Efficient Multi-Scale 3D Gaussian Splatting

Umar Farooq, Jean-Yves Guillemaut, Adrian Hilton, Marco Volino
{m.farooq,j.guillemaut,a.hilton,m.volino}@surrey.ac.uk

Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey (UK)

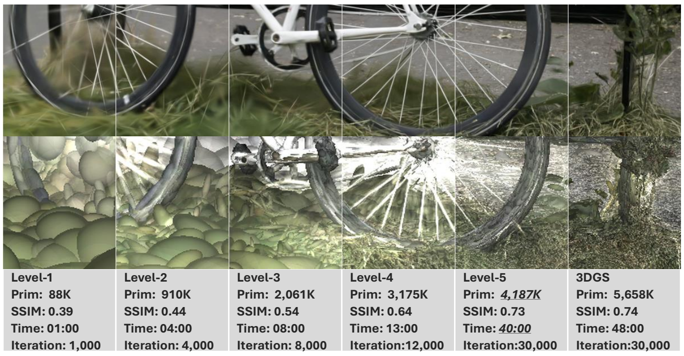

Figure 1: **Top:** RGB rendering. **Middle:** Gaussian Visualization. **Bottom:** Size, SSIM, cumulative training time and global iteration. Each column indicates one level of detail used in our coarse-to-fine optimization and the last column shows 3DGS for comparison.

Introduction: 3D Gaussian Splatting (3DGS) [1] is an explicit point-based novel view synthesis technique that achieves high visual quality and fast training times. However, 3DGS suffers from high memory and storage usage [3, 4] limiting its applicability across various device form factors. In this context, we introduce a novel and efficient 3DGS coarse-to-fine optimization strategy. Our method reduces the memory overhead of 3DGS by initiating the training process with significant over-reconstruction, which serves as an effective regularizer, and progressively refines the scene representation. Our approach also produces stand-alone scene representations at each level-of-detail in the progressive refining process, enabling variable storage, transmission, and rendering based on the downstream requirements.

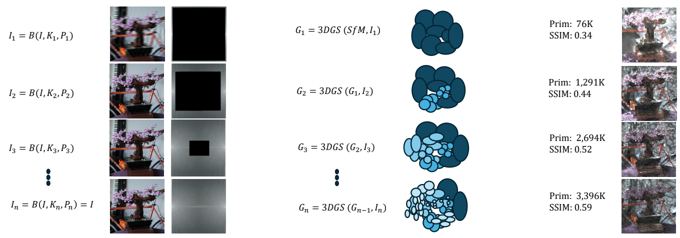

Figure 2: Our frequency modulation and progressive Gaussian levels-of-details method diagram.

Method: We employs a progressive frequency control strategy with five distinct image and scene quality levels I_n and G_n respectively in range $n = (1-5)$ as depicted in our method diagram shown in Figure 2. We start by applying the blur with a large kernel size to the training images, reducing their detail and noise. This initial reduction in detail allows the model to focus on learning the broader, more significant structures of the scene. As the training progresses, the level of blur is gradually reduced by reducing the size of the blur filter which reintroduces higher frequency details in a controlled manner.

The optimization starts from sparse initial 3D points obtained from structure from motion (SfM)[5] on the training images I and this acts as our input at G_1 as shown in Figure 2. Then for each subsequent level in our coarse-to-fine optimization process we use the last set of Gaussians G_{n-1} as the starting point and the 3DGS[1] optimization loss is used to optimize G_{n-1} using the filtered images I in range $n = (1-5)$. Where the images I_n are obtained from our frequency modulation function. For each level the images get progressively sharper and more high frequency content is allowed to remain in the image. For the last level at $n = 5$ used we directly pass the orignal training images to the 3DGS optimizer.

Results: Our method reduces the number of primitives required by 62%, lowers GPU memory usage by 40% and reduces optimization time by 20% as shown in Figure 3. Our method successfully reconstructs subtle details like grass and shrubs all the while using fewer Gaussian primitives. The proposed method is combined with an off-the-shelf compression method [3] to obtain further compression and to showcase the general nature of our contribution.

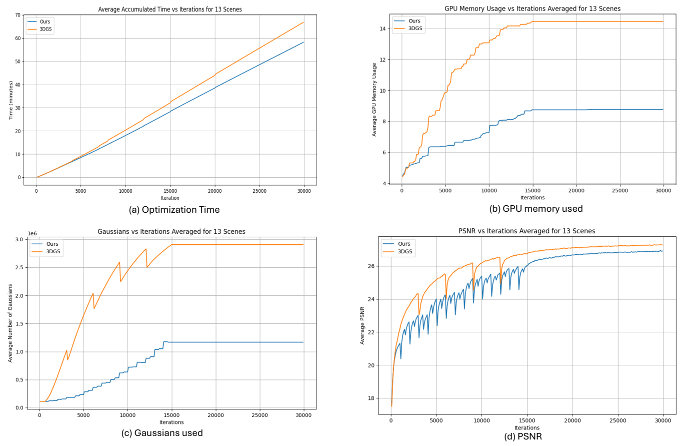

Figure 3: Shows optimization time, GPU memory usage, number of Gaussian primitives and PSNR for our method compared to 3DGS[1].

Table 1: Quantitative results for our method on commonly used benchmark datasets.

Method	SSIM ↑	PSNR ↑	LPIPS ↓	Size (MB) ↓
Mip-NeRF360				
Plenoxels	0.626	23.080	0.463	2,100.0
INGP-Base	0.671	25.300	0.371	13.0
INGP-Big	0.699	25.590	0.331	48.0
Mip-NeRF 360	0.792	27.690	0.237	8.6
3DGS	0.815	27.210	0.214	734.0
Reduced-3DGS[4]	0.809	27.100	0.226	29.0
Compact-3DGS[2]	0.797	27.030	0.247	29.1
Compress-3DGS[3]	0.801	26.981	0.238	28.8
Ours-Full	0.797	26.777	0.256	15.87
Tanks & Temples				
Plenoxels	0.719	21.080	0.379	2,300.0
INGP-Base	0.723	21.720	0.330	13.0
INGP-Big	0.745	21.920	0.305	48.0
Mip-NeRF 360	0.759	22.220	0.257	8.6
3DGS	0.841	23.140	0.183	411.0
Reduced-3DGS[4]	0.840	23.570	0.188	14.0
Compact-3DGS[2]	0.831	23.320	0.202	20.9
Compress-3DGS[3]	0.832	23.324	0.194	17.3
Ours-Full	0.819	23.061	0.224	9.93
Deep Blending				
Plenoxels	0.795	23.060	0.510	2,700.0
INGP-Base	0.797	23.620	0.423	13.0
INGP-Big	0.817	24.960	0.390	48.0
Mip-NeRF 360	0.901	29.400	0.245	8.6
3DGS	0.903	29.410	0.243	676.0
Reduced-3DGS[4]	0.902	29.630	0.249	18.0
Compact-3DGS[2]	0.900	29.730	0.258	23.8
Compress-3DGS[3]	0.898	29.381	0.253	25.3
Ours-Full	0.898	29.350	0.268	12.02

Applications: Our approach enables efficient and usable optimization of 3DGS up to the highest resolution level-of-detail before running out of GPU memory. A scene can be rendered at a different level-of-detail depending on device hardware and user requirements. We also enable downstream applications where parts of a scene can be rendered in variable quality which can further decrease memory footprint and increase render speed.

[1] 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.

[2] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. *arXiv preprint arXiv:2311.13681*, 2023.

[3] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis. *arXiv preprint arXiv:2401.02436*, 2023.

[4] Panagiotis Papantonakis, Georgios Kopanas, Bernhard Kerbl, Alexandre Lanvin, and George Drettakis. Reducing the memory footprint of 3d gaussian splatting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 7(1):1–17, 2024.

[5] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.