Spatial Pruning in 3D Gaussian Splatting

Rahul Rustagi* Nakul Singh*
Georgia Tech College of Engineering
School of Electrical and Computer Engineering
{rrustagi7, nsingh360}@gatech.edu

Abstract

3D Gaussian Splatting (3DGS) has emerged as a powerful technique for high-fidelity scene representation, offering faster training times compared to Neural Radiance Fields (NeRF). However, these representations often result in high memory consumption and are computationally heavy for downstream robotics applications such as collision checking. To address this, we propose a "Spatial Pruning" approach that exploits spatial information. Our method introduces a spatially varying sparsity penalty and utilizes 3D occupancy grids to guide pruning. We demonstrate that our approach, specifically using 2D regularization and 3D occupancy grid-based pruning, achieves competitive visual fidelity compared to the original Gaussian Splatting on the Tanks & Temples dataset while effectively reducing model complexity to 0.71 times the original model size. [1]

1. Introduction

The representation of 3D scenes from 2D imagery has seen a paradigm shift with the advent of differentiable rendering techniques. Early approaches relied on fully continuous representations, such as Coordinate Multi-Layer Perceptrons (MLPs) utilized in Neural Radiance Fields (NeRF) [9]. These methods map spatial coordinates (x, y, z) to color and density values (r, g, b, σ), offering photorealistic results but suffering from slow training and inference times due to the costly volumetric ray-marching required. Conversely, fully discrete methods like voxel grids and surface meshes offer speed but often lack the representational flexibility to handle complex, semi-transparent scene details [2].

Bridging this gap, hybrid models such as KiloNeRF [10] and tensor factorization methods have attempted to balance fidelity with speed. However, 3D Gaussian Splatting (3DGS) [4] has recently established itself as a new state-of-the-art by representing scenes as a cloud of millions of anisotropic 3D Gaussians. This explicit representation allows for rasterization-based rendering, which is significantly faster than volumetric ray-marching.

Despite its success, 3DGS is not without limitations. The standard optimization process is prone to over-reconstruction, often placing thousands of small, redundant Gaussians in flat, textureless regions to minimize photometric error, even when fewer primitives would suffice [1]. Furthermore, the lack of explicit empty-space skipping leads to the generation of "floaters", artifacts in empty space that degrade visual quality and waste memory. For robotics applications, where collision checking and memory constraints are critical, these inefficiencies render standard 3DGS models suboptimal.

To address these challenges, we introduce a method to incorporate explicit spatial information into the pruning and optimization loops. We observe that models waste computational resources in empty space. Therefore, we propose a "Spatial Pruning" strategy that leverages:

1. **Spatially Varying Regularization:** A 2D gradient-based penalty that enforces lower number of gaussians in flat regions while preserving detail in textured areas.
2. **Occupancy-Aware Pruning:** Integration of NerfAcc [7] to maintain a dynamic occupancy grid, allowing for the aggressive removal of Gaussians in empty space during optimization.

Our contributions demonstrate that it is possible to achieve competitive visual metrics compared to the original 3DGS implementation (PSNR, SSIM, LPIPS) while having a significantly lower model size.

Motivation: The Over-Reconstruction Problem. Standard 3DGS optimization densifies primitives based on positional gradients of the photometric loss. While effective for detailed textures, this heuristic fails in flat or textureless regions where gradients are dominated by high-frequency noise rather than geometric signal. Consequently, the optimizer "over-reconstructs" these areas by spawning thousands of minute, redundant Gaussians to fit the noise [1, 6]. Recent studies have empirically validated this inefficiency; for instance, *Trimming the Fat* [1] demonstrates that up to 75% of gaussians in standard 3DGS models are redundant

[1]* indicates equal contribution

and can be removed without visual degradation.

2. Related Work

Neural Scene Representations: The landscape of scene representation is vast. Continuous methods [9] utilize neural networks to implicitly define geometry and appearance. Discrete methods, such as pixel/voxel grids, offer direct access to geometry but scale poorly with resolution. Hybrid approaches like K-Planes [2] utilize tensor factorization to decompose space and time, yet often retain high computational costs for training.

3D Gaussian Splatting: 3DGS replaces MLPs with explicit 3D Gaussians defined by position, covariance, color, and opacity [4]. Rendering is performed via a differentiable tile-based rasterizer. While faster than NeRFs, the unstructured nature of the point cloud makes density control challenging.

Pruning and Compression: Recent works have attempted to optimize 3DGS models. "Trimming the Fat" [1] focuses on post-training pruning, removing primitives that contribute minimally to the final image. However, this does not aid the optimization process itself. Uncertainty-based pruning methods, such as PUP 3D-GS [3], utilize variance metrics to guide pruning but can struggle with convergence without strong priors. Probabilistic masking approaches like MaskGaussian [8] attempt adaptive density control via masking but often incur texture loss. In this work, we actively modify the loss landscape during training using spatial priors and leverage occupancy grids for real-time geometry validation.

3. Methodology

Our framework modifies the standard 3DGS optimization pipeline in two key areas: the loss function and the densification/pruning logic.

3.1. 2D Regularization via Gradient-Aware Maps

The standard 3DGS loss function is a linear combination of L1 loss and D-SSIM term. We argue that the current loss formulation encourages the model to overfit noise in smooth regions. To counter this, we introduce a spatially varying parameter $\lambda_{(x,y)}$.

Gradient Map Generation: We pre-compute a gradient map based on the ground truth training images. For each image, we convert it to grayscale and apply a Sobel filter to extract gradient magnitudes. This gradient map is normalized to the range $[0, 1]$ and then inverted. The result is a "Lambda Heat Map" where flat, smooth regions have high values (strong regularization) and detailed edges have low values (weak regularization).

Modified Loss Function: The objective function is augmented as:

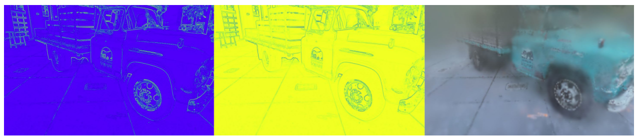


Figure 1. Visualization of the spatial regularization components on the *Trucks* scene at iteration 130. (a) On the left, we see Gradient Map extracted from the ground truth image, highlighting high-frequency details. (b) The corresponding Lambda (λ) Heat Map, where **Blue** represents low values ($\lambda \approx 0$, weak regularization) in detailed regions, and Yellow represents high values ($\lambda \approx 1$, strong regularization) in flat regions. (c) The final Rendered Image on the right, demonstrating how the adaptive regularization preserves detail while smoothing flat areas.

$$\mathcal{L} = (\lambda_{(x,y)})\mathcal{L}_1 + (\mathbf{1} - \lambda_{(x,y)})\mathcal{L}_{D-SSIM} \qquad (1)$$

By weighting the L1 term heavily in smooth regions, we force the optimizer to rely on color consistency rather than structural complexity, effectively "smoothing out" the Gaussian distribution in those areas and facilitating the removal of redundant small Gaussians during the pruning phase.

3.2. Algorithmic Implementation and Adaptive Densification

To enforce our spatial regularization strategy, we modify the core densification logic of the 3DGS optimization loop. The complete procedure is detailed in Algorithm 1.

Spatial Constraint on Densification (Line 14): A critical innovation in our approach is the conditional check introduced at Line 15 of Algorithm 1: **if** $Reg(\mu) < \tau_{reg}$. In the standard 3DGS pipeline, any Gaussian exhibiting high positional gradients ($\nabla_p L > \tau_p$) is a candidate for densification (either cloning or splitting). This mechanism is designed to add geometry where the reconstruction error is high.

However, in flat or textureless regions, high gradients often arise not from missing geometry, but from the optimizer struggling to fit noise or slight lighting variations using multiple small Gaussians. Densifying in these regions leads to over-reconstruction and "floaters." [1][3]

Our added block acts as a spatial gatekeeper. The function $Reg(\mu)$ queries the spatially varying regularization map (the Lambda Heat Map described in Sec. 3.1) at the Gaussian's projected location.

- If the Gaussian resides in a high-detail region (e.g., edges, texture), $Reg(\mu)$ is high, allowing standard densification to proceed to capture fine details.
- If the Gaussian resides in a smooth, flat region, $Reg(\mu)$ is low. The condition fails, and we strictly *block* the splitting or cloning of Gaussians in this area.

Algorithm 1 Optimization and Adaptive Densification

```
Require: $w, h$: width and height of training images
 1: $M \leftarrow$ SfM Points;
 2: $S, C, A \leftarrow$ InitAttributes();
 3: $i \leftarrow 0$;
 4: while not converged do
 5:     $V, \hat{I} \leftarrow$ SampleTrainingView();
 6:     $I \leftarrow$ Rasterize($M, S, C, A, V$);
 7:     $L \leftarrow$ Loss($I, \hat{I}$);
 8:     $M, S, C, A \leftarrow$ Adam($\nabla L$);
 9:     if IsRefinementIteration($i$) then
10:         for all Gaussian $(\mu, \Sigma, c, \alpha)$ do
11:             if $\alpha < \epsilon$ or IsTooLarge($\mu, \Sigma$) then
12:                 RemoveGaussian();
13:             end if
14:             if $\nabla_p L > \tau_p$ then
15:                 if $Reg(\mu) < \tau_{reg}$ then
16:                     if $||S|| > \tau_S$ then
17:                         SplitGaussian($\mu, \Sigma, c, \alpha$);
18:                     else
19:                         CloneGaussian($\mu, \Sigma, c, \alpha$);
20:                     end if
21:                 end if
22:             end if
23:         end for
24:     end if
25:     $i \leftarrow i + 1$;
26: end while
```

By suppressing densification in low-frequency regions, we force the existing Gaussians to optimize their parameters (scale and opacity) to fit the region, rather than spawning new primitives. This effectively creates a scenario where only essential geometry remains, leading to a sparser representation without explicit post-hoc pruning.

3.3. Occupancy Grid-Based Pruning

A major limitation of vanilla 3DGS is its blindness to global geometry; it only sees what is projected onto the camera view. To resolve this, we integrate an explicit volumetric prior using NerfAcc [7].

Occupancy Grid Estimator: We initialize an Occupancy Grid Estimator with a resolution of 128, i.e., ($128 \times 128 \times 128$ voxel grid), covering the scene's bounding box. This grid acts as a coarse proxy for scene geometry based on whether a particular voxel in the grid is occupied or not. Intuitively, a voxel being marked empty implies that the surrounding region of the voxel has low opacity, and vice-versa.

Density Querying and Update: The grid must be updated dynamically as the Gaussians move and change opacity. Every few iterations, we query the density of the scene.

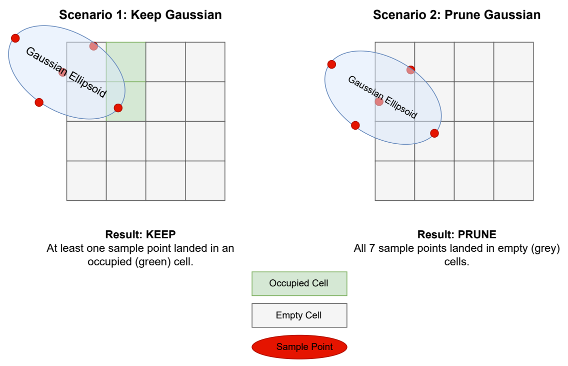

Scenario 1: Keep Gaussian
Scenario 2: Prune Gaussian
Gaussian Ellipsoid
Gaussian Ellipsoid
Result: KEEP
At least one sample point landed in an occupied (green) cell.
Result: PRUNE
All 7 sample points landed in empty (grey) cells.
Occupied Cell
Empty Cell
Sample Point

Figure 2. *The pruning heuristic visualized using a simple example. A Gaussian is marked for pruning if none of the points on its boundary overlap with the occupied voxels from the occupancy grid.*

Since 3DGS does not have a continuous density field, we approximate density by stochastically sampling Gaussians and using a weighted combination of the opacity of the neighboring Gaussians for a given grid voxel. This allows the grid to learn which voxels are occupied (dense) and which are empty.

Pruning Heuristic: We implement a strict pruning logic based on grid occupancy. For a given Gaussian ellipsoid defined by mean μ and covariance Σ:

- We first represent each Gaussian ellipsoid using a finite set of points (depicted by the red points in Figure 2). For our experiments we chose these points to be on the boundary, and the mean point.
- If all sample points fall into voxels marked as ”empty” by the occupancy grid, the Gaussian is classified as a floater or an artifact and is immediately pruned.
- If at least one sample point falls into an ”occupied” voxel, the Gaussian is retained.

This pruning workflow is also illustrated in Figure 2.

Phased Optimization Schedule: To prevent premature pruning of valid geometry that has not yet converged, we divide training into two phases:

1. **Early Phase** ($0 - 15k$ **iterations):** The model undergoes standard adaptive density control (densify and prune) to build up the scene structure. Simultaneously, the occupancy grid is updated and used for pruning, ensuring only informative Gaussians are added.
2. **Late Phase** ($15k - 30k$ **iterations):** We disable the standard densification to freeze the general structure. We then enable NerfAcc-guided pruning every 1000 iterations. This removes only a few Gaussians since the guided pruning during the early phase ensures that only useful Gaussians are retained. This observation is further bolstered by the results in the top left plot in Figure 5.

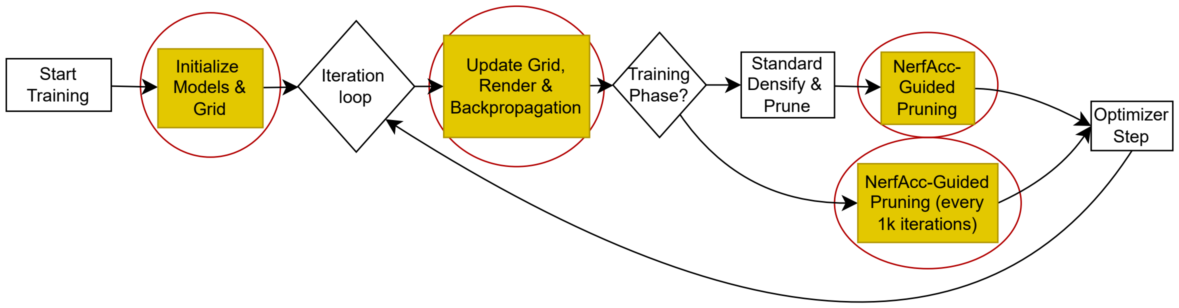


Figure 3. The 3D pruning pipeline using occupancy grids.

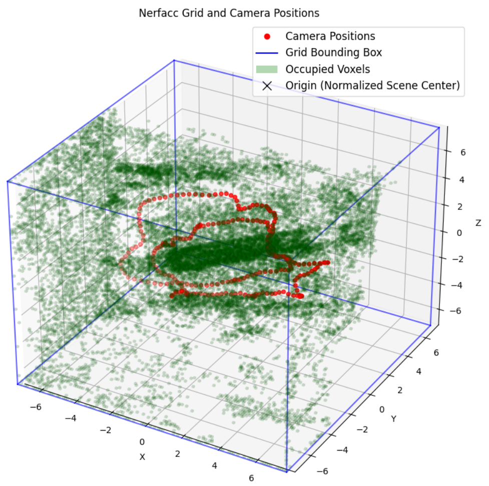


Figure 4. *Occupancy grid for the train image set from the tanks and temples dataset.*

[1].

Figure 3 illustrates the overall training and pruning pipeline. The process begins by initializing the models and the underlying grid, after which the algorithm enters the main iteration loop. In each iteration, the grid is updated, a render pass is performed, and gradients are backpropagated. Depending on the current training phase, different pruning strategies are applied: during the early phase (less than 15k iterations), standard densification and pruning are used, whereas in the late phase the system switches to a nerfacc-guided pruning scheme executed periodically. The red-circled blocks highlight the modules where our proposed modifications are introduced, specifically affecting initialization, per-iteration grid updates, and the nerfacc-guided pruning stages.

4. Experiments and Results

Dataset and Preprocessing. We evaluate our method on the *Tanks and Temples* dataset [5], specifically utilizing the “Train” scene from the 3DGS-30K. This scene captures a diesel locomotive in an outdoor environment, presenting a challenging mix of large, texture-less surfaces (the metal cabin) and high-frequency geometric details (vegetation, and tracks).

The standard input consists of calibrated images and an initial sparse point cloud ($\sim$1 million points) generated via Structure-from-Motion. We employ a train/test split consistent with the original gaussian splatting paper. For the “Train” scene, every 8^{th} image from the sorted sequence is held out for evaluation, resulting in a test set comprising approximately 12.5% of the total views. The remaining images serve as the training set for optimization. All reported quantitative metrics (PSNR, SSIM, LPIPS) are computed on these unseen test views whereas the plots are that of training.

4.1. Implementation Details

We implemented our method using the PyTorch-based 3D Gaussian Splatting framework, integrated with the `nerfacc` library. All experiments were conducted on a H200 GPU cluster from PACE-ICE. The optimization ran for a total of 30,000 iterations.

4.2. Quantitative Evaluation

We compare our proposed method against the original 3DGS implementation and three state-of-the-art pruning baselines: ”Trimming the Fat” [1], ”PUP 3D-GS” [3], and ”MaskGaussian” [8]. We report Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), Learned Perceptual Image Patch Similarity (LPIPS) and Compression Ratio.

Analysis: Table 1 summarizes the performance of several pruning and regularization strategies across PSNR, SSIM, LPIPS, and compression ratio. Although some baselines obtain more aggressive compression, for example,

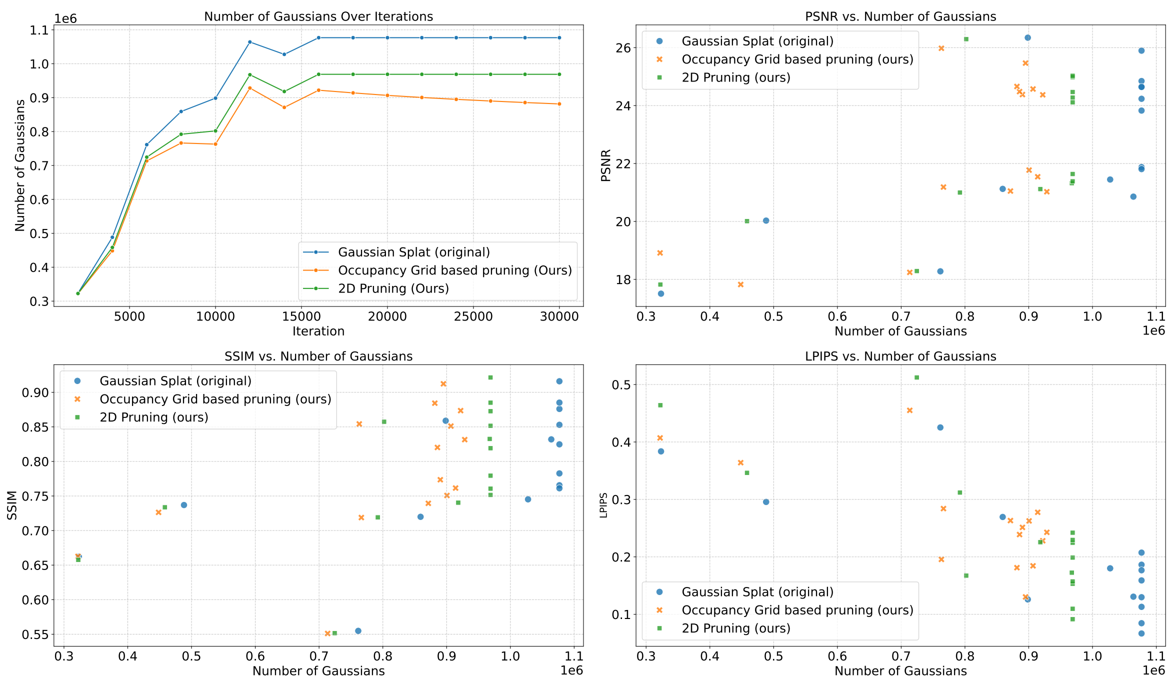


Figure 5. *Quantitative metrics for experiments performed for pruning during the training scene. (Top left) plots the number of Gaussians with iterations. We can observe that during the proliferation stage, the first 15k iterations, the cloning of Gaussians is more controlled for the Occupancy grid (nerfacc) and 2D-based pruning methods. (Top right) PSNR metrics vs. Number of gaussians for the two pruning methods and the original Gaussian splatting algorithm. (Bottom left) SSIM vs. No. of gaussians for the two pruning methods and the original Gaussian splatting algorithm. (Bottom right) LPIPS vs. No. of Gaussians for the two pruning methods and the original Gaussian splatting algorithm.*

Table 1. Quantitative comparison on the Tanks & Temples dataset (Train scene) on the held-out evaluation set

Method	**PSNR**	**SSIM**	**LPIPS (↓)**	**Compression ratio**
3D Gaussian Splatting [4]	22.14	0.802	0.218	1.0
Trimming the fat [1]	22.51	0.761	0.319	0.08
Pup 3D-GS [3]	21.03	0.7600	0.296	0.09
Probabilistic Masking [8]	22.01	0.812	0.214	0.37
2D Regularization (**Ours**)	22.31	0.794	**0.212**	0.9
3D Pruning (**Ours**)	22.12	0.794	**0.217**	0.71

Trimming the Fat[1] and Pup 3D-GS[3] reduce the model size to 0.08 and 0.09 of the original, respectively however this comes at a notable cost in reconstruction quality, as reflected by their lower SSIM and higher LPIPS scores. In contrast, our methods are not designed to be the most aggressively compressed representations; their compression ratios (0.9 for 2D Regularization and 0.71 for 3D Pruning) are more conservative compared to the pruning-focused baselines. However, despite applying much milder compression, our approaches achieve some of the best perceptual fidelity in the table. The 2D Regularization variant delivers the lowest LPIPS (0.212) and a competitive PSNR. At the same time, the 3D Pruning method preserves image quality close to the original Gaussian Splatting (PSNR 22.12 vs. 22.14) but still offers a meaningful reduction in model size.

Overall, while our methods do not target state-of-the-art compression ratios, they provide a favorable balance between fidelity and compactness, making them attractive in scenarios where reconstruction quality is prioritized over extreme compression.

Convergence Behavior: We analyzed the number of Gaussians over iterations. Our occupancy grid-based pruning (Late Phase) results in a stable reduction in primitive count after 15k iterations, whereas the baseline continues to accumulate unnecessary primitives.

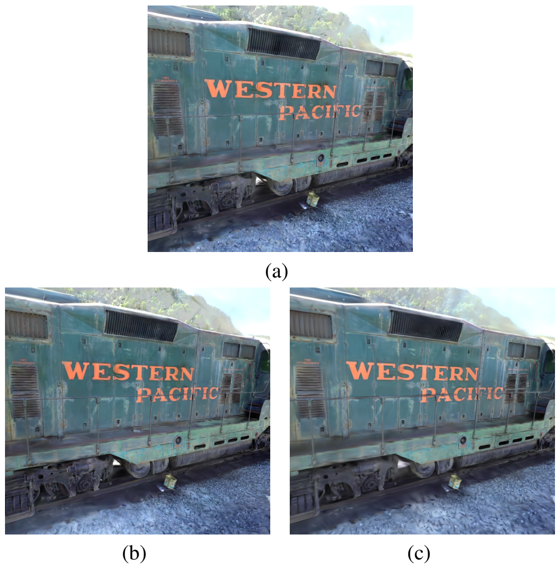


Figure 6. *(a) Reconstruction using original Gaussian splatting. (b) Reconstruction using 2D pruning. (c) Reconstruction using 3D occupancy grid based pruning. Moving from (a) to (c) we see that the fidelity of faraway points of the tree leaves becomes lower which is expected.*

4.3. Qualitative Evaluation

We present visual comparisons in Figure 6.

Artifact Removal: The original 3DGS model often produces "floaters" in the sky or empty background regions. Our 3D Pruning method, guided by the occupancy grid, effectively identifies and removes these artifacts, resulting in a cleaner background.

Surface Smoothing: In flat regions, such as the side of the train cabin, the 2D regularization enforces smoothness. The original model renders these areas with high-frequency noise due to overlapping small Gaussians. Our result shows a more coherent surface texture that accurately reflects the material properties.

5. Conclusion

In this work, we presented a comprehensive framework for Spatial Pruning in 3D Gaussian Splatting. By identifying the limitations of standard densification in empty and flat regions, we introduced a dual-pronged approach: 2D spatial regularization to penalize over-densification in smooth areas and 3D occupancy grid pruning to eliminate geometric artifacts. Our extensive evaluation on the Tanks & Temples dataset demonstrates that this approach not only compresses the scene representation but also effectively enhances visual fidelity, outperforming the current state-of-the-art 3DGS baseline.

Future work will focus on integrating depth priors from monocular estimation networks to further refine the initial sparse point cloud, potentially allowing for even faster convergence in few-shot scenarios.

References

[1] Muhammad Salman Ali, Maryam Qamar, Sung-Ho Bae, and Enzo Tartaglione. Trimming the fat: Efficient compression of 3d gaussian splats through pruning. In *British Machine Vision Conference (BMVC)*, 2024. 1, 2, 4, 5

[2] Sara Fridovich-Keil, Giacomo Meoni, R Luo, W Wang, A Pirinen, K Ilen, and A Yu. K-planes: Explicit radiance fields in space, time, and appearance. In *CVPR*, 2023. 1, 2

[3] Alex Hanson, Allen Tu, Vasu Singla, Mayuka Jayawardhana, Matthias Zwicker, and Tom Goldstein. Pup 3d-gs: Principled uncertainty pruning for 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 2, 4, 5

[4] 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. 1, 2, 5

[5] 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. 4

[6] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. In *CVPR*, 2024. 1

[7] Ruilong Li, Matthew Tancik, and Angjoo Kanazawa. Nerfacc: Efficient sampling accelerates nerfs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 18537–18546, 2023. 1, 3

[8] Yifei Liu, Zhihang Zhong, Yifan Zhan, Sheng Xu, and Xiao Sun. Maskgaussian: Adaptive 3d gaussian representation from probabilistic masks, 2025. 2, 4, 5

[9] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 2

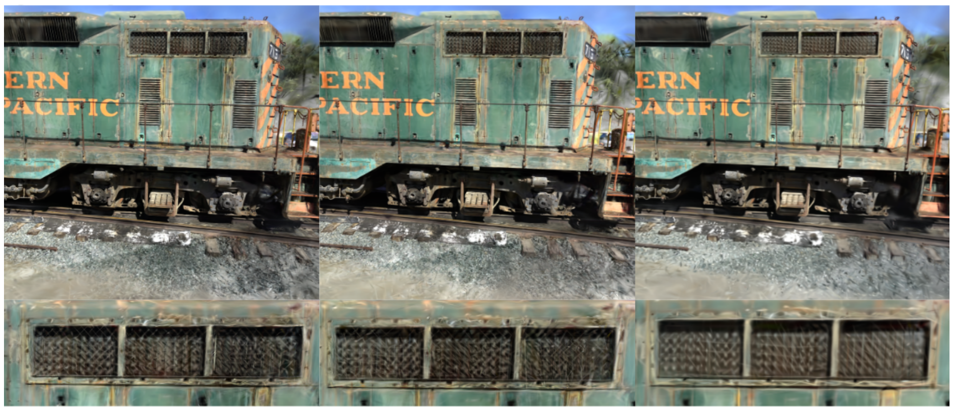
[10] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14335–14345, 2021. 1

Spatial Pruning in 3D Gaussian Splatting

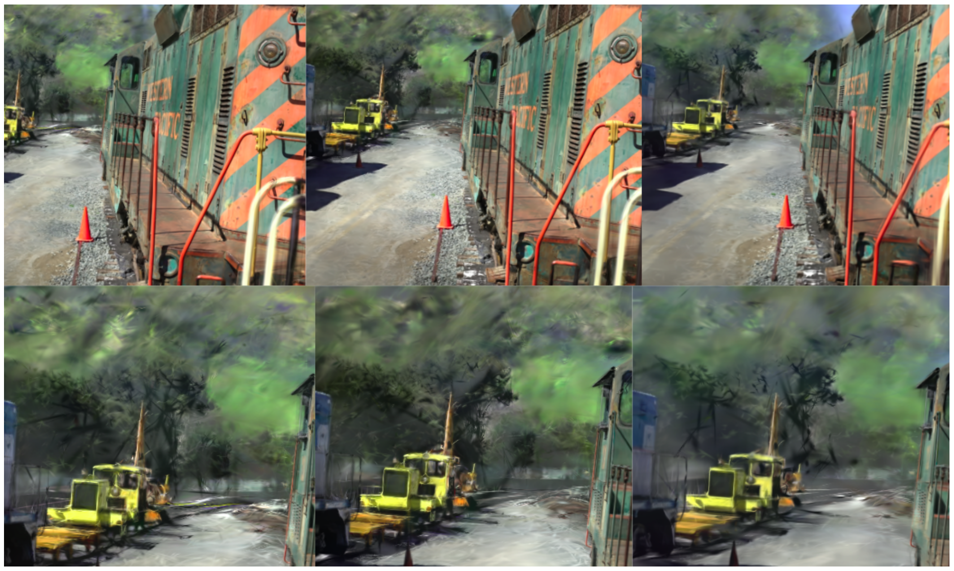
Supplementary Material

6. Additional results

In this section we present additional qualitative results for the spatial pruning in 3D gaussian splatting.

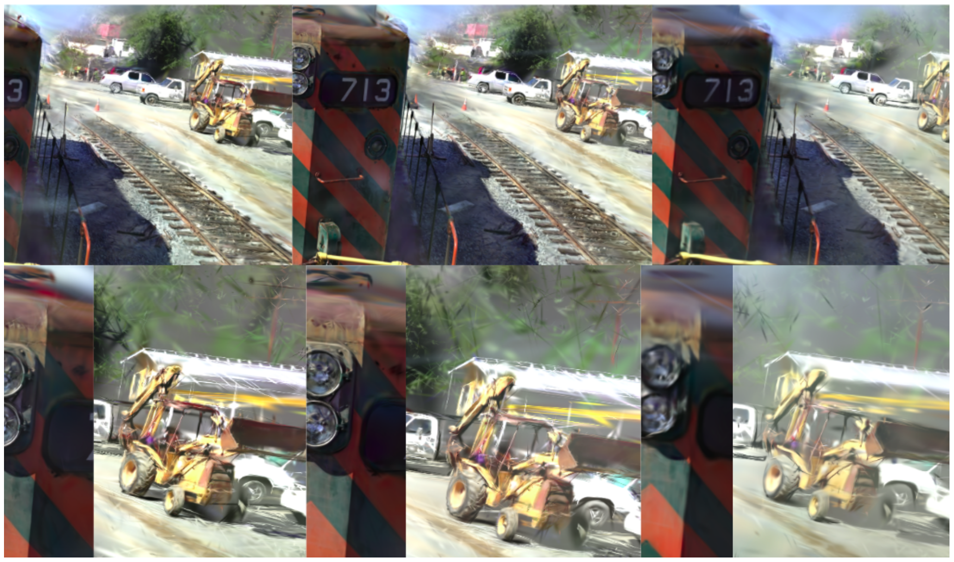


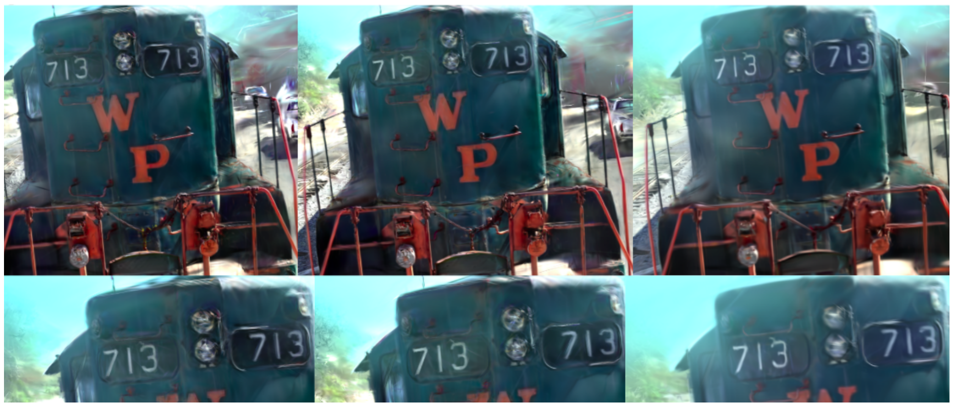
(a)



(b)

Figure 7. *(from left to right), original gaussian splatting, 2d pruning, and 3d occupancy grid based regularization. In each image set the top row is a reconstruction from a particular camera view and the bottom row is a zoomed in view highlighting the differences in reconstruction between the three algorithms. We can conclude that compared to original gaussian splatting, the two pruning algorithms provide similar reconstruction.*


(a)



(b)

Figure 8. *(from left to right), original gaussian splatting, 2d pruning, and 3d occupancy grid based regularization. In each image set the top row is a reconstruction from a particular camera view and the bottom row is a zoomed in view highlighting the differences in reconstruction between the three algorithms. We can conclude that compared to original gaussian splatting, the two pruning algorithms provide similar reconstruction.*