Splat-by-Splat: Progressive Gaussian Splatting for Efficient Scene Reconstruction

Christo Aluckal, Charuvahan Adhivarahan and Karthik Dantu

***Abstract*— 3D Gaussian Splatting (3DGS) has recently emerged as a powerful representation for novel view synthesis, achieving high-quality rendering at real-time speeds. However, conventional training requires full access to all posed images, leading to high memory usage, long training times, and limited applicability in resource constrained environments. We propose an incremental training strategy for 3DGS, where datasets are partitioned into smaller chunks and integrated sequentially. This approach reduces immediate computational burden, enables training on low power devices, and allows progressive refinement of the scene representation as new data arrives. Our framework extends 3DGS to real-time and edge compute environments, opening the door to more adaptive, efficient, and sustainable scene reconstruction systems.**

I. Introduction

Recent advances in 3D Gaussian Splatting (3DGS) have demonstrated remarkable improvements in both rendering quality and speed for novel view synthesis. However, training 3DGS [1] models on large scale datasets remains computationally demanding. Vanilla 3DGS typically requires access to the full set of posed images during optimization, leading to high memory usage and long training times. These requirements are prohibitive for edge devices or resource constrained settings, where memory, compute power, and energy availability are limited. Prior work has sought to alleviate this by pruning redundant gaussians [2], [3], [4], [5]. In contrast, our approach incrementally introduces Gaussians during training, reducing the immediate computational burden and enabling efficient training on devices with limited resources. Incremental updates also allow the model to progressively refine its representation, beginning with a coarse approximation and improving fidelity as more data is introduced. This approach is analogous to many robot exploration pipelines or SLAM [6], [7], [8] systems that incrementally add more information. By reframing 3DGS training as an incremental process, we improve scalability, extend applicability to real-time and edge compute environments, and open the door to continual or lifelong scene reconstruction frameworks.

II. Methodology

A. Preliminary

A gaussian splat consists of a collection of anisotropic 3D gaussians. These are characterized by their position $\mu \in \mathbb{R}^3$, rotations $\mathbf{R} \in \mathbb{R}^{3\times3}$, scale $\mathbf{S} \in \mathbb{R}^3$, opacity $\mathbf{O} \in \mathbb{R}$ and spherical harmonics $\mathbf{SH} \in \mathbb{R}^{48}$. From these base parameters, a covariance $\mathbf{\Sigma}$ is extracted using $\mathbf{\Sigma} = \mathbf{RSS^TR^T}$ [9]. The image is then rendered using

$$I = C(\mathbf{G} \mid \mathbf{R}, \mathbf{t}) = \sum_{i=1}^{N} T_i \alpha_i c_i \tag{1}$$

$$G(x) = e^{-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)} \tag{2}$$

$$\alpha_i = (1 - exp(-\sigma_i \delta_i)), \quad T_i = \prod_{j=1}^{i-1}(1-\alpha_j) \tag{3}$$

where x is the 3D coordinate, C is the rendered image of the gaussians $\mathbf{G}$ under rotation $\mathbf{R}$ and translation $\mathbf{t}$ of the viewport with intervals δ. This combines the effect of all overlapping Gaussians accounting for their colors c, opacities α, densities σ and transmittance T [10].

B. Preprocessing

The input to the system is a complete COLMAP [11] sparse database. This database consists of a file containing the pose and keypoints of the reconstructed images called `images` and a file containing the information of all reconstructed 3D points in the dataset called the `points3D`. Each frame in the `images` is associated with an image ID, camera ID and transformation $\mathbf{T_w^c} = (\mathbf{R_w^c}, \mathbf{t_w^c})$ from the world coordinate system to the camera coordinate system. Along with the frame information, the file also consists of global IDs of each keypoints. The `points3D` file consists of keypoints indexed using their global IDs. These keypoints have an associated position (X, Y, Z) and color (R, G, B). In 1, we partition the COLMAP dataset into N disjoint subsets of images. For each subset, we maintain the set of images belonging to that subset and the set of 3D points observed in those images. To avoid duplication, we maintain a global set of "seen" points $\mathcal{P}_{\text{seen}}$. When processing a new subset, we add all its images to the subset's image set, and we collect only the 3D points that have not appeared in previous subsets. The image sets are then saved as N distinct COLMAP models which can be used in the 3DGS pipeline.

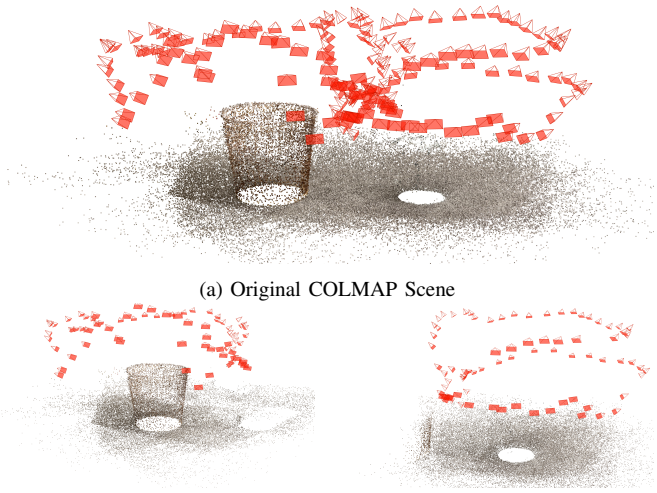
C. 3DGS

In the incremental 3DGS pipeline, training begins with an initial set of Gaussians, camera viewports, and 3D points corresponding to the first subset. These parameters are optimized as in the standard 3DGS procedure. At a user defined iteration, new Gaussians and new viewports are appended

All authors are with the Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY 14260, USA. {christoa, charuvah, kdantu}@buffalo.edu

Algorithm 1 Partitioning COLMAP dataset into N portions with unique 3D points per subset

Partition images into N disjoint subsets: $\{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_N\}$
Initialize global seen set: $\mathcal{P}_{\text{seen}} \leftarrow \varnothing$
for $k = 1 \rightarrow N$ **do**
  Initialize image set for chunk k: $\mathcal{I}'_k \leftarrow \mathcal{I}_k$
  Initialize point set for chunk k: $\mathcal{P}'_k \leftarrow \varnothing$
  for each image $i \in \mathcal{I}'_k$ **do**
    Extract visible points: $\mathcal{P}(i) \leftarrow \text{Points}(i)$
    for each $p \in \mathcal{P}(i)$ **do**
      if $p \notin \mathcal{P}_{\text{seen}}$ **then**
        Add p to $\mathcal{P}'_k$
        Add p to $\mathcal{P}_{\text{seen}}$
      end if
    end for
  end for
end for
return $\{(\mathcal{I}'_1, \mathcal{P}'_1), \ldots, (\mathcal{I}'_N, \mathcal{P}'_N)\}$ =0



(a) Original COLMAP Scene

(b) Scene 1

(c) Scene 2

Fig. 1: Splitting the COLMAP dataset

to the dataset from the next subset. This allows the model to retain previously optimized structure while progressively incorporating new information, enabling scalable training without restarting from scratch.

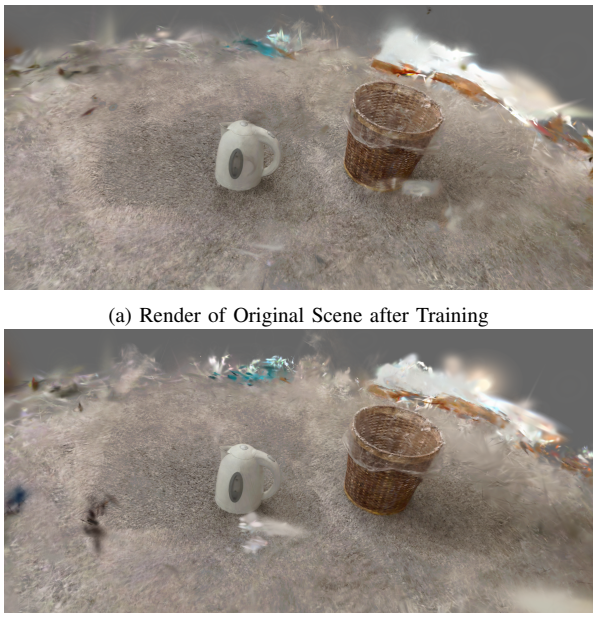
III. Results

Figure 1 shows the splitting of a custom COLMAP scene. The Original scene consists of 167 frames of the resolution 1895x1064. 10 frames/viewports are used for testing while the remaining 157 are used for training. The dataset is split into two scenes, Scene 1 Figure 1b and Scene 2 Figure 1c consisting of 90 images and 77 images respectively. From each individual scene, 5 frames are taken as test frames with the remaining being the training frames. Both scenes, when combined form the Augmented Scene.

Resolution	Scene	Time (s)	L1 Loss	PSNR	Gaussian count
$\frac{1}{8}$	Original	522	0.027	29.6	501k
	Augmented	**474**	**0.018**	**32.2**	**499k**
$\frac{1}{4}$	Original	827	0.027	28.9	843k
	Augmented	**699**	**0.022**	**30.6**	**743k**
$\frac{1}{2}$	Original	1625	0.028	28.3	954k
	Augmented	**1291**	**0.023**	**30.1**	**781k**

TABLE I: Experiment results for different resolutions of two scenes

Table I shows the results from running both the original and augmented scenes through the 3DGS pipeline. Each experiment applies a resolution reduction factor, where a $1/8$ resolution indicates resizing images to one-eighth of their original size. Training time corresponds to the total runtime for 30k iterations. Evaluation metrics include the final L1 loss and PSNR on the test set, while the Gaussian count reflects the number of primitives generated during training. From the table, we can see that the Augmented scene outperforms the Original scene. In vanilla 3DGS, densification often produces redundant Gaussians through replication, which are later pruned. Instead, our augmentation strategy introduces new Gaussians only after the previously added set has been sufficiently optimized.



(a) Render of Original Scene after Training

(b) Render of Augmented Scene after Training

Fig. 2

Figure 2 shows the rendered image from both the scenes from the $1/8$ resolution training set. The image generated from the Augmented scene shows lesser artifacts.

References

[1] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, July 2023.

[2] Z. Fan, K. Wang, K. Wen, Z. Zhu, D. Xu, Z. Wang, *et al.*, "Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps," *Advances in neural information processing systems*, vol. 37, pp. 140138–140158, 2024.

[3] G. Fang and B. Wang, "Mini-splatting: Representing scenes with a constrained number of gaussians," in *European Conference on Computer Vision*, pp. 165–181, Springer, 2024.

[4] S. Girish, K. Gupta, and A. Shrivastava, "Eagles: Efficient accelerated 3d gaussians with lightweight encodings," in *European Conference on Computer Vision*, pp. 54–71, Springer, 2024.

[5] A. Hanson, A. Tu, V. Singla, M. Jayawardhana, M. Zwicker, and T. Goldstein, "Pup 3d-gs: Principled uncertainty pruning for 3d gaussian splatting," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 5949–5958, 2025.

[6] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pp. 225–234, IEEE, 2007.

[7] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[8] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *2011 international conference on computer vision*, pp. 2320–2327, IEEE, 2011.

[9] X. Lang, L. Li, C. Wu, C. Zhao, L. Liu, Y. Liu, J. Lv, and X. Zuo, "Gaussian-lic: Real-time photo-realistic slam with gaussian splatting and lidar-inertial-camera fusion," *arXiv preprint arXiv:2404.06926*, 2024.

[10] J. Lee, M. Kong, M. Park, and E. Kim, "Geomgs: Lidar-guided geometry-aware gaussian splatting for robot localization," *arXiv preprint arXiv:2501.13417*, 2025.

[11] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.