# MesonGS: Post-training Compression of 3D Gaussians via Efficient Attribute Transformation
## Supplementary Material

Shuzhao Xie[1], Weixiang Zhang[1], Chen Tang[1,3], Yunpeng Bai[4], Rongwei Lu[1], Shijia Ge[1], and Zhi Wang[1,2,†]

[1] SIGS & TBSI, Tsinghua University, [2] Peng Cheng Laboratory
[3] MMLab, The Chinese University of Hong Kong, [4] The University of Texas at Austin
https://shuzhaoxie.github.io/mesongs/

## 1 Comparisons with covariance-based replacement

In this section, we introduce the experiment settings of our comparison with the covariance-based replacement strategy. We first introduce the replacement strategy proposed in C3DGS [9]. Then, we present how to adapt this idea into our framework. Finally, we analyze the disadvantages of the covariance-based replacement.

### 1.1 How does C3DGS utilize covariance?

C3DGS [9] compresses the 3D coordinates, opacity, the Gaussian shape, and the color feature separately. The Gaussian shape includes the scale vectors and rotation quaternions. The color feature refers to the spherical harmonic (SH) coefficients. Note that they compute a sensitivity score for color and Gaussian shape features before the attribute compression. During vector quantization, they only cluster the features with a sensitivity score below a threshold. To compress the Gaussian shape, they first convert the scale vectors and rotation quaternions into the upper triangle part of the covariance matrix – a vector $\in \mathbb{R}^6$. Then, they use vector quantization to compress the covariance vectors ($N \times 6$) into a codebook ($K \times 6$) with a corresponding index table ($N \times K$). Here, $N$ refers to the number of 3D Gaussians, and $K$ refers to the size of the codebook. Susceptible covariance vectors are added to the codebook after clustering.

As directly optimizing the covariance matrix is not possible [4], they have to decompose the covariance into scales and rotation quaternions during finetuning. Hence, they reparametrize the scale vector $\mathbf{s} = \eta_s \hat{\mathbf{s}}$ to make sure the $\hat{\mathbf{s}}$ is normalized. Meanwhile, the covariance vector for clustering is also rescaled by $\eta_s$:

$$\boldsymbol{\Sigma} = (\mathbf{R}\hat{\mathbf{S}})(\hat{\mathbf{S}}\mathbf{R}) = \frac{1}{\eta_s^2}\boldsymbol{\Sigma}. \tag{1}$$

† Corresponding author.

As the $\hat{\mathbf{s}}$ is normalized, they can easily decompose a covariance vector into a normalized scale vector and a rotation quaternion. However, to restore the $\mathbf{s}$, they have to store one more number – the scaling factor $\eta_s$ for each Gaussian in the final compressed file, requiring 7 numbers.

In contrast, we propose to replace the rotation quaternion $\mathbf{q} \in \mathbb{R}^4$ with the Euler angles $\mathbf{e} \in \mathbb{R}^3$, which requires 6 numbers for the Gaussian shape. Our replacement strategy keeps the positive definiteness of the covariance matrix and thus can support direct optimization [4].

### 1.2 Fair comparison with covariance-based replacement

To achieve a fair comparison, we implement the covariance-based replacement for the post-training scenarios. Specifically, we do not introduce the scaling factor $\eta_s$ as we do not require finetuning in the post-training scenarios. Hence, the storage of covariance-based replacement and Euler angles-based replacement are equal now. However, we find that applying RAHT to the covariance vector or quantizing the covariance vectors into 8 bits can cause severe visual quality degradation. Hence, we set the quantization bits as 16 and do not apply RAHT on both Euler-angle-based and covariance-based replacements. Notably, we use a unique non-linear quantization scheme for covariance-based replacement. For a covariance vector $\mathbf{c}$, we quantize it with:

$$\mathbf{c}_q = \lfloor \text{clamp}(\frac{\mathbf{c}}{S_{\mathbf{c}}} + Z_{\mathbf{c}}, 0, 2^b - 1) \rceil, \tag{2}$$

where

$$S_c = \frac{\mathbf{c}_l - \mathbf{c}_r}{2^b}, Z_c = \lfloor 2^b - \frac{\mathbf{c}_r}{S_c} \rceil. \tag{3}$$

Here,

$$\mathbf{c}_l = \mathbf{c}_m - \lambda_{\mathbf{c}}\sigma, \mathbf{c}_r = \mathbf{c}_m + \lambda_{\mathbf{c}}\sigma, \tag{4}$$

where $\mathbf{c}_m$ and $\sigma$ is the mean value and the standard deviation of $\mathbf{c}$, respectively. We use the $\lambda_{\mathbf{c}}$ to control the value of $\mathbf{c}_l$ and $\mathbf{c}_r$. Notably, we store values less than $\mathbf{c}_l$ or greater than $\mathbf{c}_r$ in float format directly. In the context of preserving covariance in two distinct data types, namely `int` and `float`, it becomes necessary to incorporate an additional indicator to denote whether the covariance vector associated with each attribute tuple necessitates quantization. Here, an attribute tuple is defined as the set `(opacity, scale, covariance, 0-D SH coefficients)`. To circumvent the need for such an indicator, we opt to position attribute tuples requiring `covariance` quantization at the start of the complete attribute tensor, while those exempt from such quantization are positioned towards the end of the whole attribute tuples.

### 1.3 Disadvantages of covariance-based replacement

We tune the $\lambda_{\mathbf{c}}$ and fix other hyperparameters to obtain the compressed files that with different sizes and quality. We display the rendering results of the

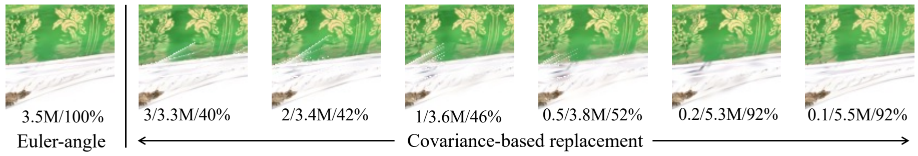


**Fig. 1: Euler angles-based vs. Covariance-based.** "A/B/C" refers to the "$\lambda_{\mathbf{c}}$ / the size of the compressed file / the percent of positive-definite covariance matrices". Replacing scales and rotations with covariance leads to white line artifacts, which greatly affects the visual effect. We adjust the final file size by compressing a portion of the covariance using the $\lambda_{\mathbf{c}}$.

decompressed 3D Gaussians files in Fig. 1. We list the $\lambda_{\mathbf{c}}$, the final file size, and the percent of positive-definite covariance matrices for each case. It can be observed that there are white line artifacts in the covariance-based replacement strategy. The reason for the appearance of these artifacts is that most of the covariance matrices become non-positive definite after decompression. As shown in Fig. 1, when we reduce $\lambda_{\mathbf{c}}$, the artifacts gradually weaken as the percent of non-positive definite covariance matrices decreases. When the proportion of positive definite covariance matrices reaches 92%, the artifacts essentially disappear, and the visual quality is on par with the Euler angles-based replacement strategy. However, in the case of 92%, the file size is much larger than the Euler angles-based replacement strategy.

### 1.4 Impacts of NPD covariance matrices

*Q: 8% of covariance matrices are not positive definite (NPD) in the final compressed file. Doesn't this cause any issues?* No, the code still runs normally. However, in an experiment on the *mic* scene, we observed decreased PSNR (32.7 $\rightarrow$ 32.0) after removing Gaussians with NPD covariance matrices, which suggests that some of the NPD Gaussians still affect the rendering process. This phenomenon is attributed to the robust implementation: a low-pass filter is applied to the cov2d to ensure it covers at least one pixel.

## 2 Hyperparameters and environments of evaluation

Please find the hyperparameter settings in our open-sourced code[†]. We collect the encoding time in a machine with an NVIDIA RTX 3090, an Intel Xeon E5 24C48T CPU, and Ubuntu 20.04. We use the evaluation results proposed in the paper of baselines [2, 6, 7, 9] for comparison.

† https://github.com/ShuzhaoXie/MesonGS

**Table 1: Ablation study of the depth of octree.** The unit of Size is MB. Evaluations on the Synthetic-NeRF dataset. We record the metrics of the final zip file. “Points” refers to the number of points before the voxelization step. “Voxels” refers to the number of voxels after the voxelization step.

| depth | PSNR (dB) | SSIM | LPIPS | Size (MB) | Points | Voxels |
|---|---|---|---|---|---|---|
| 10 | 28.69 | 0.9412 | 0.0532 | 1.03 | 98.5K | 96.2K |
| 11 | 29.30 | 0.9462 | 0.0516 | 1.09 | 98.5K | 97.8K |
| 12 | 29.47 | 0.9476 | 0.0511 | 1.14 | 98.5K | 98.3K |
| 13 | 29.51 | 0.9479 | 0.0510 | 1.18 | 98.5K | 98.5K |
| 14 | 29.51 | 0.9480 | 0.0510 | 1.22 | 98.5K | 98.5K |

## 3 More Ablation Study

**Octree depth.** The octree expression is a lossy compression. Hence, we employed an experiment to show the influence of the depth of octree at Tab. 1 on the Synthetic-NeRF dataset. When the depth is set to 10, there are a significant number of points allocated to the same voxel, resulting in a substantial loss of information. However, when the depth reaches 13, only fewer than one thousand points are merged, resulting in minimal impact on performance.

## 4 More quantitative and qualitative results

In this section, we list more quantitative results in Tab. 2, Tab. 3, Tab. 4, Tab. 5, and Tab. 6. As for qualitative results, please download the full rendering results of the test set at the project page[†].

[†] https://shuzhaoxie.github.io/mesongs/

**Table 2: Mip-NeRF 360 [1] results.** "FT" refers to finetune.

| 3D-GS | | | | |
|---|---|---|---|---|
| Scene | PSNR | SSIM | LPIPS | Size(MB) |
| *bicycle* | 24.90 | 0.733 | 0.265 | 1246.54 |
| *bonsai* | 32.34 | 0.945 | 0.182 | 315.74 |
| *counter* | 29.02 | 0.913 | 0.186 | 266.11 |
| *garden* | 26.89 | 0.849 | 0.134 | 1017.18 |
| *kitchen* | 31.41 | 0.930 | 0.121 | 338.25 |
| *room* | 31.91 | 0.925 | 0.204 | 385.89 |
| *stump* | 26.41 | 0.757 | 0.260 | 922.37 |
| Average | 28.98 | 0.865 | 0.193 | 641.73 |
| **MesonGS** | | | | |
| Scene | PSNR | SSIM | LPIPS | Size(MB) |
| *bicycle* | 24.18 | 0.699 | 0.297 | 46.56 |
| *bonsai* | 30.02 | 0.921 | 0.213 | 12.65 |
| *counter* | 27.61 | 0.887 | 0.214 | 13.82 |
| *garden* | 25.90 | 0.813 | 0.181 | 46.46 |
| *kitchen* | 30.06 | 0.916 | 0.137 | 18.60 |
| *room* | 30.58 | 0.905 | 0.229 | 14.71 |
| *stump* | 25.58 | 0.727 | 0.294 | 39.96 |
| Average | 27.70 | 0.838 | 0.224 | 27.54 |
| **MesonGS-FT** | | | | |
| Scene | PSNR | SSIM | LPIPS | Size(MB) |
| *bicycle* | 24.70 | 0.720 | 0.281 | 46.70 |
| *bonsai* | 31.65 | 0.940 | 0.192 | 12.69 |
| *counter* | 28.70 | 0.908 | 0.196 | 13.86 |
| *garden* | 26.59 | 0.837 | 0.155 | 46.55 |
| *kitchen* | 31.12 | 0.929 | 0.124 | 18.86 |
| *room* | 31.62 | 0.920 | 0.213 | 14.71 |
| *stump* | 25.91 | 0.740 | 0.283 | 39.97 |
| Average | 28.61 | 0.856 | 0.206 | 27.62 |

**Table 3: Synthetic-NeRF [8] results when comparing with 3DGS compression works.** "FT" refers to finetune.

| 3D-GS | | | | |
|---|---|---|---|---|
| Scene | PSNR | SSIM | LPIPS | Size(MB) |
| *chair* | 35.72 | 0.987 | 0.012 | 66.56 |
| *drums* | 26.18 | 0.954 | 0.038 | 82.30 |
| *ficus* | 35.01 | 0.987 | 0.012 | 70.80 |
| *hotdog* | 37.75 | 0.985 | 0.020 | 35.67 |
| *lego* | 35.89 | 0.983 | 0.016 | 76.73 |
| *materials* | 30.01 | 0.961 | 0.035 | 64.83 |
| *mic* | 35.44 | 0.992 | 0.006 | 73.38 |
| *ship* | 30.97 | 0.907 | 0.106 | 78.17 |
| Average | 33.37 | 0.970 | 0.030 | 68.55 |
| MesonGS | | | | |
| Scene | PSNR | SSIM | LPIPS | Size(MB) |
| *chair* | 34.19 | 0.981 | 0.017 | 3.55 |
| *drums* | 25.76 | 0.947 | 0.046 | 3.77 |
| *ficus* | 33.85 | 0.984 | 0.015 | 3.06 |
| *hotdog* | 36.19 | 0.981 | 0.027 | 2.72 |
| *lego* | 34.47 | 0.976 | 0.021 | 4.30 |
| *mat* | 29.04 | 0.950 | 0.048 | 3.97 |
| *mic* | 34.28 | 0.988 | 0.012 | 2.98 |
| *ship* | 30.23 | 0.896 | 0.117 | 4.87 |
| Average | 32.25 | 0.963 | 0.038 | 3.65 |
| MesonGS-FT | | | | |
| Scene | PSNR | SSIM | LPIPS | Size(MB) |
| *chair* | 35.05 | 0.985 | 0.014 | 3.56 |
| *drums* | 25.91 | 0.952 | 0.041 | 3.79 |
| *ficus* | 34.44 | 0.986 | 0.013 | 3.06 |
| *hotdog* | 37.29 | 0.984 | 0.023 | 2.72 |
| *lego* | 35.15 | 0.981 | 0.018 | 4.31 |
| *mat* | 29.67 | 0.958 | 0.038 | 3.98 |
| *mic* | 35.06 | 0.991 | 0.007 | 2.98 |
| *ship* | 30.82 | 0.904 | 0.111 | 4.87 |
| Average | 32.92 | 0.968 | 0.033 | 3.66 |

**Table 4: Synthetic-NeRF [8] results when comparing with NeRF compression works.** "FT" refers to finetune.

| 3D-GS | | | | |
|---|---|---|---|---|
| Scene | PSNR | SSIM | LPIPS | Size(MB) |
| *chair* | 35.72 | 0.987 | 0.012 | 66.56 |
| *drums* | 26.18 | 0.954 | 0.038 | 82.30 |
| *ficus* | 35.01 | 0.987 | 0.012 | 70.80 |
| *hotdog* | 37.75 | 0.985 | 0.020 | 35.67 |
| *lego* | 35.89 | 0.983 | 0.016 | 76.73 |
| *materials* | 30.01 | 0.961 | 0.035 | 64.83 |
| *mic* | 35.44 | 0.992 | 0.006 | 73.38 |
| *ship* | 30.97 | 0.907 | 0.106 | 78.17 |
| Average | 33.37 | 0.970 | 0.030 | 68.55 |
| MesonGS | | | | |
| Scene | PSNR | SSIM | LPIPS | Size(MB) |
| *chair* | 31.25 | 0.971 | 0.025 | 1.04 |
| *drums* | 24.93 | 0.937 | 0.054 | 1.22 |
| *ficus* | 32.45 | 0.979 | 0.019 | 1.05 |
| *hotdog* | 31.46 | 0.965 | 0.045 | 0.57 |
| *lego* | 28.95 | 0.944 | 0.048 | 1.13 |
| *materials* | 27.13 | 0.933 | 0.062 | 0.97 |
| *mic* | 32.02 | 0.982 | 0.019 | 1.02 |
| *ship* | 26.75 | 0.868 | 0.139 | 1.21 |
| Average | 29.37 | 0.947 | 0.051 | 1.03 |
| MesonGS-FT | | | | |
| Scene | PSNR | SSIM | LPIPS | Size(MB) |
| *chair* | 34.03 | 0.980 | 0.019 | 1.04 |
| *drums* | 25.42 | 0.946 | 0.049 | 1.23 |
| *ficus* | 33.04 | 0.981 | 0.018 | 1.04 |
| *hotdog* | 35.73 | 0.980 | 0.031 | 0.58 |
| *lego* | 32.47 | 0.968 | 0.035 | 1.15 |
| *materials* | 28.87 | 0.952 | 0.047 | 0.98 |
| *mic* | 34.22 | 0.989 | 0.010 | 1.03 |
| *ship* | 30.20 | 0.899 | 0.123 | 1.20 |
| Average | 31.75 | 0.962 | 0.042 | 1.03 |

**Table 5: Tank&Temples [5] results.** "FT" refers to finetune. We directly use the pre-trained checkpoints provided by 3D-GS [4]. Hence, please check the 3D-GS paper [4] to obtain the detailed evaluation metrics of 3D-GS.

| MesonGS | | | | |
|---|---|---|---|---|
| Scene | PSNR | SSIM | LPIPS | Size(MB) |
| *train* | 21.34 | 0.792 | 0.235 | 13.03 |
| *truck* | 24.35 | 0.851 | 0.182 | 20.93 |
| Average | 22.85 | 0.822 | 0.208 | 16.98 |
| MesonGS-FT | | | | |
| Scene | PSNR | SSIM | LPIPS | Size(MB) |
| *train* | 21.95 | 0.807 | 0.218 | 13.06 |
| *truck* | 24.70 | 0.868 | 0.167 | 20.93 |
| Average | 23.32 | 0.837 | 0.193 | 16.99 |

**Table 6: Deep Blending [3] results.** "FT" refers to finetune. We directly use the pre-trained checkpoints provided by 3D-GS [4]. Hence, please check the 3D-GS paper [4] to obtain the detailed evaluation metrics of 3D-GS.

| MesonGS | | | | |
|---|---|---|---|---|
| Scene | PSNR | SSIM | LPIPS | Size(MB) |
| *drjohnson* | 28.61 | 0.894 | 0.260 | 27.87 |
| *playroom* | 29.57 | 0.896 | 0.259 | 21.65 |
| Average | 29.09 | 0.895 | 0.260 | 24.76 |
| MesonGS-FT | | | | |
| Scene | PSNR | SSIM | LPIPS | Size(MB) |
| *drjohnson* | 29.00 | 0.900 | 0.252 | 27.87 |
| *playroom* | 30.03 | 0.902 | 0.250 | 21.66 |
| Average | 29.52 | 0.901 | 0.251 | 24.76 |

## References

1. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5470–5479 (2022)
2. Fang, G., Hu, Q., Wang, L., Guo, Y.: ACRF: Compressing explicit neural radiance fields via attribute compression. In: International Conference on Learning Representations(ICLR) (2024)
3. Hedman, P., *et al.*: Deep blending for free-viewpoint image-based rendering. ToG (2018)
4. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics (ToG) **42**(4), 1–14 (2023)
5. Knapitsch, A., *et al.*: Tanks and temples: benchmarking large-scale scene reconstruction. ToG (2017)
6. Lee, J.C., Rho, D., Sun, X., Ko, J.H., Park, E.: Compact 3d gaussian representation for radiance field. arXiv preprint arXiv:2311.13681 (2023)
7. Li, L., Shen, Z., Wang, Z., Shen, L., Bo, L.: Compressing volumetric radiance fields to 1 mb. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4222–4231 (2023)
8. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
9. Niedermayr, S., Stumpfegger, J., Westermann, R.: Compressed 3d gaussian splatting for accelerated novel view synthesis. In: CVPR (2024)