A. Appendix

A.1. Fisher Information Derivation: A Bayesian Interpretation of PUP 3D-GS

The Fisher Information is the variance of the score:

$$I(\theta) = E_\theta[(\nabla_\theta \log p(x|\theta))^2]. \tag{11}$$

Lemma 5.3 from [15] gives that this is equivalent to:

$$I(\theta) = E_\theta[-\nabla_\theta \nabla_\theta \log p(x|\theta)], \tag{12}$$

assuming $\log p(x|\theta)$ is twice differentiable and with certain regularity conditions.

To start, we reformulate our L_2 objective as a log-likelihood:

$$-\log p(\mathcal{I}|\Phi, \mathcal{G}) = E_{(I,\phi)\sim(\mathcal{I},\Phi)}[(I - I_\mathcal{G}(\phi))^T (I - I_\mathcal{G}(\phi))], \tag{13}$$

where $\mathcal{I}$ is the set of ground truth image, Φ is the set of their corresponding poses, $\mathcal{G}$ are the model Gaussian parameters, $I_\mathcal{G}(\phi)$ is the rendering function for pose ϕ, and we assume that the error follows a Gaussian distribution.

We can take the Laplace approximation of the log of the posterior distribution over the model parameters $\mathcal{G}$ on the converged scene parameters $\hat{\mathcal{G}}$ as:

$$-\log p(\mathcal{G}|\mathcal{I}, \Phi) \approx -\log p(\hat{\mathcal{G}}|\mathcal{I}, \Phi) + \frac{1}{2}(\mathcal{G} - \hat{\mathcal{G}}) H(\hat{\mathcal{G}})(\mathcal{G} - \hat{\mathcal{G}}), \tag{14}$$

where:

$$H(\hat{\mathcal{G}}) = -\nabla_\mathcal{G} \nabla_\mathcal{G} \log p(\hat{\mathcal{G}}|\mathcal{I}, \Phi). \tag{15}$$

If we assume a uniform prior, then our claimed Fisher Information matrix from Section 4.1 is precisely the Hessian $H(\hat{\mathcal{G}})$ of this posterior.

From this formulation of our Fisher Information matrix, Proposition 3.5 from [12] gives that the log determinant of the Fisher as the entropy of the second order approximation of $p(\mathcal{G}|\mathcal{I}, \Phi)$ around $\hat{\mathcal{G}}$. If we restrict the posterior to a particular Gaussian's parameters $\mathcal{G}_i$, giving $p(\mathcal{G}_i|\mathcal{I}, \Phi)$, the log determinant of the block diagonal element corresponding to this Gaussian is a measure of entropy for that particular Gaussian. This interpretation gives our pruning scores as a ranking of the Gaussians by their entropy on this posterior.

A.2. Ablation on Patch Size

We ablate our choice of patch size in Table 6. Notice that, although the 4×4 patches that we use in our experiments produce slightly better image quality metrics, the 2×2 and 8×8 patches also produce similar results.

A.3. Ablation on Per-Round Pruning Percentages

Figure 8 plots the average metrics for each permutation of per-round pruning percentages that results in approximately 90% total pruning across all scenes in the Mip-NeRF 360 dataset. Similar to the *bicycle* scene evaluated in Figure 7, pruning 80% of Gaussians in the first round and then 50% in the second optimizes image quality and rendering speed at exactly 90% total pruning. Our method outperforms LightGaussian across all metrics and per-round pruning percentage permutations.

Table 6. Mean PSNR, SSIM, LPIPS, FPS, and point cloud size for the Mip-NeRF 360 dataset using our sensitivity score computed with 2×2, 4×4 and 8×8 patches.

Methods	Mip-NeRF 360				
	PSNR↑	SSIM↑	LPIPS↓	FPS↑	Size (MB)↓
3D-GS	27.47	0.8123	0.2216	83.88	746.46
2×2	26.46	**0.7869**	0.2742	**218.59**	74.65
4×4 (Ours)	**26.67**	0.7862	**0.2719**	204.81	74.65
8×8	26.53	0.7775	0.2780	189.23	74.65

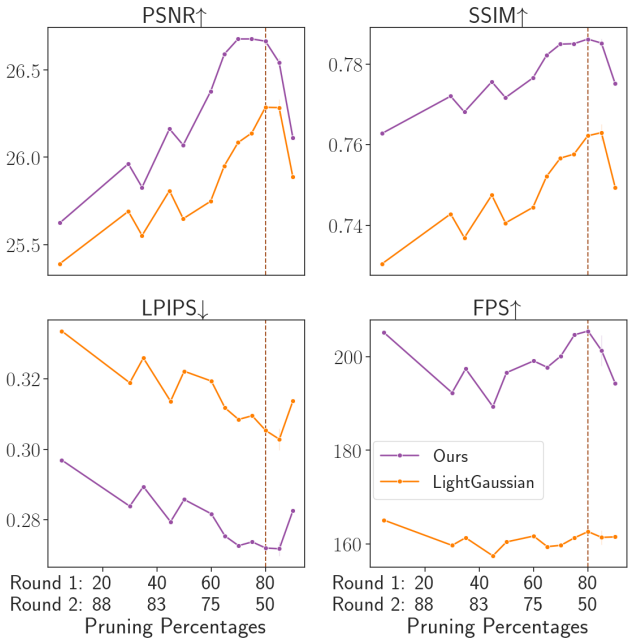

Figure 8. The average PSNR, SSIM, LPIPS, and FPS across the MipNeRF-360 dataset scene after pruning approximately 90% of Gaussians with different per-round percentages in our two-round pipeline. The dotted red line denotes our chosen per-round pruning percentages of 80% then 50%.

A.4. Scene Evaluations

PSNR, SSIM, LPIPS, and FPS for each scene from the Mip-NeRF 360, Tanks&Temples, and Deep Blending datasets that was used in 3D-GS [11] are recorded in Tables 7, 8, 9, and 10, respectively. Note that the sizes of the pruned scenes in this section are identical because exactly 90% of Gaussians were removed from each of them using our two step prune-refine method. FPS is collected using a Nvidia RTXA4000 GPU.

Table 7. PSNR on each scene after two steps of prune-refine.

Methods	Mip-NeRF 360									Tanks&Temples		Deep Blending	
	bicycle	bonsai	counter	flowers	garden	kitchen	room	stump	treehill	train	truck	drjohnson	playroom
Baseline (3D-GS)	25.09	32.25	29.11	21.34	27.28	31.57	31.51	26.55	22.56	22.10	25.43	28.15	29.81
LightGaussian	24.34	29.64	27.57	20.70	25.78	29.45	30.65	25.88	**22.49**	**21.35**	**24.81**	27.73	29.29
Ours	**24.72**	**30.64**	**28.00**	**20.86**	**26.23**	**29.83**	**31.03**	**26.30**	22.39	21.03	24.40	**28.00**	**29.71**

Table 8. SSIM on each scene after two steps of prune-refine.

Methods	Mip-NeRF 360									Tanks&Temples		Deep Blending	
	bicycle	bonsai	counter	flowers	garden	kitchen	room	stump	treehill	train	truck	drjohnson	playroom
Baseline (3D-GS)	0.7467	0.9457	0.9140	0.5875	0.8558	0.9317	0.9255	0.7687	0.6352	0.8134	0.8782	0.8778	0.8854
LightGaussian	0.6801	0.8921	0.8562	0.5343	0.7833	0.8830	0.8989	0.7256	0.5956	0.7349	0.8512	0.8546	0.8747
Ours	**0.7270**	**0.9261**	**0.8917**	**0.5548**	**0.8189**	**0.9128**	**0.9152**	**0.7570**	**0.6248**	**0.7600**	**0.8541**	**0.8762**	**0.8861**

Table 9. LPIPS on each scene after two steps of prune-refine.

Methods	Mip-NeRF 360									Tanks&Temples		Deep Blending	
	bicycle	bonsai	counter	flowers	garden	kitchen	room	stump	treehill	train	truck	drjohnson	playroom
Baseline (3D-GS)	0.2442	0.1811	0.1838	0.3602	0.1225	0.1165	0.1973	0.2429	0.3460	0.2077	0.1476	0.2895	0.2823
LightGaussian	0.3338	0.2568	0.2801	0.4258	0.2407	0.2042	0.2625	0.3132	0.4315	0.3227	0.2041	0.3383	0.3201
Ours	**0.2965**	**0.2281**	**0.2297**	**0.4211**	**0.1997**	**0.1545**	**0.2278**	**0.2836**	**0.4062**	**0.2967**	**0.1916**	**0.3067**	**0.2963**

Table 10. FPS on each scene after two steps of prune-refine. Results were collected with a Nvidia RTXA4000 GPU.

Methods	Mip-NeRF 360									Tanks&Temples		Deep Blending	
	bicycle	bonsai	counter	flowers	garden	kitchen	room	stump	treehill	train	truck	drjohnson	playroom
Baseline (3D-GS)	32.32	106.65	79.70	66.48	37.60	64.46	76.31	54.04	59.08	114.45	81.26	56.79	76.79
LightGaussian	110.94	208.15	168.48	182.28	133.77	165.98	172.46	169.98	147.03	378.50	279.55	206.94	261.27
Ours	**127.85**	**289.21**	**222.77**	**205.21**	**164.79**	**237.66**	**231.63**	**179.38**	**184.78**	**494.76**	**287.44**	**284.81**	**318.06**

A.5. Prune-Refining EAGLES

In Table 11, we ablate our PUP 3D-GS pipeline against LightGaussian's pipeline on an EAGLES [6] model of the Mip-NeRF 360 *bicycle* scene. Notice that the base EAGLES model produces similar metrics to vanilla 3D-GS despite being $2.51\times$ smaller than it. By applying PUP 3D-GS to the EAGLES model, we further reduce its size to $25.14\times$ smaller than the vanilla 3D-GS model while achieving better image quality and rendering speed than LightGaussian.

Table 11. Results from training EAGLES [6] on the *bicycle* scene and then running two steps of prune-refine with our and LightGaussian's methods.

Methods	Mip-NeRF 360 Bicycle Scene				
	PSNR↑	SSIM↑	LPIPS↓	FPS↑	Size (MB)↓
3D-GS	25.09	0.7467	0.2442	32.12	1345.58
Baseline (EAGLES)	25.07	0.7508	0.2433	47.82	535.21
EAGLES + LightGaussian	23.57	0.6082	0.4039	109.26	53.52
EAGLES + Ours	**24.01**	**0.6686**	**0.3566**	**144.56**	53.52

A.6. Comparison with Other Pruning Methods

Several other papers also introduce pruning techniques. Compact-3DGS[14] reports that they prune 58.7% of Gaussians from the Mip-NeRF 360 *bonsai* scene, slightly improving PSNR from 29.87 to 29.91. After pruning 58.7% of Gaussians with a single round of prune-refine, our method produces a much larger PSNR boost from 32.25 to 32.55. Mini-Splatting [5] and RadSplat [24] achieve higher rendering quality through orthogonal methods like improved densification and pretrained NeRF models. Since our pruning approach can be combined with these methods, it is unclear if a direct comparison with their results is useful. As reported in Table 2, our $10\times$ reduction in the number of Gaussians is significantly higher than the $2.28\times$ and $6.94\times$ reductions reported by Compact-3DGS and Mini-Splatting.

A.7. Potential Directions for Future Research

The theory behind our Hessian approximation relies on a set of views whose L1 loss is close to zero, so we compute our spatial sensitivity score on the training views after

optimization to remain as mathematically principled as possible. However, our score (1) does not rely on ground-truth data and (2) can be computed when the L1 loss is sufficiently low earlier in the 3D-GS training pipeline. These properties may provide a useful first step for several potential directions for future research.

Further optimizations and/or sufficiently powerful hardware could allow our sensitivity score to be used to prune the model during training. This is a promising research direction that could potentially lead to even higher compression and rendering speeds. Additionally, it may be possible to identify a subset of pixels, rays, or views that would produce a score that is most effective for pruning. Since this is a non-trivial, combinatorially difficult problem and our pruning score can be computed across the entire training set in seconds, we also leave this as an open research question.

Our score can be extended to other 3D Gaussian pipelines such as the Mini-Splatting and PGSR pipelines [2, 5]. Some considerations must be made, such as the effect of each Gaussian on the depth and normal maps produced by these approaches along with the rendered image. Nevertheless, the math in our main paper holds if we include the depth and normal maps as additional channels. Pruning anchor-based approaches like Scaffold-GS [17] and derived works like Octree-GS [26] is a more difficult problem. Directly pruning Gaussians with our current pipeline will not work because a fixed number of Gaussians are produced for each anchor point; pruning anchor points is ill-advised because they are generated in areas of the scene that do not have sufficient geometry. While our PUP 3D-GS approach presents a method for directly quantifying a Gaussian's importance in the scene, determining how to map that score to pruning anchors is an open research problem.

FisherRF [9] also computes Fisher information for 3D Gaussian Splats, but uses it to perform active-view selection and post-hoc uncertainty visualization instead of pruning. Furthermore, FisherRF only approximates the *diagonal* of the Fisher matrix and uses the color parameters of the Gaussians, whereas our approach uses the spatial mean and scaling parameters to compute a more accurate *block-wise* approximation. Our sensitivity pruning score can be directly repurposed for these applications, but we leave this to future work because it is not the focus of our paper.

A.8. Additional Scene Visualizations

Figure 9 provides a visual comparison of the ground truth image against renderings from the 3D-GS model before and after pruning with our PUP 3D-GS and LightGaussian's pipelines. Notice that our method consistently achieves higher visual fidelity and retains more salient foreground information like individual leaves and legible text.

A.9. Background Degradation

At the extreme pruning ratios used in our work, we observe some background degradation in pruned models. Intuitively, background regions that are observed from fewer viewpoints exhibit higher uncertainty than well-observed foreground regions, making them more susceptible to pruning. Given our aggressive pruning threshold of 90%, lossless compression is not expected. However, our method prioritizes preserving fine details in the foreground while pruning less important Gaussians in the background to maintain overall visual quality. We do not consider this a failure case because retaining foreground details over background details is often preferable.

In contrast, LightGaussian exhibits the opposite tendency, often favoring background preservation over foreground fidelity. Figure 10 illustrates this difference by comparing L1 residuals against renderings from the base 3D-GS model and highlighting instances where PUP 3D-GS degrades the background more than LightGaussian. A potential strategy for mitigating this trade-off is to reweight the background Gaussian scores using masking, incorporating a tunable parameter to control the balance between foreground and background degradation. We leave this exploration to future work.

A.10. Scene Residuals

Figures 11 and 12 provide visual comparisons of the L1 residual of renderings from the 3D-GS model before and after pruning 90% of Gaussians with our PUP 3D-GS and LightGaussian's pipelines. Figure 11 compares the L1 residuals with respect to the ground truth image, while Figure 12 compares them with respect to renderings from the base 3D-GS model. In both cases, our PUP 3D-GS pipeline produces less L1 error and retains more salient foreground information than LightGaussian's.

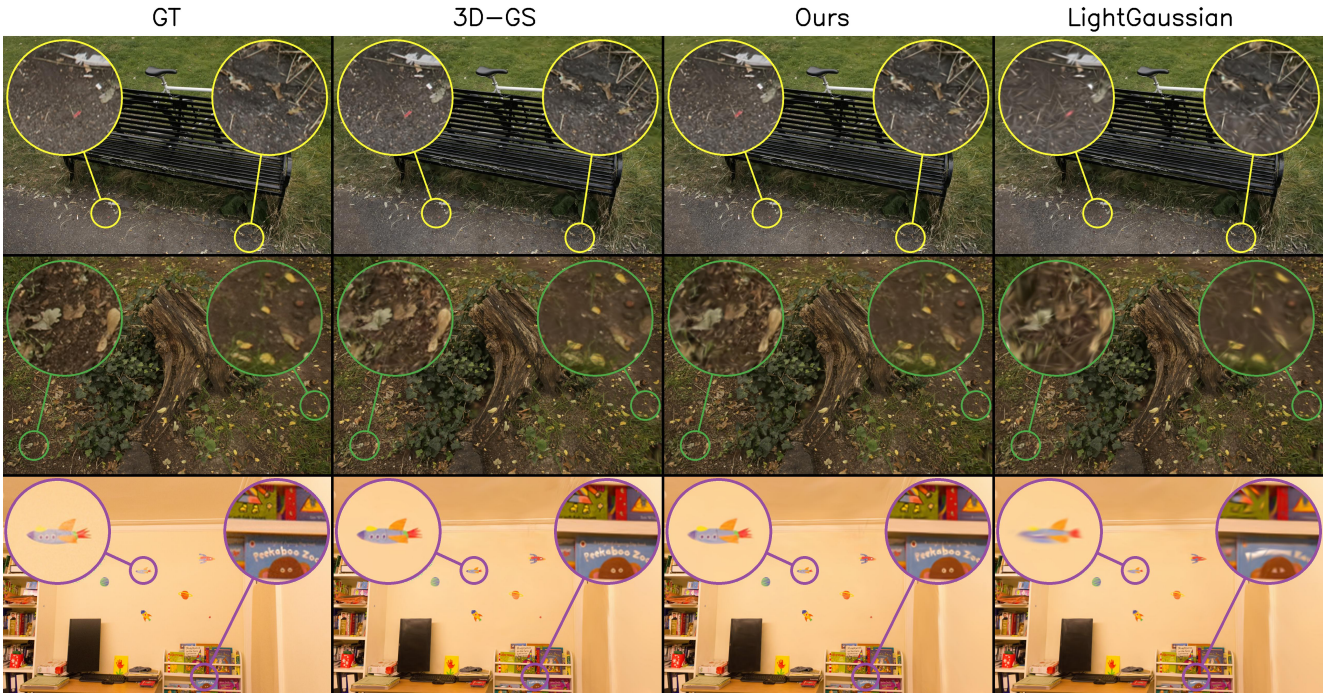

Figure 9. Visual comparison after two rounds of prune-refine using our method and LightGaussian's method. Top: *bicycle* from the Mip-Nerf 360 dataset. Middle: *stump* from the Mip-Nerf 360 dataset. Bottom: *playroom* from the Deep Blending dataset. A larger example image of *playroom* can be found in Figure 1.

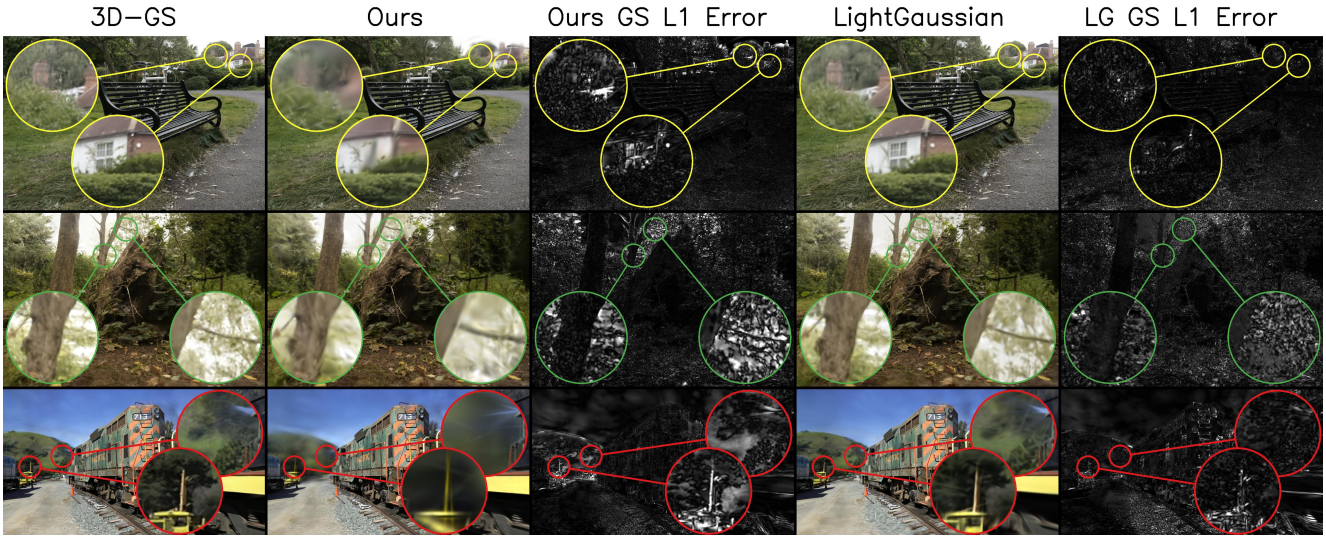

Figure 10. Cases where our method produces more background L1 error against renderings from the base 3D-GS model than LightGaussian. Columns "3D-GS", "Ours", and "LightGaussian" are the same as in Figure 4. Columns "Ours GS L1 Error" and "LG GS L1 Error" are the L1 error images of our method and LightGaussian against the original 3D-GS reconstruction of the scene. Our method prioritizes retaining visual quality and fine details in the foreground over preserving less important information in the background.

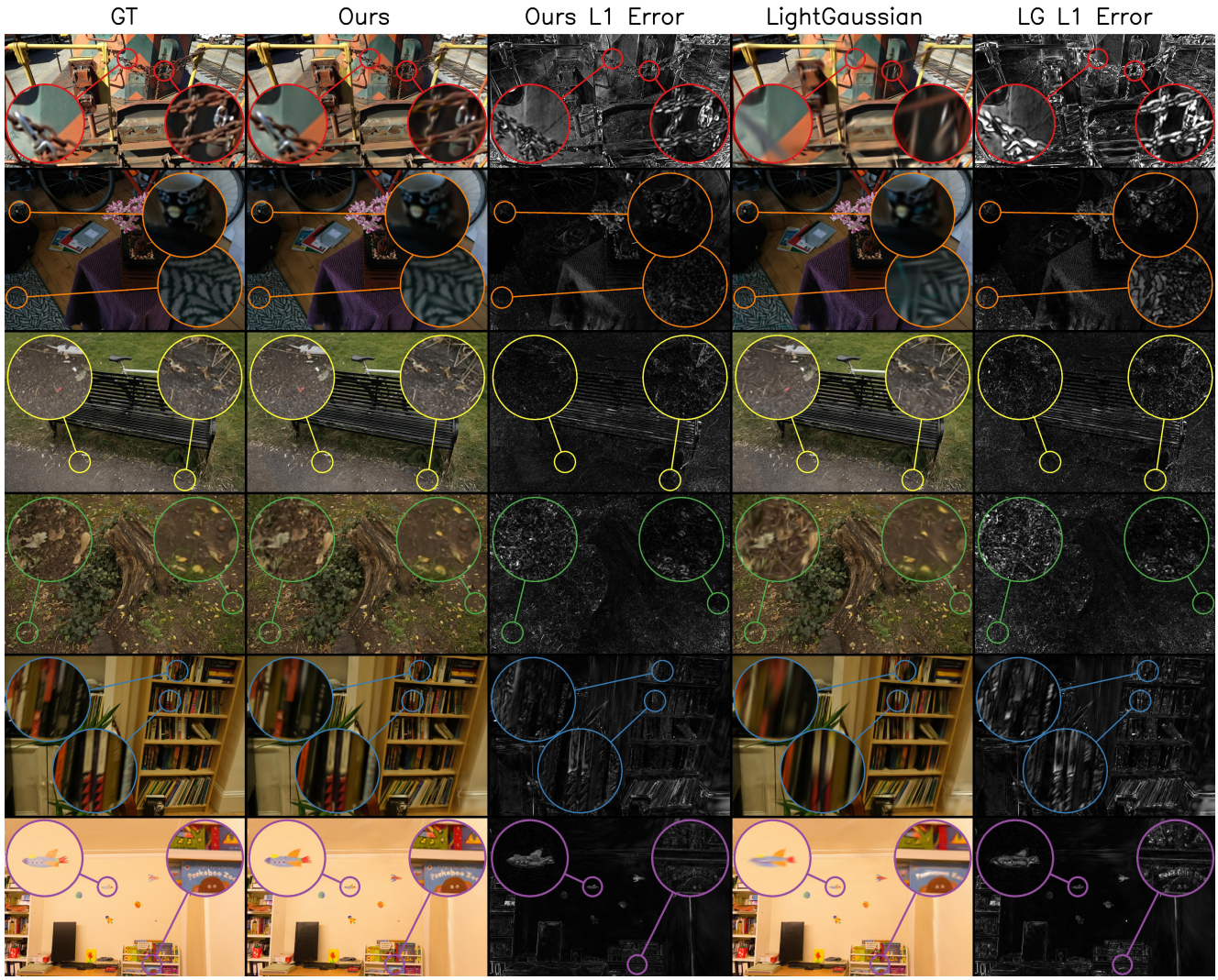

Figure 11. Visual comparison after two rounds of prune-refine using our method and LightGaussian's with additional L1 error visualizations against the ground truth images. Columns "GT", "Ours", and "LightGaussian" are the same as in Figure 4. Columns "Ours L1 Error" and "LG L1 Error" are the L1 Error images of our method and LightGaussian against the ground truth images. Our method produces lower error than LightGaussian in all examples.

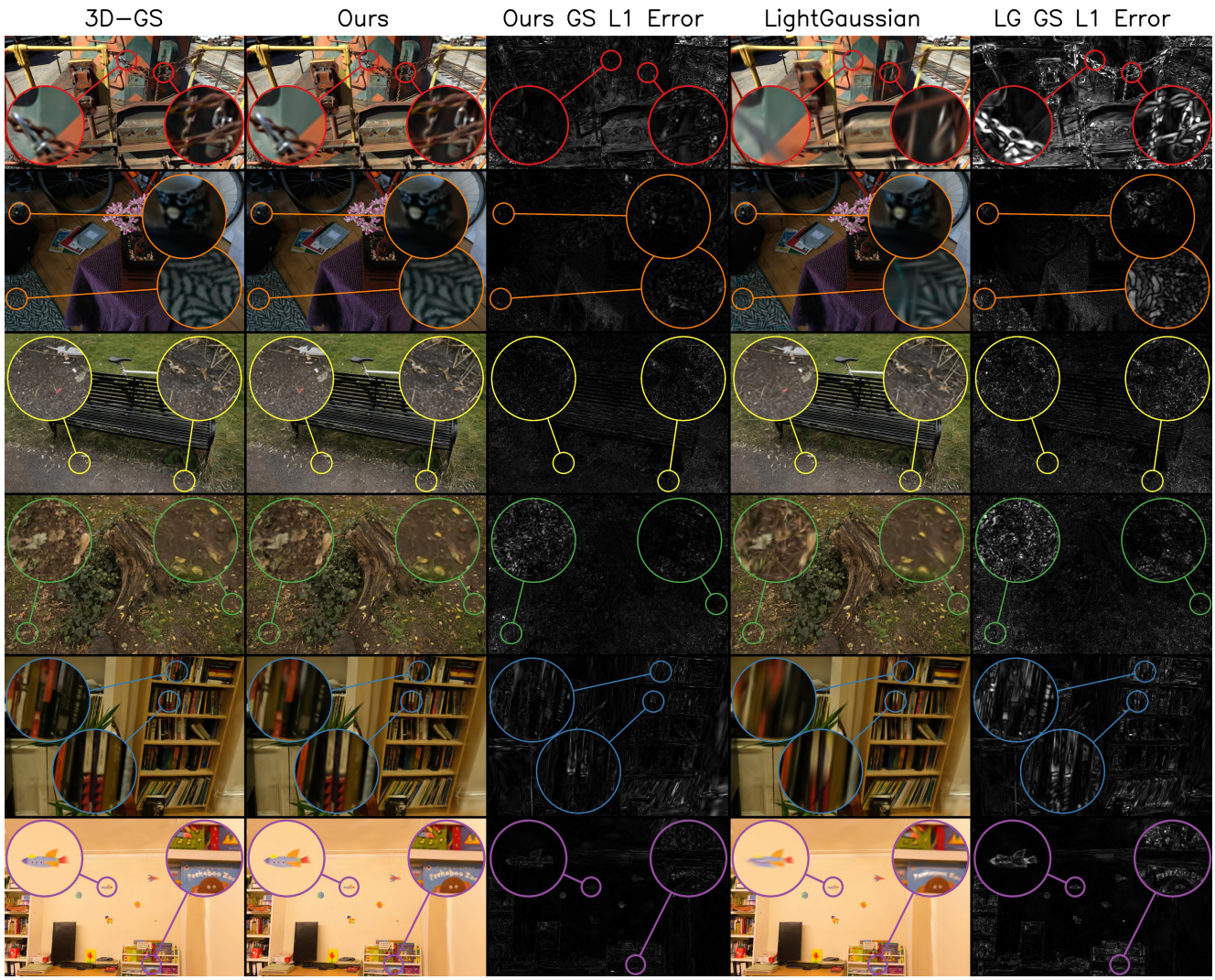

Figure 12. Visual comparison after two rounds of prune-refine using our method and LightGaussian's with additional L1 error visualizations against renderings from the base 3D-GS model. Columns "3D-GS", "Ours", and "LightGaussian" are the same as in Figure 4. Columns "Ours GS L1 Error" and "LG GS L1 Error" are the L1 error images of our method and LightGaussian against the original 3D-GS reconstruction of the scene. Our method produces lower error than LightGaussian in all examples.