Tile-wise vs. Image-wise: Random-Tile Loss and Training Paradigm for Gaussian Splatting

Supplementary Material

6. Details of NDC Coordinates Gradient

Recall that $\sum \|\mathbf{g}_i\|$ represents the accumulated magnitude of the gradient of the NDC coordinates for the Gaussian within the density interval, as described in Sec. 3.3. More specifically, During "Adaptive Density Control" every 100 iterations, Gaussian i participates in the calculation for M^i viewpoints. The NDC gradient for these viewpoints is accumulated as follows:

$$\frac{\sum_{k=1}^{M^i}\sqrt{\left(\frac{\partial L_k}{\partial \mu_{\text{ndc,x}}^{i,k}}\right)^2+\left(\frac{\partial L_k}{\partial \mu_{\text{ndc,y}}^{i,k}}\right)^2}}{M^i}, \tag{14}$$

where L_k represents the loss under viewpoint k, $\frac{\partial L_k}{\partial \mu_{\text{ndc,x}}^{i,k}}$ and $\frac{\partial L_k}{\partial \mu_{\text{ndc,y}}^{i,k}}$ represent the gradients of Gaussian i in the (x, y) directions of NDC space at viewpoint k. For a given pixel $p = (x, y)$,where c represents its color, we can apply the chain rule to find the derivatives:

$$\frac{\partial L_k}{\partial \mu_{\text{ndc}}^{i,k}} = \frac{\partial L_k}{\partial c_p} \times \frac{\partial c_p}{\partial \alpha_{k,p}^i} \times \frac{\partial \alpha_{k,p}^i}{\partial \mu_{\text{ndc}}^{i,k}}. \tag{15}$$

where α_p^i is a function of the distance between the center of the projected Gaussian and the pixel center, exhibiting exponential decay as the distance increases:

$$\alpha_{k,p}^i = \sigma^i \times \exp\left(-\tfrac{1}{2}\left(p-\mu^{i,k}\right)^T (\Sigma_{2D}^{i,k})^{-1}\left(p-\mu^{i,k}\right)\right), \tag{16}$$

When rendering a complete image, the 2D projection of a Gaussian participates in the alpha blending of multiple pixels within the range of 3σ, where these gradients are accumulated pixel-wise:

$$\frac{\partial L_k}{\partial \mu_{\text{ndc}}^{i,k}} = \sum_{p=1}^{m_k^i}\left(\frac{\partial L_k}{\partial c_p} \times \frac{\partial c_p}{\partial \alpha_{k,p}^i} \times \frac{\partial \alpha_{k,p}^i}{\partial \mu_{\text{ndc}}^{i,k}}\right), \tag{17}$$

where m_k^i represents the number of pixels involved in the calculation for Gaussian i under viewpoint k. Under tile-based rendering, we can accumulate the NDC Coordinates gradients with weights across tiles:

$$\frac{\partial L_{RT}}{\partial \mu_{\text{ndc}}^{i}} = \sum_{tile=1}^{\mathcal{T}}\left(\sum_{p=1}^{n_k^i}\left(\frac{\partial L_k}{\partial c_p} \times \frac{\partial c_p}{\partial \alpha_{k,p}^i} \times \frac{\partial \alpha_{k,p}^i}{\partial \mu_{\text{ndc}}^{i,k}}\right) \times W_{tile}\right), \tag{18}$$

where n_k^i denotes the number of pixels projected onto the tile and involved in the calculation for Gaussian i under viewpoint k, while W_{tile} determines the weight of the NDC gradient for the current projection. In the original 3D-GS, the accumulation of the NDC gradient is not normalized based on the number of pixels covered by the Gaussian projection. This results in distant observations contributing fewer pixels, while closer observations contribute more. To address this imbalance in tile-based training, we propose two feasible splitting strategies in Sec. 3.3. In Iteration-Count Densification, the NDC gradient is weighted by the ratio of the projected area to the pixel area involved in the calculation as in Eq. (13). In contrast, Tile-Count Densification directly averages the NDC gradients across all observations as in Eq. (12).

7. Details of RT-SSIM

Structural Similarity Index Measure(SSIM) is formulated as a combination of three key components: luminance, contrast, and structure comparison metrics. It is expressed mathematically as:

$$\text{SSIM}(a, b) = l(\boldsymbol{a}, \boldsymbol{b})c(\boldsymbol{a}, \boldsymbol{b})s(\boldsymbol{a}, \boldsymbol{b}). \tag{19}$$

Here, $l(\boldsymbol{a}, \boldsymbol{b})$ represents the luminance comparison, $c(\boldsymbol{a}, \boldsymbol{b})$ evaluates the contrast,and $s(\boldsymbol{a}, \boldsymbol{b})$ captures the structural similarity.

$$l(\boldsymbol{a}, \boldsymbol{b}) = \frac{2\mu_a\mu_b + C_1}{\mu_a^2 + \mu_b^2 + C_1}, \tag{20}$$

$$c(\boldsymbol{a}, \boldsymbol{b}) = \frac{2\sigma_a\sigma_b + C_2}{\sigma_a^2 + \sigma_b^2 + C_2}, \tag{21}$$

$$s(\boldsymbol{a}, \boldsymbol{b}) = \frac{\sigma_{ab} + C_3}{\sigma_a\sigma_b + C_3}, \tag{22}$$

where C_1, C_2, and C_3 are small constants, $C_1 = (K_1L)^2$, $C_2 = (K_2L)^2$, and $C_3 = C_2/2$. .And μ_a and μ_b are the means of the patches a and b, σ_a^2 and σ_b^2 are the variances, σ_{ab} is the covariance between the patches.

We use commonly set $K_1 = 0.01$ and $K_2 = 0.03$ throughout this paper. The data range L is defined as 1 for pixel RGB values, and the SSIM index lies within the range $[-1, 1]$. The local statistics and the SSIM index are computed within the sliding window. To evaluate the quality, the final SSIM metric used is the mean SSIM (MSSIM), obtained by averaging the SSIM indices across all steps.

$$\text{SSIM}(\hat{\mathcal{I}}, \mathcal{I}) = \frac{1}{N_w}\sum_{w=1}^{N_w}\text{SSIM}(\hat{W}_w, W_w) \tag{23}$$

where $\hat{W}_w$ and W_w denote the w-th sliding window in the rendered image and the corresponding window in the ground truth image, respectively.N_w represents the number of sliding windows.

In Random-Tile training, to retain the structural information inherent in the viewpoint, we compute the SSIM across all random tiles selected within the viewpoint v after shuffling the tiles:

$$\text{RT-SSIM}(\hat{\mathcal{T}}, \mathcal{T}) = \frac{1}{N_{tilew}^{(v)}} \sum_{w=1}^{N_{tilew}^{(v)}} \text{SSIM}(\hat{t_w^{(v)}}, t_w^{(v)})) \quad (24)$$

where $N_w^{(v)}$ represents the total number of sliding windows on tiles selected under the random viewpoint v, and the windows slide only within the selected tiles. V represents the number of sampled viewpoints, which determines the diversity of viewpoints in the training constraints. However, a larger V is not always better. To maintain the same training batch size, increasing V reduces the number of tiles per viewpoint, thereby weakening the constraints. And it increases the memory overhead for projections across different viewpoints. In our experiments, we set $V = 5$ by default.

8. Densifycation Algorithm

Our optimization and densification algorithms are summarized in Algorithm 2.

Algorithm 2: Optimization and Densification

while *not converged* **do**
  $V, \mathcal{T}, M \leftarrow$ SampleRandomTiles() ;
  $\hat{\mathcal{T}} \leftarrow$ Rasterize(P, S, M, C, A, V) ; ▷ Alg. 3
  $L \leftarrow RthLoss(\mathcal{T}, \hat{\mathcal{T}})$; ▷ Loss Eq. ??
  $P, S, C, A \leftarrow$ Adam(∇L) ; ▷ Backprop
  if *IsDensificationIteration(iter)* **then**
    forall *Gaussians* (μ, Σ, c, α) ***in*** (P, S, C, A) **do**
      if $\alpha < \epsilon$ *or IsTooLarge*(μ, Σ) **then**
        RemoveGaussian() ; ▷ Pruning
      end
      if $\overline{\nabla_\mu L_{RT}} > \tau_{density}$ **then**
        if $\|S\| > \tau_S$ **then**
          SplitGaussian(μ, Σ, c, α)
        end
        else
          CloneGaussian(μ, Σ, c, α)
        end
      end
    end
  end
  $iter \leftarrow iter + 1$
end

$\overline{\nabla_\mu L_{RT}}$, i.e., $\frac{\partial L_{RT}}{\partial \mu_{\text{ndc}}^i}$ in Eq. (18), is derived from the NDC coordinate gradient based on Random-Tile inputs and serves as the core of densify strategy, as outlined in Sec. 3.3.

9. Details of The Random-Tile Rasterizer

We adjusted the rasterizer to accommodate the multi-view input introduced by Random-Tile while simultaneously fully leveraging GPU parallelism.

Algorithm 3: GPU Software for RT-Rasterization
K: 64-bit Keys encoding index and depth of splats
P,S: Gaussian means and covariances
M: Mask list for selecting random tiles
C,A: Gaussian colors and opacities
V: View configuration list

Function Random-Tile Rasterize(P, S, M, C, A, V) **:**
  forall *view* v *in* V **do**
    CullGaussian(p, v) ; ▷ Frustum Culling
  end
  $P', S' \leftarrow$ ScreenspaceGaussians(P, S, V) ;
  $\mathcal{T} \leftarrow$ CreateTiles(V, M) ;
  $L, K \leftarrow$ DuplicateWithKeys(P', T) ;
  SortByKeys(K, L) ; ▷ Globally Sort
  $R \leftarrow$ IdentifyTileRanges($\mathcal{T}$, K)
  $\hat{\mathcal{T}} \leftarrow \mathbf{0}$; ▷ Init Canvas
  forall *Tile* t *in* $\mathcal{T}$ **do**
    forall *Pixel* i *in* t **do**
      $r \leftarrow$ GetRandomTileRange(R, t) ;
      $t[i] \leftarrow$ BlendInOrder(i, L, r, K, P', S', C, A);
    end
  end
  return $\hat{\mathcal{T}}$

Sorting. In the preprocessing stage, a splat list is created for each tile using projection filtering and sorting. To reduce the number of Gaussians processed while maintaining the high parallelism of GPU radix sorting, a select mask is applied to retain only the splats affecting the selected random tiles. Each splat instance is assigned a key of up to 64 bits, with the lower 32 bits encoding its projected depth and the upper bits encoding the overlapping tile's index, following the 3D-GS approach. As a result, a single radix sort can efficiently resolve the depth ordering of splats across all tiles in parallel, regardless of their originating viewpoints.We summarize the rasterization approach in Algorithm 3.

Blending. In the blending process, we use the original settings, while during both the forward and backward passes, we skip any blending updates with $\alpha < \epsilon$ (we choose ϵ as $\frac{1}{255}$) and clamp α to a maximum of 0.99. The accumulated opacity is capped at 0.99 to prevent a large number of low-opacity Gaussians from participating in the forward

rasterization, thereby reducing unnecessary computational overhead.

10. Per-Scene Metrics

Tabs. 9 to 16 list the error metrics used in our evaluation across all considered methods and scenes. The primary goal of our experimental design is to ensure a fair comparison between our Rth-Loss and origin image-wise loss along with their respective training paradigms for Gaussian splatting. Unless stated otherwise, our experimental setup follows the original configurations to establish baseline results.

Table 9. SSIM scores for Mip-NeRF360 [3] scenes.

Method | Scenes	bicycle	garden	stump	room	counter	kitchen	bonsai
Mip-NeRF360 [3]	0.685	0.813	0.744	0.913	0.894	0.920	0.941
iNPG [34]	0.491	0.649	0.574	0.855	0.798	0.818	0.890
Plenoxels [13]	0.496	0.6063	0.523	0.842	0.759	0.648	0.814
3D-GS [19]	0.771	0.868	0.775	0.914	0.905	0.932	0.938
Scaffold-GS [31]	0.705	0.842	**0.784**	0.925	0.914	0.928	0.946
Ours	**0.790**	**0.886**	0.783	**0.937**	**0.927**	**0.948**	**0.963**

Table 10. PSNR scores for Mip-NeRF360 [3] scenes.

Method | Scenes	bicycle	garden	stump	room	counter	kitchen	bonsai
Mip-NeRF360 [3]	24.37	26.98	26.40	31.63	29.55	32.23	**33.46**
iNPG [34]	22.19	24.60	23.63	29.27	26.44	28.55	30.34
Plenoxels [13]	21.91	23.49	20.66	27.59	23.62	23.42	24.67
3D-GS [19]	25.25	27.41	26.55	30.63	28.70	31.63	31.98
Scaffold-GS [31]	24.50	27.17	26.27	31.93	29.34	31.30	32.70
Ours	**25.40**	**27.49**	**26.76**	**32.83**	**29.72**	**32.35**	33.10

Table 11. LPIPS scores for Mip-NeRF360 [3] scenes.

Method | Scenes	bicycle	garden	stump	room	counter	kitchen	bonsai
Mip-NeRF360 [3]	0.301	0.170	0.261	0.211	0.204	0.127	**0.176**
iNPG [34]	0.487	0.312	0.450	0.301	0.342	0.254	0.227
Plenoxels [13]	0.506	0.386	0.503	0.419	0.441	0.447	0.398
3D-GS [19]	**0.205**	**0.103**	**0.210**	0.220	0.204	0.115	0.205
Scaffold-GS [31]	0.306	0.146	0.284	0.202	0.191	0.126	0.185
Ours	0.206	0.110	**0.210**	**0.188**	**0.177**	**0.112**	**0.176**

Table 12. Storage size (MB) for Mip-NeRF360 [3] scenes.

Method | Scenes	bicycle	garden	stump	room	counter	kitchen	bonsai
3D-GS [19]	1291	**1268**	1034	**317**	261	372	281
Ours	**1113**	1279	**916**	318	**194**	**320**	**236**

11. More Ablation Studies and Discussions

Extended Training Iterations. We extend the training iterations to assess the impact of training sufficiency. The results in Tab. 17 show that the improvement from Rth-Loss cannot be achieved by simply extending the training time for image-wise loss. Naively extending the training steps not only increases the time without achieving the effect of Rth's multi-view constraints but also raises the risk of overfitting. As shown by the 3D-GS (90k) results of the COURTHOUSE in Tab. 17, additional training not only fails to improve but even reduces the PSNR on the test set.

Freezing the Densification. As shown in Sec. 4, Rth-Loss improves the geometric structure of Gaussian points in many scenes. To further validate its impact on optimizing Gaussian attributes, we freeze the densification of Gaussians for comparison. That is, the quantities of the initialized Gaussian primitives are kept fixed from the beginning to the end of training. Tab. 18 demonstrates that Rth-Loss still achieves faster and better convergence without densification. This demonstrates that the multi-view constraints

Table 13. Metrics Scores for Tanks&Temples [21] scenes.

SSIM↑

Method | Scenes	Barn	Caterpillar	Courthouse	Ignatius	Meetingroom	Truck	Train
3D-GS [19]	0.859	0.791	0.810	0.814	0.879	0.870	0.813
Ours	**0.876**	**0.815**	**0.832**	**0.818**	**0.904**	**0.883**	**0.839**

PSNR↑

Method | Scenes	Barn	Caterpillar	Courthouse	Ignatius	Meetingroom	Truck	Train
3D-GS [19]	28.44	23.74	23.13	22.53	26.27	25.22	22.06
Ours	**29.17**	**24.62**	**24.14**	**22.78**	**26.98**	**25.62**	**22.56**

LPIPS↓

Method | Scenes	Barn	Caterpillar	Courthouse	Ignatius	Meetingroom	Truck	Train
3D-GS [19]	0.217	0.244	0.255	0.165	0.205	0.189	0.200
Ours	**0.201**	**0.221**	**0.232**	**0.148**	**0.188**	**0.178**	**0.182**

Table 14. Storage size (MB) for Tanks&Temples [21] scenes.

Method | Scenes	Barn	Caterpillar	Courthouse	Ignatius	Meetingroom	Truck	Train
3D-GS [19]	231	281	**174**	371	**276**	633	**255**
Ours	**206**	**241**	192	**322**	279	**618**	262

Table 15. Metrics Scores for SMERF [11] and Deep Blending [16] scenes.

SSIM↑

Method | Scenes	Alameda	Berlin	London	Nyc	Dr Johnson	Playroom
3D-GS [19]	0.733	0.891	0.808	0.844	0.899	0.906
Ours	**0.764**	**0.906**	**0.827**	**0.863**	**0.905**	**0.916**

PSNR↑

Method | Scenes	Alameda	Berlin	London	Nyc	Dr Johnson	Playroom
3D-GS [19]	22.31	27.54	25.74	26.89	28.77	30.04
Ours	**23.92**	**28.80**	**26.63**	**28.03**	**29.78**	**30.52**

LPIPS↓

Method | Scenes	Alameda	Berlin	London	Nyc	Dr Johnson	Playroom
3D-GS [19]	0.407	0.290	0.370	0.330	0.244	0.241
Ours	**0.366**	**0.264**	**0.341**	**0.298**	**0.227**	**0.221**

Table 16. Storage size (MB) for SMERF [11] and Deep Blending [16] scenes.

Method | Scenes	Alameda	Berlin	London	Nyc	Dr Johnson	Playroom
3D-GS [19]	**229**	233	**163**	286	**715**	515
Ours	263	**224**	247	**259**	787	**456**

Table 17. **Extended Training Iterations.** The default number of iterations is 30k. We extended this to 90k for 3D-GS to evaluate its performance, with the last row presenting our results at 30k as a reference.

Scene(views) / Methods(iterations)	KICHEN(279) SSIM↑	KICHEN(279) PSNR↑	KICHEN(279) LPIPS↓	COURTHOUSE(1106) SSIM↑	COURTHOUSE(1106) PSNR↑	COURTHOUSE(1106) LPIPS↓
3D-GS (30k)	0.932	31.64	0.115	0.810	23.13	0.255
3D-GS (60k)	0.934	31.90	0.113	0.818	23.69	0.242
3D-GS (90k)	0.934	31.94	0.113	0.819	23.67	0.241
Ours (30k)	**0.948**	**32.35**	**0.112**	**0.832**	**24.14**	**0.232**

Table 18. **Freezing the Densification.** Performance evaluation with densification disabled, tested at 7k and 30k iterations.

Scene(views) Methods(iterations)	KICHEN(279) SSIM↑	PSNR↑	LPIPS↓	Mem(MB)	COURTHOUSE(1106) SSIM↑	PSNR↑	LPIPS↓	Mem(MB)
3D-GS(7k)	0.894	28.32	0.184	58	0.731	20.61	0.380	118
3D-GS(30k)	0.913	29.84	0.157	58	0.806	22.96	0.261	118
Ours(7k)	0.904	29.28	0.169	58	0.759	21.85	0.340	118
Ours(30k)	**0.917**	**30.32**	**0.151**	58	**0.816**	**23.83**	**0.246**	118

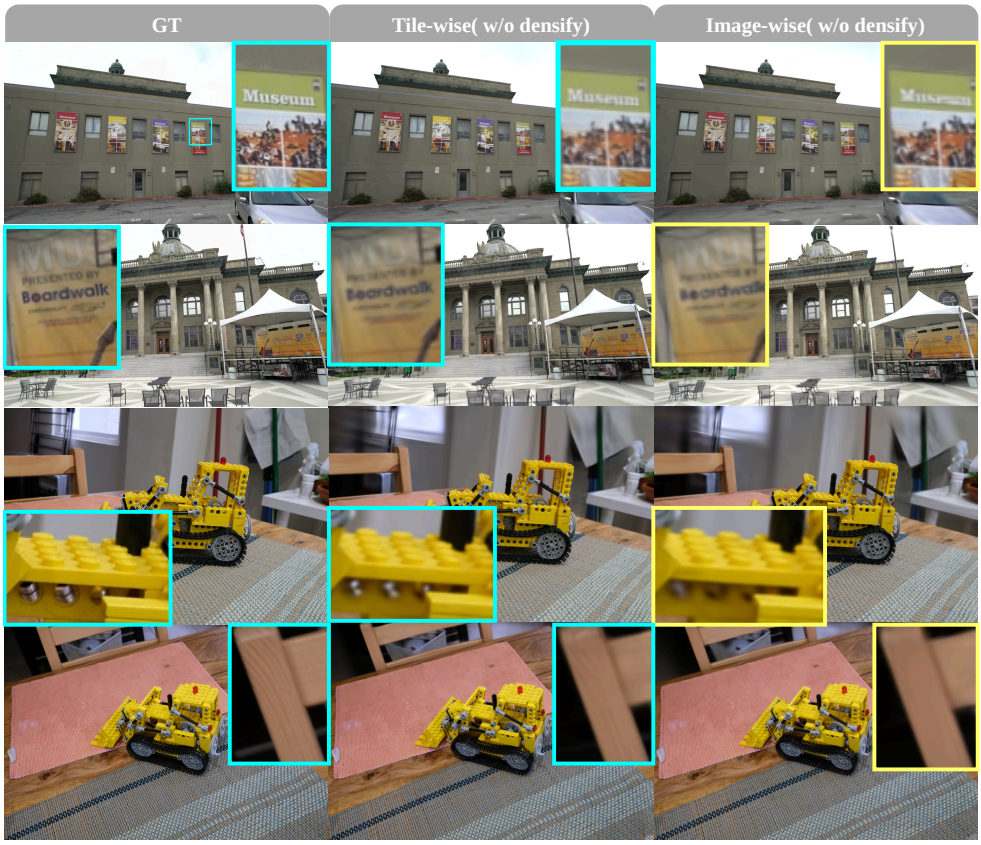

Figure 8. **Freezing the Densification.** Optimizing the same set of Gaussians, Rth-Loss achieves better quality.Sec. 11

introduced by tile-wise Rth-Loss offer superior optimization for Gaussian attributes compared to image-wise loss. Visual comparisons are provided in Fig. 8.