
Appendices to “Gaussian Herding across Pens: An Optimal Transport Perspective on Global Gaussian Reduction for 3DGS”

A More Details about the Blockwise GMR Compaction

A.1 Introduction to Optimal Transport (OT)

The Optimal Transport (OT) problem dates back to 1781, when French mathematician *Gaspard Monge (1746–1818)* first formulated it as finding the minimal-cost way to move sand into a hole. This “transport plan” minimizes the transportation cost, hence the term “optimal transport”. The original formulation is known as Monge’s problem. In 1947, Russian economist *Leonid Kantorovich (1912–1986)* relaxed Monge’s formulation, leading to the so-called Monge-Kantorovich problem.

Specifically, let P and Q be two distributions over a metric space \mathcal{X} , let $c : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+$ be a cost function, and let the coupling

$$\Pi(P, Q) = \left\{ \pi(x, y) : \int \pi(dx, \cdot) = Q(\cdot), \int \pi(\cdot, dy) = P(\cdot) \right\}$$

be the set of joint distributions with marginals P and Q . For any cost function c , the total transportation cost induced by a plan $\pi \in \Pi(P, Q)$ is defined as

$$\mathcal{I}_c(\pi) = \int_{\mathcal{X} \times \mathcal{X}} c(x, y) \pi(dx, dy).$$

Here, $\pi(x, y)$ indicates how much “mass” is transported from x to y . The first constraint, $\int \pi(x, dy) = P(x)$, ensures that the mass at location x is spread over \mathcal{X} , while the second constraint, $\int \pi(dx, y) = Q(y)$, ensures that the destination at y receives the required mass. The OT problem seeks the optimal plan

$$\pi^* = \arg \min \{ \mathcal{I}_c(\pi) : \pi \in \Pi(P, Q) \},$$

minimizing the transportation cost. The corresponding minimal cost,

$$\mathcal{T}_c(P, Q) = \mathcal{I}_c(\pi^*),$$

induces a divergence between P and Q . This OT divergence provides a principled way to measure distributional similarity, enabling applications in density matching, generative modeling, and dimensionality reduction.

The optimal plan π^* and OT divergence typically lack closed-form solutions. Numerical algorithms [18] are used to compute the OT between discrete measures. When P and Q are discrete, the OT divergence reduces to:

Example 1 (OT Divergence Between Discrete Distributions). For $P = \sum_{i=1}^n u_i \delta_{x_i}$ and $Q = \sum_{j=1}^m v_j \delta_{y_j}$, the OT divergence is

$$\mathcal{T}_c(P, Q) = \min \left\{ \sum_{i=1}^n \sum_{j=1}^m \pi_{ij} c(x_i, y_j) : \sum_{i=1}^n \pi_{ij} = v_j, \sum_{j=1}^m \pi_{ij} = u_i \right\}. \quad (3)$$

Exact solutions can be found via linear programming, while approximations are obtained using algorithms like Sinkhorn.

698 A.2 Relationship Between Composite Transportation Divergence and OT

699 The Composite Transportation Divergence (CTD) [32, 33] extends OT to Gaussian mixtures. For two
700 mixtures $\phi_n = \sum_{i=1}^n \alpha_i \phi(\cdot; \mu_i, \Sigma_i)$ and $\phi'_m = \sum_{j=1}^m \alpha'_j \phi(\cdot; \mu'_j, \Sigma'_j)$, the CTD is:

$$\mathcal{T}_c(\phi_n, \phi'_m) = \inf \left\{ \sum_{i=1}^n \sum_{j=1}^m \pi_{ij} c(\phi(\cdot; \mu_i, \Sigma_i), \phi(\cdot; \mu'_j, \Sigma'_j)) : \sum_{j=1}^m \pi_{ij} = \alpha_i, \sum_{i=1}^n \pi_{ij} = \alpha'_j \right\},$$

701 Comparing this with the discrete OT divergence (3), the CTD treats Gaussian mixtures as discrete
702 distributions over the space of Gaussians, defining the divergence as the OT between them.

703 To illustrate, consider n warehouses and m factories in the space of Gaussian distributions \mathcal{F} . The i th
704 warehouse, at $\phi(\cdot; \mu_i, \Sigma_i)$, holds α_i units of material, while the j th factory, at $\phi(\cdot; \mu'_j, \Sigma'_j)$, requires
705 α'_j units. The cost to transport material from i to j is $c(\phi(\cdot; \mu_i, \Sigma_i), \phi(\cdot; \mu'_j, \Sigma'_j))$, and $\pi_{ij} \geq 0$
706 denotes the transported amount. The total cost under plan π is $\sum_{i,j} \pi_{ij} c(\phi(\cdot; \mu_i, \Sigma_i), \phi(\cdot; \mu'_j, \Sigma'_j))$.
707 The coupling set $\Pi(\alpha, \alpha') = \{\pi_{ij} : \sum_{j=1}^m \pi_{ij} = \alpha_i, \sum_{i=1}^n \pi_{ij} = \alpha'_j\}$ ensures: (a) correct material
708 removal from warehouses, and (b) correct delivery to factories. The OT problem seeks the plan
709 $\pi^* \in \Pi(\alpha, \alpha')$ minimizing the total cost. The minimal cost corresponds to the CTD between the
710 mixtures, quantifying the optimal transport cost between them.

711 Our compaction method leverages this interpretation, approximating a mixture with fewer Gaussians
712 by minimizing their CTD-based dissimilarity.

713 A.3 Algorithm Intuition

714 At first glance, the optimization problem in (1) is bilevel: the OT plan must be found for any candidate
715 $\{\bar{\alpha}_j, \bar{\mu}_j, \bar{\Sigma}_j\}_{j=1}^m$, and the objective must be minimized. However, as shown by Zhang et al. [17], this
716 simplifies with a clear interpretation. The column-wise marginal constraints on π are redundant, and
717 each Gaussian primitive (μ_i, Σ_i) transports its full mass to its “closest” counterpart. The optimal plan
718 thus corresponds to clustering $\{(\mu_i, \Sigma_i)\}_{i=1}^n$ into m clusters $\{\mathcal{C}_j\}_{j=1}^m$, with the original Gaussians
719 forming cluster “barycenters”. Mathematically, the simplified program is:

$$\min \left\{ \sum_{j=1}^m \sum_{i \in \mathcal{C}_j} \pi_{ij} c(\phi(\cdot; \mu_i, \Sigma_i), \phi(\cdot; \bar{\mu}_j, \bar{\Sigma}_j)) : [n] = \mathcal{C}_1 \sqcup \dots \sqcup \mathcal{C}_m \right\},$$

720 where \sqcup denotes disjoint union. The optimal plan has a closed form:

$$\pi_{ij} = \begin{cases} \alpha_i & \text{if } j = \arg \min_k c(\phi(\cdot; \mu_i, \Sigma_i), \phi(\cdot; \bar{\mu}_k, \bar{\Sigma}_k)), \\ 0 & \text{otherwise.} \end{cases}$$

721 With this, π and the reduced mixture parameters can be updated alternately, formalized in the
722 following k -means-like algorithm:

- 723 • **Assignment Step:** Each Gaussian $\phi(\cdot; \mu_i, \Sigma_i)$ is assigned to cluster \mathcal{C}_j by minimizing
724 $c(\phi(\cdot; \mu_i, \Sigma_i), \phi(\cdot; \bar{\mu}_j, \bar{\Sigma}_j))$, analogous to k -means’ nearest-centroid assignment.
- 725 • **Update Step:** For the cost function

$$c(\phi, \phi') = \|\mu - \bar{\mu}\|_2^2 + \|\Sigma - \bar{\Sigma}\|_F^2,$$

726 the parameters $(\bar{\mu}_j, \bar{\Sigma}_j)$ are updated as weighted averages:

$$\bar{\mu}_j = \frac{\sum_{i \in \mathcal{C}_j} \alpha_i \mu_i}{\sum_{i \in \mathcal{C}_j} \alpha_i}, \quad \bar{\Sigma}_j = \frac{\sum_{i \in \mathcal{C}_j} \alpha_i \Sigma_i}{\sum_{i \in \mathcal{C}_j} \alpha_i}.$$

727 This mirrors k -means’ centroid update.

728 A.4 Algorithm Convergence

729 The following theorem guarantees convergence of the CTD sequence generated by Algorithm 1. In
730 the worst case, the algorithm requires exponentially many steps, but it typically converges in about 5
731 iterations. A full proof is given in Zhang et al. [17].

Theorem 1 (Convergence of the Algorithm). Suppose $c(\cdot, \cdot)$ is continuous, and for any $\Delta > 0$ and ϕ^* , the set $\{\phi : c(\phi^*, \phi) \leq \Delta\}$ is compact under some Gaussian space metric. Let $\{\bar{\phi}_m^{(t)}\}$ be the sequence generated by the update step with initial $\{\bar{\alpha}_j^{(0)}, \bar{\mu}_j^{(0)}, \bar{\Sigma}_j^{(0)}\}$, and let $\mathcal{T}_c^{(t+1)}$ be the CTD at iteration t . Then:

1. There exists T and $\{\bar{\alpha}_j^*, \bar{\mu}_j^*, \bar{\Sigma}_j^*\}$ such that for all $t \geq T$, $\{\bar{\alpha}_j^{(t)}, \bar{\mu}_j^{(t)}, \bar{\Sigma}_j^{(t)}\} = \{\bar{\alpha}_j^*, \bar{\mu}_j^*, \bar{\Sigma}_j^*\}$ and $\mathcal{T}_c^{(t+1)} = \mathcal{T}_c^{(*)}$, where $\mathcal{T}_c^{(*)}$ is the CTD between the original mixture and $\{\bar{\alpha}_j^*, \bar{\mu}_j^*, \bar{\Sigma}_j^*\}$.
2. The limit point $\{\bar{\alpha}_j^*, \bar{\mu}_j^*, \bar{\Sigma}_j^*\}$ is a local minimum of \mathcal{T}_c .
3. An MM-based exhaustive algorithm with $O(m^n)$ complexity solves (1).

B More Background and Involved Techniques in Paper

B.1 Gaussian splatting and efficient variants

Due to its advantages in rendering speed and visual fidelity, 3DGS has been adopted in a broad range of applications, including human reconstruction, AI-generated content, autonomous driving, and beyond [34, 35, 36, 37]. Extensions to dynamic 3DGS, editable 3DGS, and surface representation have further broadened its utility [38, 39, 40]. However, the standard 3DGS suffers from significant storage, memory, and rendering costs when applied to large-scale or complex scenes. This limits its deployment on resource-constrained platforms and in broader areas.

To improve storage and rendering efficiency while maintaining visual quality, recent research has focused on two complementary directions: compression and compaction [8]. Compression methods shrink file size by exploiting redundancy in Gaussian attributes (e.g., using vector quantization to cluster similar parameters) [11, 41] or structuring Gaussians into more compact forms (e.g., grids, anchors, and shared codes) [42, 43]. On the other hand, compaction aims to reduce the number of Gaussians required for accurate rendering by optimizing their spatial distribution and pruning redundant primitives [12, 6, 5]. Hybrid pipelines such as LightGaussian [11] and Octree-GS [7] combine compression and compaction together to deliver highly compact and lightweight neural renders.

B.2 Detailed 3DGS pipeline

The advantages of 3DGS in rendering speed and image fidelity have made it applicable to a wide range of tasks, including human reconstruction, AI-generated content, autonomous driving, and beyond [34, 35, 36, 37]. Extensions to dynamic 3DGS, editable 3DGS, and surface representation have further broadened its utility [38, 39, 40]. 3D Gaussian Splatting (3DGS) represents a scene as a set of anisotropic 3D Gaussian primitives, each parameterized by its spatial location, shape, opacity, and radiance. The pipeline consists of the following key steps:

1. **Initialization.** From Structure-from-Motion (SfM), obtain calibrated camera poses and a sparse point cloud. Each point is initialized as a 3D Gaussian with an opacity α_i :

$$\phi(x; \mu_i, \Sigma_i) = |2\pi\Sigma|^{-1} \exp\left(-\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i)\right),$$

where $\mu_i \in \mathbb{R}^3$ is the position (mean), $\Sigma_i \in \mathbb{R}^{3 \times 3}$ is the covariance matrix (anisotropic shape), and $\alpha_i \in [0, 1]$.

2. **Projection & Rasterization.** Each 3D Gaussian is projected to 2D using the camera model,

$$\Sigma'_i = JW\Sigma_iJW^T,$$

where W is a view transformation matrix and J is the Jacobian of the projective transform. Rasterization is done using a differentiable splatting approach which makes optimization possible.

3. **Image Formation (Alpha Blending).** The pixel color is computed via volumetric blending:

$$C = \sum_{i=1}^N T_i \cdot \alpha_i \cdot c_i \text{ with } T_i = \prod_{j=1}^{i-1} (1 - \alpha_j),$$

772 where c_i is the SH-predicted color of the i -th Gaussian in front-to-back order.

773 4. **Optimization.** The Gaussian parameters $\{\mu_i, \Sigma_i, \alpha_i, c_i\}$ are optimized to minimize a photometric
774 loss:

$$\mathcal{L} = (1 - \lambda) \left\| \widehat{C} - C^* \right\|_1 + \lambda \cdot \mathcal{L}_{\text{SSIM}},$$

775 where \widehat{C} is the rendered image, C^* the ground truth, and $\lambda \in [0, 1]$ balances the two loss terms.

776 5. **Adaptive Densification & Pruning.** During training, Gaussians are cloned (under-reconstruction)
777 or split (over-reconstruction) based on view-space gradient magnitude, and low-contribution
778 Gaussians are pruned.

779 B.3 Involved 3DGS Variants

780 Although 3DGS features a well-designed pipeline and an efficient differentiable rasterization scheme,
781 its densification process still suffers from several limitations at the detail level. For example, the
782 non-uniform distribution of Gaussians can lead to overlapping primitives in dense regions and under-
783 reconstruction in sparse areas. In addition, the optimization tends to favor large Gaussians that
784 quickly cover the scene to achieve fast convergence. This behavior, however, often results in poor
785 reconstruction of high-frequency details and introduces geometric noise.

786 Consequently, a growing body of work has focused on refining the densification stage of the original
787 3DGS framework. In this section, we highlight two such methods—Atom-GS [13] and Mini-Splatting
788 (-D) [12]—which specifically enhance the densification process. While these methods also modify
789 other components such as pruning, we isolate and utilize only the early-stage point cloud generated
790 before 15K training iterations. Applying our compaction method to these two distinct variants further
791 illustrates its generality.

792 It is important to note that our approach is broadly applicable to any point cloud produced by 3DGS-
793 style pipelines, as long as each Gaussian retains its mean, covariance, and opacity, along with a
794 reasonable optimization strategy for opacity and color.

795 **Atom-GS** To address the tendency of 3DGS to rely on large Gaussians for rapid scene coverage,
796 AtomGS replaces the free-form Gaussian optimization in 3DGS with a structured formulation based
797 on atom Gaussians—uniform, isotropic primitives that better adhere to the underlying scene geometry.
798 In addition, AtomGS introduces geometry-guided optimization to further enhance the spatial fidelity
799 of the reconstructed scene. AtomGS builds upon two key techniques:

800 1. Atomized Proliferation.

- 801 • Constrains Gaussians in high-detail regions into fixed-size, isotropic Atom Gaussians.
- 802 • Uses a geometric schedule to refine scale and enhance surface coverage over training
- 803 iterations.
- 804 • Merges Gaussians in flat areas to reduce redundancy.

805 2. Geometry-Guided Optimization.

- 806 • Introduces Edge-Aware Normal Loss by weighting normal smoothness using an edge map
- 807 derived from the RGB image.
- 808 • Preserves sharp features while smoothing flat regions, improving alignment with the true 3D
- 809 geometry.

810 **Mini-Splatting (-D)** The original 3DGS often suffers from non-uniform Gaussian distributions,
811 resulting in excessive overlap in dense regions and insufficient reconstruction in sparse areas. Mini-
812 Splatting addresses this issue by reorganizing the spatial distribution of Gaussians through a two-stage
813 pipeline consisting of densification and simplification. Its goal is to reduce the total number of
814 Gaussians without compromising rendering quality. Mini-Splatting-D is a variant of Mini-Splatting,
815 tailored for quality-oriented rendering. While it follows a similar overall pipeline, the key difference
816 lies in the omission of the simplification stage, allowing the method to focus entirely on maximizing
817 rendering quality. In our implementation, we focus solely on the densification process used in our
818 pipeline.

- 819 • **Blur Split:** Identifies Gaussians that dominate blurry regions (via screen-space analysis) and splits
820 them to recover fine detail.

- Depth Reinitialization: Reprojects depth maps derived from midpoints between ray-ellipsoid intersections to generate dense and uniform 3D points for reinitialization.

Beyond evaluating compatibility with different 3DGS densification variants, we compare our method against recent state-of-the-art compaction techniques. Specifically, we consider two representative baselines: LightGaussian [11] and PUP-3DGS [15]. A brief overview of each method is provided below.

LightGaussian Original 3DGS overpopulates Gaussians during densification, causing storage bloat and rendering inefficiency. LightGaussian prunes redundant Gaussians based on a global significance score (computed via hit count, opacity, and normalized volume), followed by fine-tuning to recover lost details. The steps include:

1. Compute significance for each Gaussian by a score based on the volume of the Gaussian sphere and the number of covered pixels.
2. Prune low-score Gaussians.
3. Fine-tune retained Gaussians with photometric loss.

PUP-3DGS LightGaussian employs a visibility-aware importance score, which tends to favor pruning small Gaussians while retaining larger background primitives. In contrast, PUP-3DGS introduces a mathematically principled importance measure based on second-order derivatives, allowing it to preserve small but geometrically critical Gaussians in foreground regions. The technical steps are:

1. Fisher Approximation. Computes the gradients of the Gaussian parameters to approximate the Hessian matrix, which is then used to derive the sensitivity score.
2. Iterative Pruning. Rank Gaussians by sensitivity scores, prune low-sensitivity ones, and fine-tune iteratively to recover lost fidelity.

C Sensitivity Analysis

C.1 Choice of KD-tree Depth

In our algorithm, the KD-tree depth d (or equivalently, the number s of Gaussians per KD block) is treated as a hyperparameter. We investigate how rendering quality varies with different values of d on the Truck scene. The results are shown in Figure 6.

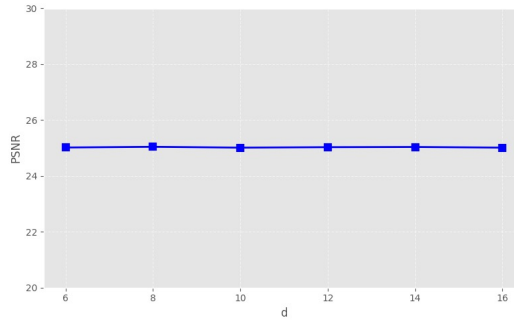


Figure 6: Sensitivity Analysis of d . Our GHAP algorithm exhibits stable rendering performance across varying KD-tree depths d .

We observe that, within a reasonable range, the KD-tree depth d has minimal impact on rendering quality. In our experiments on the Truck scene, setting $d < 6$ results in distance matrices that are too large to fit into GPU memory, causing memory overflow. Conversely, when $d > 16$, the number of partitions becomes excessively large, and most blocks contain one or zero Gaussians, rendering the algorithm inapplicable.

C.2 Sampling Iteration

Since most 3DGS-based algorithms complete the densification stage before 15K training iterations, we apply our sampling procedure after this point to achieve a target retention ratio. To study the effect of sampling at different post-densification stages, we evaluate our method at various iterations beyond 15K. The experimental results are presented in Figure 7.

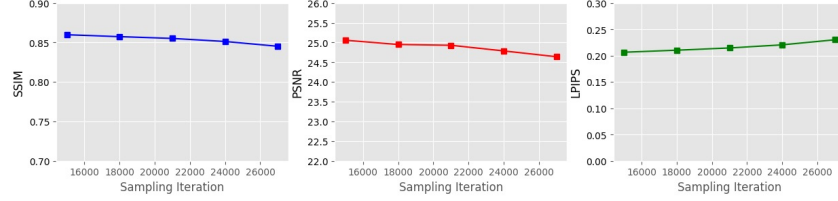


Figure 7: Sensitivity Analysis of Sampling Iteration. The rendering performance of our GHAP algorithm has negligible loss by variations in sampling iteration.

As shown in the results, sampling at any iteration beyond 15K has minimal impact on the final rendering quality. These results indicate that continuing optimization beyond 15K provides diminishing returns. Moreover, once sampling is performed, only a small number of fine-tuning iterations are needed for the compacted model to converge. Based on these observations, we recommend applying sampling uniformly at the 15K iteration, followed by an additional 15K fine-tuning iterations for all experiments.

D Additional Scene Visualizations

We report a more comprehensive set of results for the plug-in experiments in Table 4, including two additional metrics: the number of Gaussians (in thousands) and rendering speed measured in FPS. As shown in the updated results, our compaction method significantly improves FPS compared to the original models, often yielding over a 2 \times speedup in rendering performance.

In addition to the previously shown Figure 4, we present more detailed qualitative comparisons across multiple scenes in Figure 8. Our method consistently preserves the visual quality of the original models. Notably, when applied to stronger 3DGS variants with improved densification strategies—such as Mini-Splatting-D—our compaction framework performs even better. This is reflected in the fact that, after compaction, Mini-Splatting-D often achieves higher rendering quality than the original 3DGS baseline.

Table 4: Additional information in plug-in experiment.

Method-Compact Rate	ρ	Tank&Temples					MipNeRF-360					Deep Blendering				
		SSIM	PSNR	LPIPS	k Gaussians	FPS	SSIM	PSNR	LPIPS	k Gaussians	FPS	SSIM	PSNR	LPIPS	k Gaussians	FPS
3DGS-30k	10%	0.818	23.312	0.242	157	341	0.764	26.404	0.314	272	272	0.905	29.647	0.264	248	290
	20%	0.835	23.615	0.212	313	300	0.788	26.973	0.275	545	230	0.907	29.864	0.252	496	237
	100%	0.853	23.785	0.169	1577	168	0.813	27.554	0.221	2625	126	0.907	29.816	0.238	2476	126
Mini-Splatting-D	10%	0.835	23.232	0.198	147	564	0.802	27.090	0.25	318	389	0.909	30.042	0.254	156	549
	20%	0.855	23.403	0.171	293	526	0.821	27.310	0.214	636	299	0.912	30.17	0.238	318	489
	100%	0.848	23.338	0.14	1473	209	0.832	27.486	0.176	3177	111	0.907	29.980	0.211	1597	197
Atom-GS	10%	0.793	22.988	0.274	426	401	0.764	26.535	0.307	468	304	0.899	29.347	0.282	464	419
	20%	0.812	23.282	0.24	852	283	0.788	27.025	0.269	937	236	0.902	29.347	0.268	925	323
	100%	0.814	23.289	0.235	4274	87	0.796	27.135	0.251	4699	86	0.902	29.314	0.267	4625	121

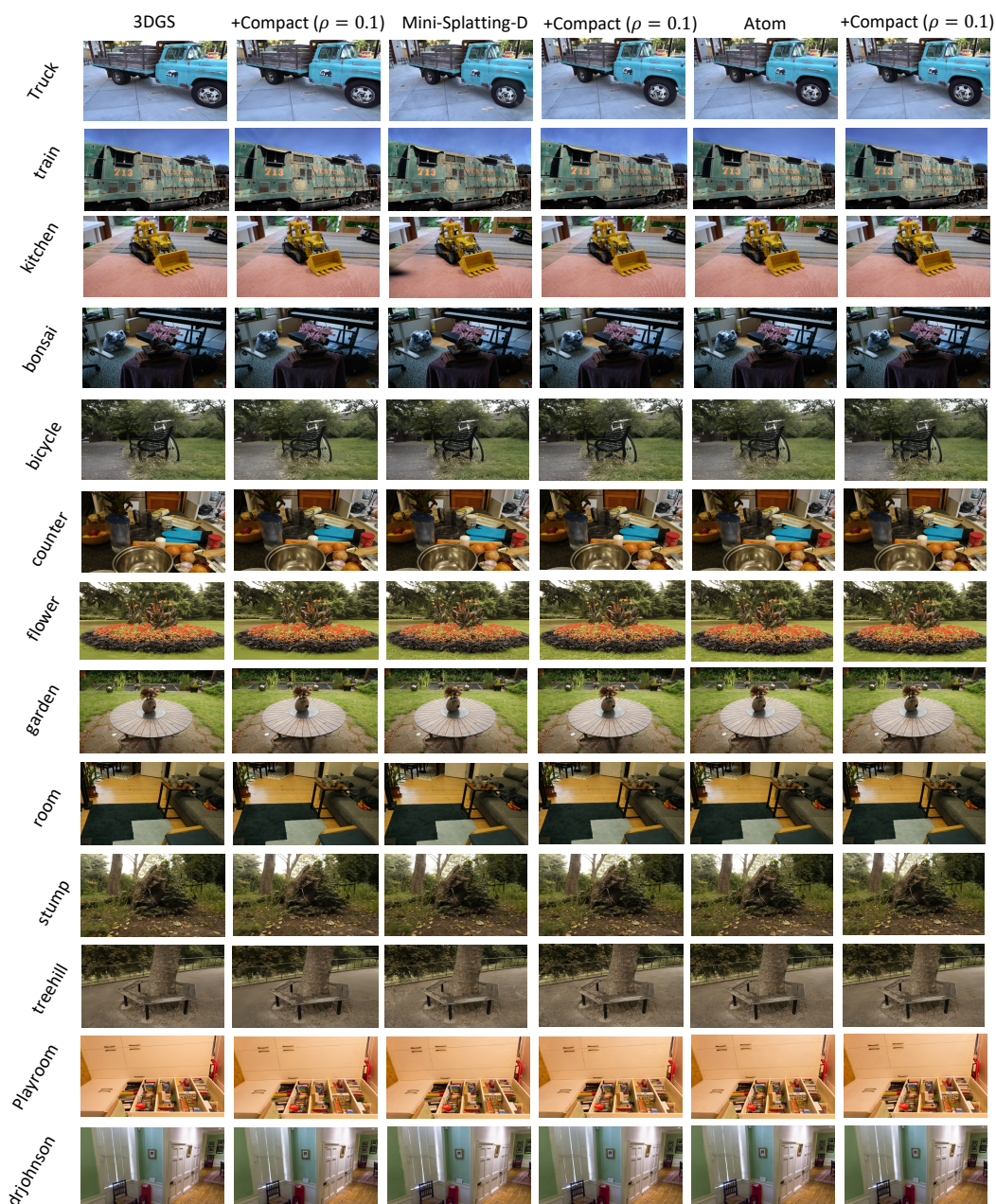


Figure 8: **More scenes visualization.** Visual comparisons across additional scenes before and after compaction.

References

- [1] Ben Fei, Jingyi Xu, Rui Zhang, Qingyuan Zhou, Weidong Yang, and Ying He. 3D Gaussian splatting as new era: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 2024.
- [2] Guikun Chen and Wenguan Wang. A survey on 3D Gaussian splatting. *arXiv preprint arXiv:2401.03890*, 2024.
- [3] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 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.
- [5] Michael Niemeyer, Fabian Manhardt, Marie-Julie Rakotosaona, Michael Oechsle, Daniel Duckworth, Rama Gosula, Keisuke Tateno, John Bates, Dominik Kaeser, and Federico Tombari. RadSplat: Radiance field-informed Gaussian splatting for robust real-time rendering with 900+ FPS. *arXiv preprint arXiv:2403.13806*, 2024.
- [6] Zhaoliang Zhang, Tianchen Song, Yongjae Lee, Li Yang, Cheng Peng, Rama Chellappa, and Deliang Fan. LP-3DGS: Learning to prune 3D Gaussian splatting. *Advances in Neural Information Processing Systems*, 38, 2024.
- [7] Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. Octree-GS: Towards consistent real-time rendering with LOD-structured 3D Gaussians. *arXiv preprint arXiv:2403.17898*, 2024.
- [8] Milena T Bagdasarian, Paul Knoll, Yi-Hsin Li, Florian Barthel, Anna Hilsmann, Peter Eisert, and Wieland Morgenstern. 3DGS.zip: A survey on 3D Gaussian splatting compression methods. *arXiv preprint arXiv:2407.09510*, 2024.
- [9] Kai Cheng, Xiaoxiao Long, Kaizhi Yang, Yao Yao, Wei Yin, Yuexin Ma, Wenping Wang, and Xuejin Chen. GaussianPro: 3D Gaussian splatting with progressive propagation. In *Forty-first International Conference on Machine Learning*, 2024.
- [10] Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando De La Torre. Taming 3DGS: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024.
- [11] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, Zhangyang Wang, et al. Light-Gaussian: Unbounded 3D Gaussian compression with 15x reduction and 200+ FPS. *Advances in Neural Information Processing Systems*, 37:140138–140158, 2024.
- [12] Guangchi Fang and Bing Wang. Mini-splatting: Representing scenes with a constrained number of Gaussians. In *European Conference on Computer Vision*, pages 165–181. Springer, 2024.
- [13] Rong Liu, Rui Xu, Yue Hu, Meida Chen, and Andrew Feng. AtomGS: Atomizing Gaussian splatting for high-fidelity radiance field. *arXiv preprint arXiv:2405.12369*, 2024.
- [14] Muhammad Salman Ali, Maryam Qamar, Sung-Ho Bae, and Enzo Tartaglione. Trimming the fat: Efficient compression of 3D Gaussian splats through pruning. *arXiv preprint arXiv:2406.18214*, 2024.
- [15] Alex Hanson, Allen Tu, Vasu Singla, Mayuka Jayawardhana, Matthias Zwicker, and Tom Goldstein. PUP 3D-GS: Principled uncertainty pruning for 3D Gaussian splatting. *arXiv preprint arXiv:2406.10219*, 2024.
- [16] 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, 2024.

- [17] Qiong Zhang, Archer Gong Zhang, and Jiahua Chen. Gaussian mixture reduction with composite transportation divergence. *IEEE Transactions on Information Theory*, 70(7):5191–5212, 2023.
- [18] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.
- [19] Cédric Villani. *Optimal Transport: Old and New*. Springer Berlin, Heidelberg, 2009.
- [20] Sieun Kim, Kyungjin Lee, and Youngki Lee. Color-cued efficient densification method for 3D Gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 775–783, 2024.
- [21] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18039–18048, 2024.
- [22] Yiming Ji, Yang Liu, Guanghu Xie, Boyu Ma, Zongwu Xie, and Hong Liu. NEDS-SLAM: A neural explicit dense semantic SLAM framework using 3D Gaussian splatting. *IEEE Robotics and Automation Letters*, 9(10):8778–8785, 2024.
- [23] Yanqi Bao, Tianyu Ding, Jing Huo, Yaoli Liu, Yuxin Li, Wenbin Li, Yang Gao, and Jiebo Luo. 3D Gaussian splatting: Survey, technologies, challenges, and opportunities. *IEEE Transactions on Circuits and Systems for Video Technology*, 2025.
- [24] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3D Gaussian splatting as Markov Chain Monte Carlo. *Advances in Neural Information Processing Systems*, 37:80965–80986, 2024.
- [25] Yutian Chen, Max Welling, and Alexander J. Smola. Super-samples from kernel herding. *CoRR*, abs/1203.3472, 2012.
- [26] David F Crouse, Peter Willett, Krishna Pattipati, and Lennart Svensson. A look at Gaussian mixture reduction algorithms. In *14th International Conference on Information Fusion*, pages 1–8. IEEE, 2011.
- [27] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.
- [28] Donald Meagher. Geometric modeling using octree encoding. *Computer graphics and image processing*, 19(2):129–147, 1982.
- [29] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017.
- [30] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)*, 37(6):1–15, 2018.
- [31] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022.
- [32] Yongxin Chen, Tryphon T Georgiou, and Allen Tannenbaum. Optimal transport for Gaussian mixture models. *IEEE Access*, 7:6269–6278, 2018.
- [33] Julie Delon and Agnès Desolneux. A Wasserstein-type distance in the space of Gaussian mixture models. *SIAM Journal on Imaging Sciences*, 13(2):936–970, 2020.
- [34] Shoukang Hu, Tao Hu, and Ziwei Liu. GauHuman: Articulated Gaussian splatting from monocular human videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20418–20431, 2024.

- 970 [35] Shunyuan Zheng, Boyao Zhou, Ruizhi Shao, Boning Liu, Shengping Zhang, Liqiang Nie, and
971 Yebin Liu. GPS-Gaussian: Generalizable pixel-wise 3D Gaussian splatting for real-time human
972 novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and
973 Pattern Recognition*, pages 19680–19690, 2024.
- 974 [36] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. DreamGaussian: Generative
975 Gaussian splatting for efficient 3D content creation. In *The Twelfth International Conference on
976 Learning Representations*, 2024.
- 977 [37] Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan
978 Yang. DrivingGaussian: Composite Gaussian splatting for surrounding dynamic autonomous
979 driving scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern
980 Recognition*, pages 21634–21643, 2024.
- 981 [38] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu,
982 Qi Tian, and Xinggang Wang. 4D Gaussian splatting for real-time dynamic scene rendering. In
983 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages
984 20310–20320, 2024.
- 985 [39] Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. High-
986 quality surface reconstruction using Gaussian surfels. In *ACM SIGGRAPH 2024 Conference
987 Papers*, pages 1–11, 2024.
- 988 [40] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai,
989 Lei Yang, Huaping Liu, and Guosheng Lin. GaussianEditor: Swift and controllable 3D editing
990 with Gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and
991 Pattern Recognition*, pages 21476–21485, 2024.
- 992 [41] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3D Gaussian
993 splatting for accelerated novel view synthesis. In *Proceedings of the IEEE/CVF Conference on
994 Computer Vision and Pattern Recognition*, pages 10349–10358, 2024.
- 995 [42] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-
996 GS: Structured 3D Gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF
997 Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024.
- 998 [43] Yufei Wang, Zhihao Li, Lanqing Guo, Wenhan Yang, Alex Kot, and Bihan Wen. ContextGS:
999 Compact 3D Gaussian splatting with anchor level context model. *Advances in Neural Informa-
1000 tion Processing Systems*, 37:51532–51551, 2024.