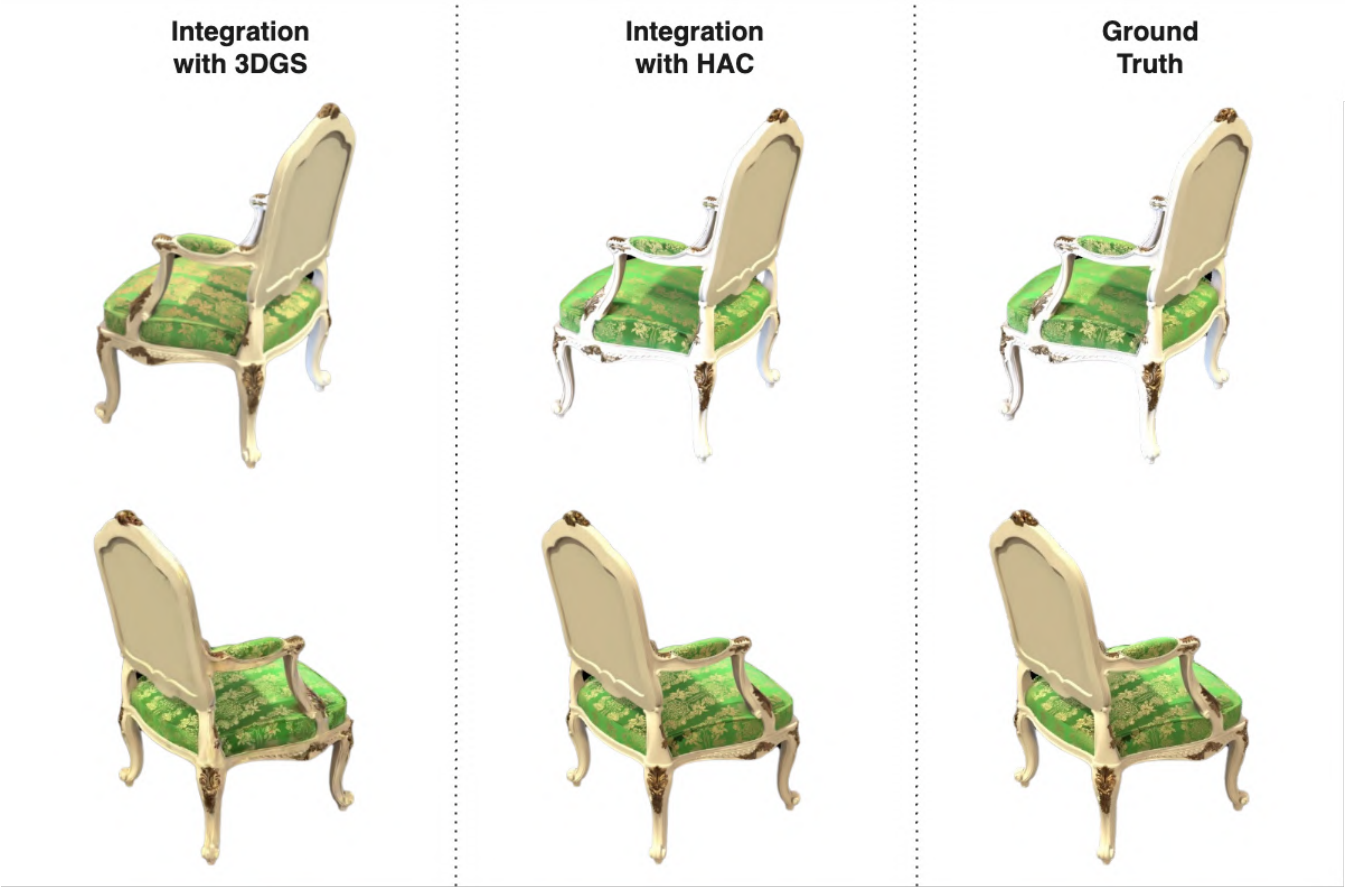

Figure 1: (**Left**) Integration of a our approach with the 3DGS framework, where lighting effects from one side is mirrored (baked) onto the another viewpoint. As a result, the chair's legs appear yellowish in both views. (**Middle**) Integration of a our approach with the HAC framework. Here, despite the presence of reflection, the lighting on each side remains distinct, as the color is inferred by a view-dependent MLP, which accurately models the directional dependence of appearance. Consequently, the legs exhibit correct colors - white on one side and yellow on the other. (**Right**) Ground truth for reference.

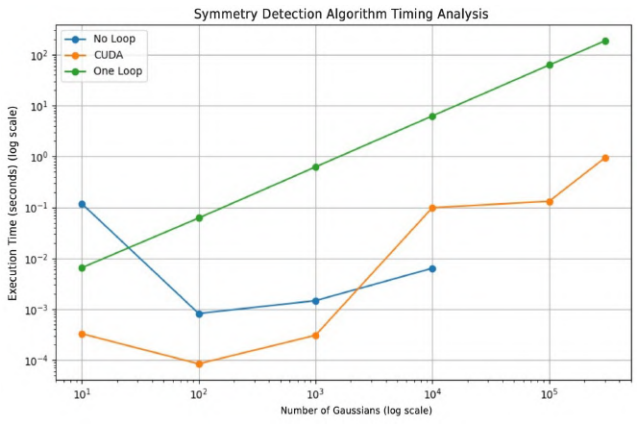

Figure 2: Execution time comparison (log scale) of different mirror detection implementations. **CUDA-based:** Highest efficiency; **No-Loop:** Fully vectorized, memory-intensive. **One-Loop:** Partially Vectorized with a single loop, reduced memory usage.

rotation and translation parameters. While effective, this method is limited to point sets and suffers from discretization artifacts in mirror estimation. Other methods have approached this task through model-driven algorithms that detect global or partial symmetries using descriptors like generalized moments (Martinet et al. 2006), planar reflective transforms (Podolak et al. 2006), and symmetry factored embeddings (Lipman et al. 2010). These methods often assume clean geometry and are sensitive to noise or partial data. To overcome such limitations, learning-based approaches have emerged. PRS-Net (Gao et al. 2020) and E3Sym (Li et al. 2023) employ supervised and unsupervised deep networks, respectively, to detect global planar symmetries. The most recent progress in partial extrinsic symmetry detection is SymCL and SymML (Kobsik, Lim, and Kobbelt 2023), which leverage geodesic patches and contrastive learning to produce transformation-invariant features without requiring labels. These methods demonstrate strong generalization across diverse shape categories and are computationally efficient due to latent space comparisons. Despite such progress, no work to date incorporates symme-

try priors into 3D Gaussian Splatting (3DGS), a popular representation for real-time radiance field rendering. Existing 3DGS compression techniques focus on reducing memory via codebooks (Navaneet et al. 2024), anchor-based primitives (Lu et al. 2024), or hash-based redundancy modeling (Chen et al. 2024), but overlook structural regularities like symmetry. Given the inherent symmetry in most man-made environments (Mitra et al. 2013), symmetry-aware compression holds the potential to significantly improve efficiency and interpretability. This presents an open and impactful direction for extending symmetry detection methods to 3DGS.

Limitations & Future Works

We propose a novel reflective symmetry detection-based 3DGS compression framework, which employs iterative global-to-local mirror detection and mirror-aware optimization, and achieves SOTA performance in compression while maintaining rendering quality. One major limitation of our approach is that if the scene has highly unique features and lacks sufficient symmetry, achieving significant compression becomes difficult. However, we consistently find sufficient symmetries in the standard datasets to enable effective compression. We also observe that our approach is constrained by the discretization of the mirror parameters space. While existing quantization-based compression methods like Scaffold-GS and HAC face similar challenges, this discretization can help reduce noise (e.g. floating artifacts); however, it may also lead to a decline in rendering quality, particularly when the scene contains high-frequency appearance details.

As part of future work, we plan to investigate this issue further and develop approaches to resolve it completely. We would also like to re-emphasize that the plug-and-play capability of our module allows us to apply our method flexibly, enabling further improvements to any new compression algorithms that emerge. Naturally, this also motivate us to extend our framework to deformable Gaussians to enable effective compression of dynamic scenes.

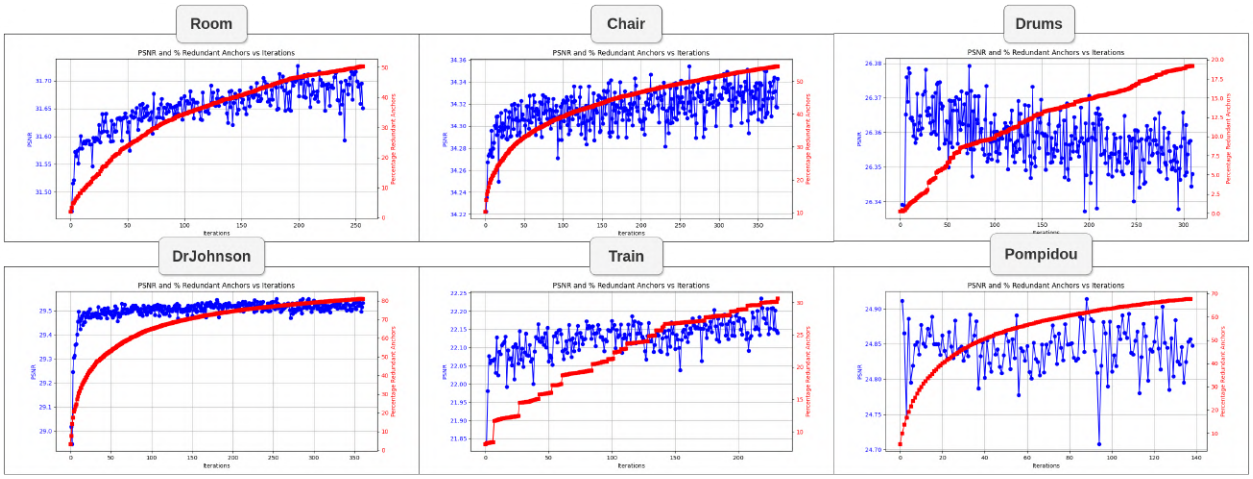

Figure 3: **Mirror Level vs PSNR (Red) and % Redundant Anchors (Blue) :** As we keep on adding more mirrors, the PSNR increases or stays approximately the same, whereas the % redundant anchors increases.

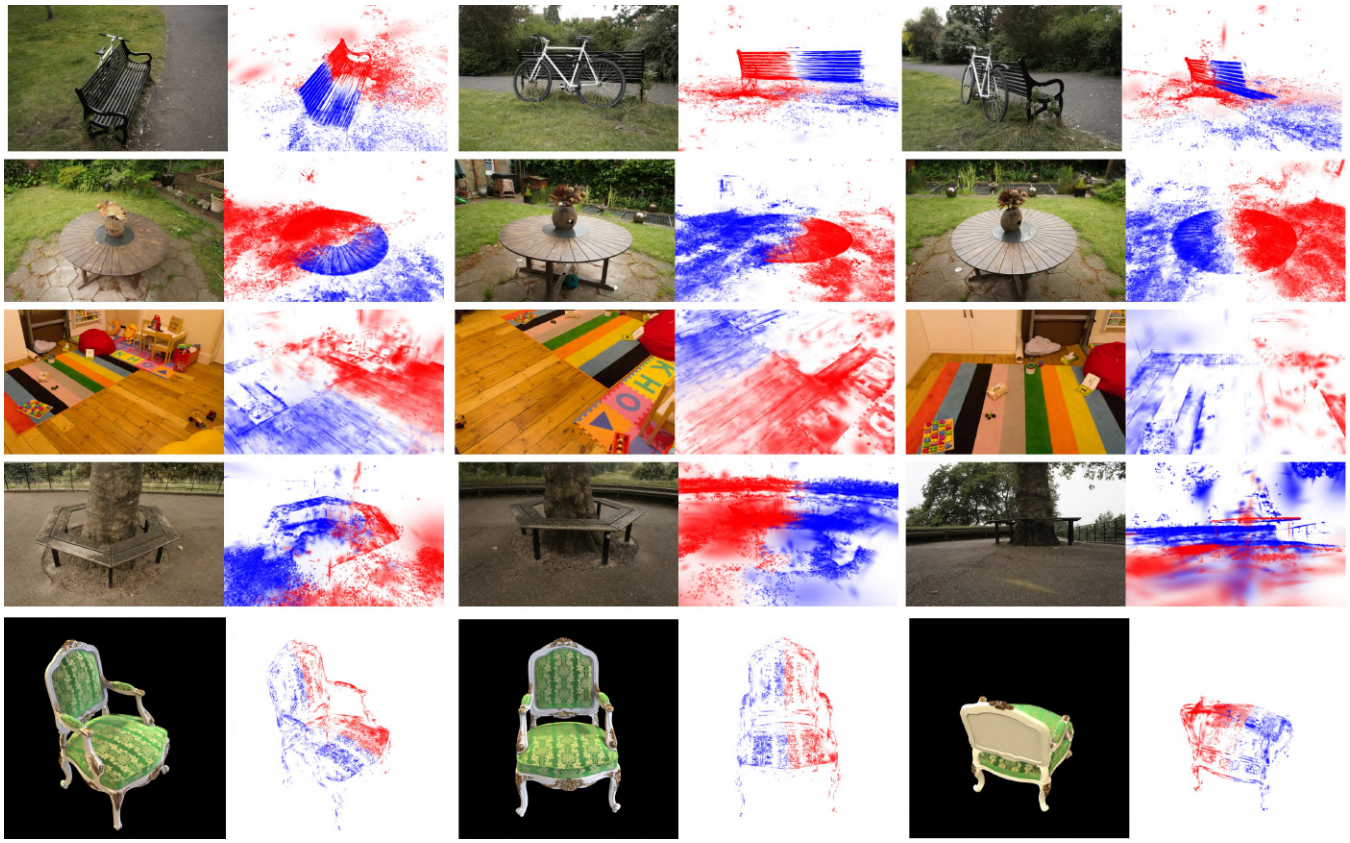

Figure 4: Top-most reflective symmetry detected in several scenes taken from diverse datasets (shown from different views).

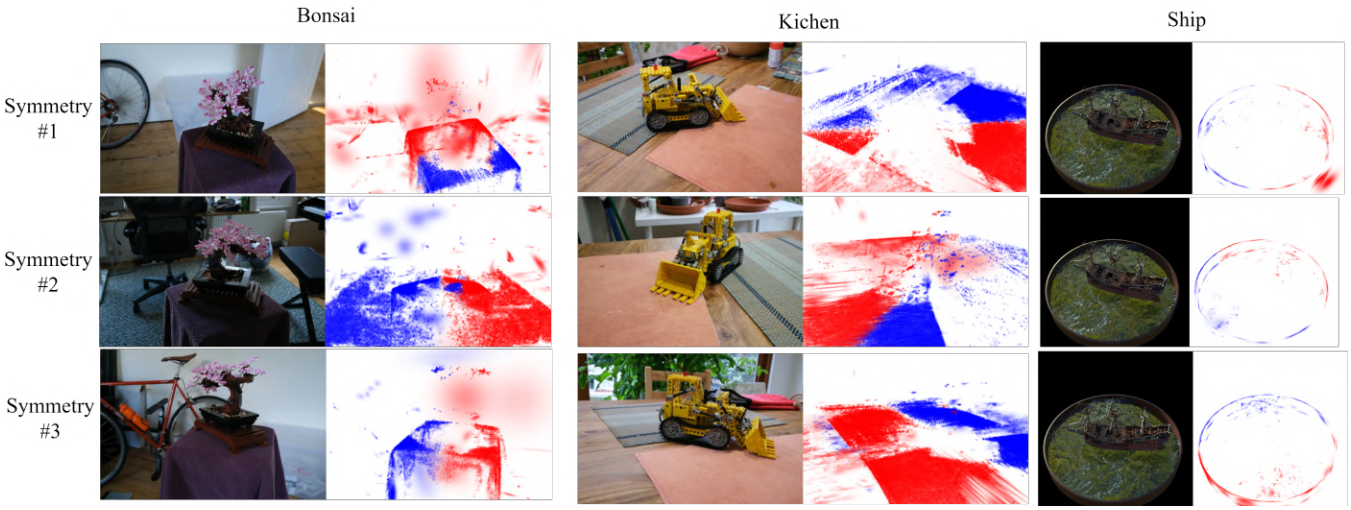

Figure 5: Multiple reflective symmetries detected in a single scene.

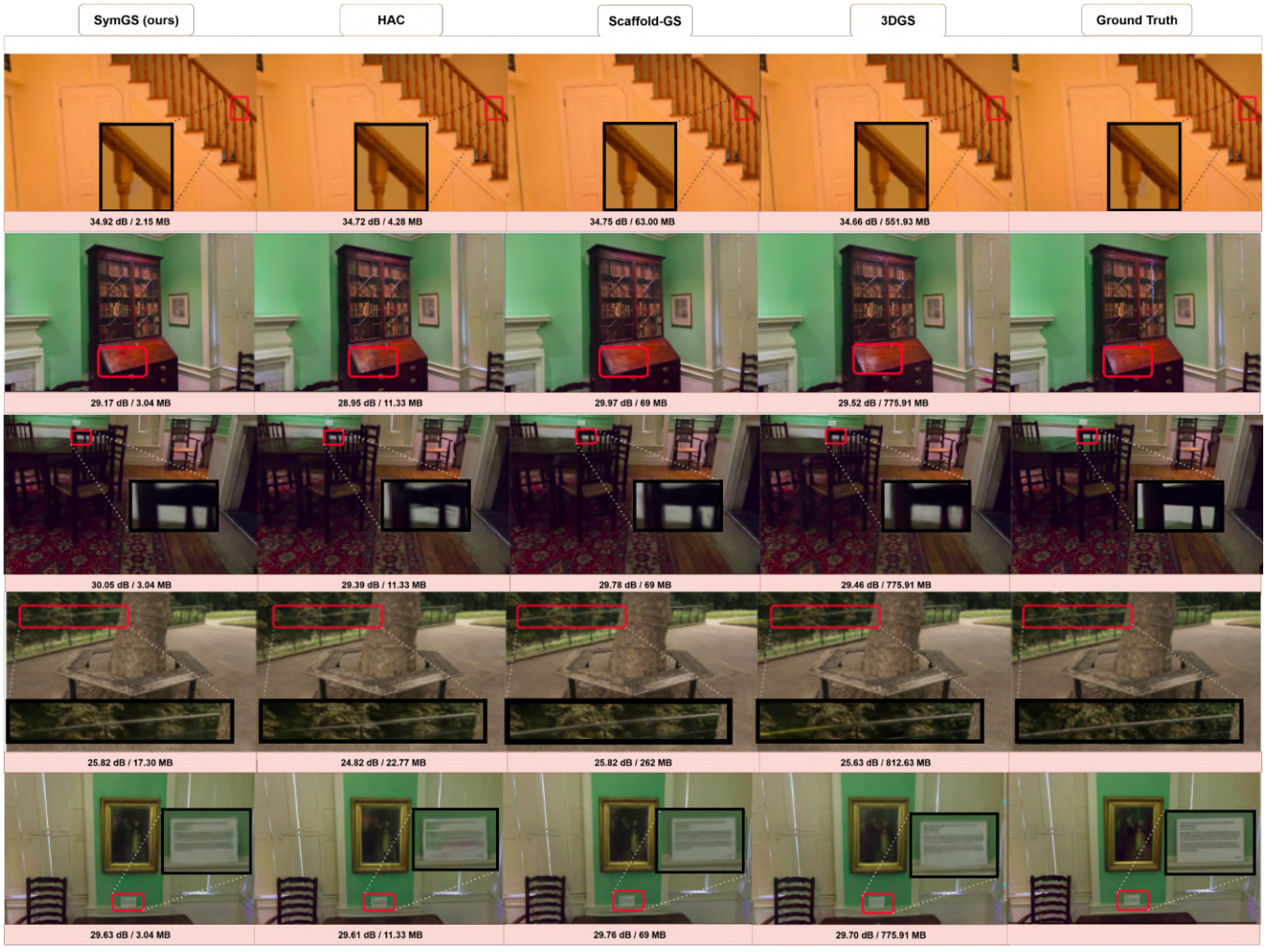

Figure 6: Qualitative Comparison with HAC, Scaffold-GS and 3DGS.

Table 1: **Per-scene results across multiple datasets: SymGS vs. HAC (reproduced results)**

Dataset	SymGS				HAC			
	PSNR↑	SSIM↑	LPIPS↓	Size↓	PSNR↑	SSIM↑	LPIPS↓	Size↓
Tanks and Temples								
truck	25.89	0.879	0.158	7.82	25.87	0.879	0.159	9.18
train	22.16	0.803	0.234	6.47	22.42	0.814	0.220	6.95
Average	**24.02**	**0.841**	**0.196**	**7.14**	**24.14**	**0.846**	**0.190**	**8.07**
DeepBlending								
playroom	30.49	0.885	0.317	2.15	30.66	0.888	0.313	4.28
drjohnson	29.51	0.893	0.303	3.04	29.04	0.884	0.311	11.33
Average	**29.99**	**0.889**	**0.310**	**2.59**	**29.85**	**0.886**	**0.312**	**7.80**
NeRF Synthetic								
chair	34.35	0.981	0.017	0.74	34.33	0.981	0.017	1.48
drums	26.35	0.951	0.043	1.46	26.05	0.947	0.046	1.94
ficus	34.93	0.985	0.014	0.78	34.50	0.983	0.015	0.97
hotdog	37.13	0.982	0.029	0.49	37.11	0.981	0.029	0.64
lego	35.09	0.978	0.023	0.92	35.18	0.979	0.022	1.26
materials	30.55	0.961	0.042	1.20	30.26	0.958	0.044	2.04
mic	35.73	0.990	0.010	0.55	35.83	0.990	0.010	0.65
ship	31.36	0.902	0.121	1.50	31.15	0.900	0.123	2.10
Average	**33.19**	**0.966**	**0.038**	**0.95**	**33.05**	**0.965**	**0.038**	**1.38**
Mip-NeRF360								
bicycle	24.93	0.734	0.273	24.71	24.90	0.731	0.270	30.16
garden	26.89	0.827	0.171	13.55	26.87	0.827	0.162	24.56
stump	26.54	0.754	0.278	18.29	25.72	0.714	0.298	21.58
room	31.70	0.922	0.213	3.84	31.82	0.923	0.210	5.33
counter	29.00	0.895	0.199	4.60	29.18	0.907	0.205	8.07
kitchen	30.63	0.912	0.151	5.91	31.07	0.922	0.133	8.39
bonsai	31.83	0.937	0.200	5.97	32.23	0.942	0.194	8.67
flowers	20.97	0.558	0.381	11.48	21.21	0.566	0.387	20.40
treehill	23.06	0.633	0.371	17.30	22.61	0.584	0.387	22.77
AVG	**27.29**	**0.797**	**0.248**	**11.74**	**27.29**	**0.790**	**0.250**	**16.66**
BungeeNeRF								
amsterdam	26.70	0.864	0.227	13.08	26.69	0.863	0.226	23.89
bilbao	27.38	0.864	0.232	9.31	27.69	0.865	0.230	19.31
hollywood	23.79	0.738	0.352	11.29	24.14	0.740	0.350	16.80
pompidou	24.86	0.827	0.266	9.13	25.16	0.829	0.265	25.26
quebec	29.31	0.920	0.192	9.63	29.34	0.918	0.194	15.48
rome	25.42	0.835	0.251	9.34	25.60	0.842	0.244	20.86
Average	**26.24**	**0.841**	**0.253**	**10.29**	**26.44**	**0.843**	**0.252**	**20.28**

Table 2: **Comparison with state-of-the-art on multiple datasets.** Metrics: PSNR ↑, SSIM ↑, LPIPS ↓, Size (MB) ↓

Datasets	**Synthetic-NeRF**				**Mip-NeRF360**				**Tank & Temples**			
Methods	PSNR ↑	SSIM ↑	LPIPS ↓	Size ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Size ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Size ↓
3DGS	33.80	0.970	0.031	68.46	27.49	0.813	0.222	744.7	23.69	0.844	0.178	431.0
Lee et al.	33.33	0.968	0.034	5.54	27.08	0.798	0.247	48.80	23.32	0.831	0.201	39.43
Compressed3D	32.94	0.967	0.033	3.68	26.98	0.801	0.238	28.80	23.32	0.832	0.194	17.28
EAGLES	32.54	0.965	0.039	5.74	27.15	0.808	0.238	68.89	23.41	0.840	0.200	34.00
Light Gaussian	32.73	0.965	0.037	7.84	27.00	0.799	0.249	44.54	22.83	0.822	0.242	22.43
Morgenstern et al.	31.05	0.955	0.047	2.20	26.01	0.772	0.259	23.90	22.78	0.817	0.211	13.05
CompGS	33.09	0.967	0.036	4.42	27.16	0.808	0.228	50.30	23.47	0.840	0.188	27.97
Scaffold-GS	33.41	0.966	0.035	19.36	27.50	0.806	0.252	253.9	23.96	0.853	0.177	86.50
HAC	33.05	0.965	0.038	1.38	27.29	0.79	0.25	16.66	24.14	0.846	0.190	8.07
SymGS (Ours)	33.19	0.966	0.038	0.95	27.29	0.797	0.248	11.74	24.02	0.841	0.196	7.14

Datasets	**Deep Blending**				**BungeeNeRF**			
Methods	PSNR ↑	SSIM ↑	LPIPS ↓	Size ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Size ↓
3DGS	29.42	0.899	0.247	663.9	24.87	0.841	0.205	1616
Lee et al.	29.79	0.901	0.258	43.21	23.36	0.788	0.251	82.60
Compressed3D	29.38	0.898	0.253	25.30	24.13	0.802	0.245	55.79
EAGLES	29.91	0.910	0.250	62.00	25.24	0.843	0.221	117.1
Light Gaussian	27.01	0.872	0.308	33.94	24.52	0.825	0.255	87.28
Morgenstern et al.	28.92	0.891	0.276	8.40	-	-	-	-
Navaneet et al.	29.75	0.903	0.247	42.77	24.63	0.823	0.239	104.3
Scaffold-GS	30.21	0.906	0.254	66.00	26.62	0.865	0.241	183.0
HAC	29.85	0.886	0.312	7.80	26.44	0.843	0.252	20.28
SymGS (ours)	29.99	0.889	0.310	2.59	26.24	0.841	0.253	10.29

Table 3: Encoded size breakdown per scene of SymGS (in MB). We take one scene from each dataset. For comparison, we also mention the total size of the corresponding HAC model. Note that Total Anchors (first column) correspond to the sum of sizes of positions of the anchors corresponding to G_{left}, i.e $\mathcal{X}_{\text{ret}}$ along with the positions of the last level remaining anchors (this in all corresponds to the sum of sizes of anchor positions of Ω)

Scene	Total Anchors	Feat	Scaling	Offsets	Hash	Masks	MLPs	Mirror	SymGS Size	HAC Size
Chair	0.22	0.18	0.05	0.07	0.03	0.01	0.16	0.0081	0.74	1.48
Flowers	4.53	3.29	1.03	2.06	0.11	0.31	0.16	0.0062	11.48	20.40
Truck	2.27	2.47	1.16	1.46	0.04	0.25	0.16	0.0050	7.82	9.18
Rome	2.94	3.49	0.89	1.54	0.12	0.19	0.16	0.0046	9.34	20.86
DrJohnson	1.60	0.56	0.28	0.36	0.01	0.06	0.15	0.0040	3.04	11.33

References

Chen, Y.; Wu, Q.; Lin, W.; Harandi, M.; and Cai, J. 2024. HAC: Hash-grid Assisted Context for 3D Gaussian Splatting Compression. In *European Conference on Computer Vision*.

Chen, Y.; Wu, Q.; Lin, W.; Harandi, M.; and Cai, J. 2025. Hac++: Towards 100x compression of 3d gaussian splatting. *arXiv preprint arXiv:2501.12255*.

Fan, L.; et al. 2024. LightGaussian: Memory-Efficient 3D Gaussian Splatting with Pseudo-View Knowledge Distillation. ArXiv preprint arXiv:2403.XXXX.

Fang, G.; and Wang, B. 2024. Mini-Splatting: Representing Scenes with a Constrained Number of Gaussians. arXiv:2403.14166.

Gao, L.; Zhang, L.-X.; Meng, H.-Y.; Ren, Y.-H.; Lai, Y.-K.; and Kobbelt, L. 2020. PRS-Net: Planar Reflective Symmetry Detection Net for 3D Models. *IEEE Transactions on Visualization and Computer Graphics*, 27(6): 3007–3018.

Kerbl, B.; Kopanas, G.; Leimkühler, T.; and Drettakis, G. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics (TOG)*.

Kobsik, G.; Lim, I.; and Kobbelt, L. 2023. Partial Symmetry Detection for 3D Geometry using Contrastive Learning with Geodesic Point Cloud Patches. *arXiv preprint arXiv:2312.08230*.

Lee, J. H.; et al. 2024. EAGLES: Efficient Attribute Gaussian Latent Encoding with Splatting. ArXiv:2404.XXXX.

Li, R.-W.; Zhang, L.-X.; Li, C.; Lai, Y.-K.; and Gao, L. 2023. E3Sym: Leveraging E(3) invariance for unsupervised 3D planar reflective symmetry detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 14543–14553.

Lipman, Y.; Chen, X.; Daubechies, I.; and Funkhouser, T. 2010. Symmetry factored embedding and distance. *ACM Transactions on Graphics (TOG)*, 29(4): 1–12.

Lu, T.; Yu, M.; Xu, L.; Xiangli, Y.; Wang, L.; Lin, D.; and Dai, B. 2024. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 20654–20664.

Mallick, S. S.; Goel, R.; Kerbl, B.; Carrasco, F. V.; Steinberger, M.; and Torre, F. D. L. 2024. Taming 3DGS: High-Quality Radiance Fields with Limited Resources. arXiv:2406.15643.

Mao, Y.; et al. 2024. CodecGS: Tri-Plane Encoding with Video Compression for 3DGS. ArXiv preprint arXiv:2404.XXXX.

Martinet, A.; Soler, C.; Holzschuch, N.; and Sillion, F. X. 2006. Accurate detection of symmetries in 3D shapes. *ACM Transactions on Graphics (TOG)*, 25(2): 439–464.

Mitra, N. J.; Guibas, L. J.; and Pauly, M. 2006. Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics (ToG)*, 25(3): 560–568.

Mitra, N. J.; Pauly, M.; Wand, M.; and Ceylan, D. 2013. Symmetry in 3D geometry: Extraction and applications. *Computer Graphics Forum*, 32(6): 1–23.

Morgenstern, W.; et al. 2024. Self-Organizing Gaussians for 3DGS Compression. *CVPR*.

Navaneet, K.; Pourahmadi Meibodi, K.; Abbasi Koohpayegani, S.; and Pirsiavash, H. 2024. Compgs: Smaller and faster gaussian splatting with vector quantization. In *European Conference on Computer Vision*, 330–349. Springer.

Niedermayr, A.; et al. 2024. MesonGS: Octree-based Compression for Gaussian Splatting. CVPR 2024.

Papantonakis, P.; Kopanas, G.; Kerbl, B.; Lanvin, A.; and Drettakis, G. 2024. Reducing the Memory Footprint of 3D Gaussian Splatting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 7(1): 1–17.

Placeholder, N.; et al. 2024. RDGS: Revising Densification for Gaussian Splatting.

Podolak, J.; Shilane, P.; Golovinskiy, A.; Rusinkiewicz, S.; and Funkhouser, T. 2006. A planar-reflective symmetry transform for 3D shapes. In *ACM Transactions on Graphics (TOG)*, volume 25, 549–559.

Ren, K.; Jiang, L.; Lu, T.; Yu, M.; Xu, L.; Ni, Z.; and Dai, B. 2024. Octree-GS: Towards Consistent Real-time Rendering with LOD-Structured 3D Gaussians. arXiv:2403.17898.

Smith, A.; et al. 2024. Compact3DGS: Efficient Color Encoding for 3DGS. ArXiv:2403.XXXX.

Xie, Q.; et al. 2024. RDO-Gaussian: Rate-Distortion Optimized 3D Gaussian Splatting. ArXiv preprint arXiv:2401.XXXX.

Yang, R.; Zhu, Z.; Jiang, Z.; Ye, B.; Chen, X.; Zhang, Y.; Chen, Y.; Zhao, J.; and Zhao, H. 2024. Spectrally pruned gaussian fields with neural compensation. *arXiv preprint arXiv:2405.00676*.

Zhang, J.; Zhan, F.; Xu, M.; Lu, S.; and Xing, E. 2024a. FreGS: 3D Gaussian Splatting with Progressive Frequency Regularization. arXiv:2403.06908.

Zhang, X.; et al. 2024b. Compact3D: Efficient Compression of 3D Gaussian Splatting using Sensitivity-Aware VQ. CVPR 2024.

Zhang, Z.; Hu, W.; Lao, Y.; He, T.; and Zhao, H. 2024c. Pixel-GS: Density Control with Pixel-aware Gradient for 3D Gaussian Splatting. arXiv:2403.15530.

Zhou, L.; et al. 2024. ContextGS: Hierarchical Compression for 3D Gaussian Splatting. CVPR.