# FastGS: Training 3D Gaussian Splatting in 100 Seconds

Supplementary Material

## 7. Overview

The supplementary material provides the following contents: Sec. 8 gives a detailed description of the proposed compact box. Sec. 9 presents additional experimental details, where Sec. 9.1 provides implementation details, and Secs. 9.2 and 9.3 describe the execution details of integrating FastGS into different tasks and backbones. Sec. 10 reports the computational overhead, Sec. 11 includes additional ablations, and Sec. 12 provides scene-wise results.

## 8. Details of Compact Box

Formally, during the $\alpha$-blending process, the alpha value of the $i$-th Gaussian at pixel $p$ is defined as:

$$\alpha_i(p) = \sigma_i \cdot \exp\left( -\tfrac{1}{2}(p-\boldsymbol{\mu}_{i_{2D}})\boldsymbol{\Sigma}_{i_{2D}}^{-1}(p-\boldsymbol{\mu}_{i_{2D}})^\top \right). \quad (1)$$

This equation implies that $\alpha_i(p)$ decays exponentially with the Mahalanobis distance:

$$A(p) = (p-\boldsymbol{\mu}_{i_{2D}})\boldsymbol{\Sigma}_{i_{2D}}^{-1}(p-\boldsymbol{\mu}_{i_{2D}})^\top. \quad (2)$$

Intuitively, pixels closer to the Gaussian center contribute more, while those farther away have negligible influence. Based on this observation, a reasonable criterion can be established: Gaussian–tile pairs corresponding to pixels with large Mahalanobis distances can be safely discarded, as their effect on the final rendering is minimal.

To prune Gaussian–tile pairs whose contributions are negligible, we define a threshold for the Mahalanobis distance as:

$$(p-\mu_{i_{2D}})\,\Sigma_{i_{2D}}^{-1}\,(p-\mu_{i_{2D}})^T = \beta\left(2\ln\frac{\sigma_i}{\tau_\alpha}\right), \quad (3)$$

where $\beta$ is a scaling factor. By adjusting $\beta$, the effective support region of each 2D Gaussian can be flexibly controlled. A smaller $\beta$ yields a tighter ellipse around the mean $\mu_{i_{2D}}$, thereby reducing the spatial extent of $(p-\mu_{i_{2D}})$ and limiting the number of pixels influenced by the Gaussian $\mathcal{G}_j$. This selective suppression of marginal Gaussian contributions effectively reduces redundant Gaussian–tile pairs and accelerates rasterization.

In implementation, Eq. (3) is integrated into Speedy-Splat [6]'s snugbox, where the parameter $\beta$ further reduces the 2D Gaussian footprint and shrinks its intersection region with tiles, thus forming our CB. We then obtain the Gaus- sian–tile pairs using Speedy-Splat [6]'s accutile method. We sincerely appreciate the excellent work of Speedy-Splat [6].

## 9. More Details

### 9.1. Implementation Details

Our FastGS integrate VCD, VCP, and CB into 3DGS-accel [10, 14] and adopt the widely used absolute gradients [20] to ensure more accurate densification, ultimately achieving an average training time of around 100 seconds per scene. The core of our method lies in strictly controlling the number of Gaussians throughout training via VCD and VCP, maintaining it at a very low level and thereby enabling significant acceleration, as shown in Fig. 2.

In practice, our VCD and VCP introduce a multi-view consistency constraint that strictly regulates Gaussian densification and pruning, preventing the creation of redundant Gaussians. For densification, newly added Gaussians are required not only to satisfy the gradient-based criteria but also to have an importance score greater than $\tau_d$, which is set to 5. We follow the vanilla 3DGS [10] gradient for cloning, while adopting the absolute gradient from AbsGS [20] for splitting. For pruning, before 15k iterations, we follow 3DGS [10] but retain VCP by sampling half of the candidate Gaussians according to their pruning scores. After 15k iterations, pruning is performed every 3k iterations by removing Gaussians with opacity below 0.1 or pruning score exceeding 0.9, ensuring multi-view consistency.

Our baseline, 3DGS-accel [10, 14], preserves the vanilla 3DGS [10] pipeline while integrating the per-splat parallel backpropagation and accelerated SH optimization from Taming-3DGS [14], along with an optimizer schedule that updates the optimizer every 32 iterations from 15,000 to 20,000 iterations and every 64 iterations thereafter. This schedule is inspired by the SH optimization strategy in Taming-3DGS [14]. We do not consider it a conceptual contribution of our method, so instead of discussing it in the main paper, we incorporate it directly into the baseline configuration. As shown in Tab. 16, this scheduling strategy provides acceleration comparable to sparse Adam [14] while fully preserving rendering quality.

We sincerely thank Taming-3DGS [14] for providing a strong baseline, upon which our work builds. By further integrating the proposed VCD, VCP, and CB components, together with the absolute gradients from AbsGS [20], our method achieves significant acceleration. This improvement largely stems from the strict control of Gaussian count enforced by VCD and VCP, ensuring that the number of Gaussians remains as low as possible throughout training. Other modules contribute only marginally to this aspect, which we further analyze in the ablation study.

Table 8. **Quantitative results of sparse-view reconstruction.** We present the results under the 3-view and 6-view settings.

| Method | LLFF [15] 3-view | | | | | | LLFF [15] 6-view | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ | FPS↑ | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ | FPS↑ |
| DropGaussian [16] | 0.73 | 20.43 | 0.707 | **0.202** | 0.08M | 183 | 0.88 | 24.67 | **0.836** | **0.116** | 0.18M | 174 |
| +Ours | **0.30** | **20.58** | **0.708** | 0.217 | **0.03M** | **206** | **0.37** | **24.68** | 0.834 | 0.131 | **0.09M** | **199** |

## 9.2. Generalizing FastGS to Other Tasks

**Dynamic Scene Reconstruction:** Deformable-3DGS [19] adopts the same ADC strategy as vanilla 3DGS [10], while additionally predicting per-Gaussian deformation parameters. This design remains fully compatible with our framework. Building on our accelerated 3DGS backbone, we employ VCD and VCP to precisely regulate densification and pruning, ensuring that the number of Gaussians remains low throughout training. CB is further integrated to speed up rendering, and the optimizer is Adam [11].

**Surface Reconstruction:** PGSR [2] uses an ADC strategy similar to vanilla 3DGS [10]. Based on our accelerated 3DGS backbone, we replace ADC with VCD and VCP to strictly control Gaussian growth and elimination. CB is integrated into the rasterization stage to further improve efficiency. Adam [11] is used as the optimizer.

**Sparse-view Reconstruction:** DropGaussian [16] differs from vanilla 3DGS [10] in that it randomly sets the opacity of a subset of Gaussians to zero during rendering, eliminating their contribution. Based on our accelerated 3DGS backbone, we apply VCD and VCP to precisely control densification and pruning, keeping the number of Gaussians low throughout training. CB is incorporated to accelerate rendering, and Adam [11] is used as the optimizer. Additional experimental results are presented in Tab. 8.

**Large-scale Reconstruction:** Octree-GS [17] uses an anchor-based parameterization where each anchor generates multiple Gaussians. Based on our accelerated 3DGS backbone, we apply VCD to constrain anchor expansion, requiring the importance score of associated Gaussians to exceed 5. VCP is not applied since pruning is performed at the anchor level. CB is integrated into rasterization. The optimizer is Adam [11].

**SLAM:** Photo-SLAM [9] follows vanilla 3DGS [10]'s ADC strategy. Based on our accelerated 3DGS backbone, we integrate VCD and VCP for effective densification and pruning, and incorporate CB to speed up rendering. Adam [11] is used as the optimizer. Results are presented in Tab. 9.

## 9.3. Equipping FastGS to Backbones

**Mip-Splatting [21]:** Mip-Splatting [21] introduces a filtering mechanism for anti-aliasing while following an ADC strategy similar to vanilla 3DGS [10]. Based on our accelerated 3DGS backbone, we replace its ADC pipeline with VCD and VCP, which precisely control Gaussian densification and pruning to keep the number of Gaussians low throughout training. CB is also integrated into the rasterization stage to further accelerate rendering. Adam [11] is used as the optimizer. Additional experimental results are presented in Tab. 10.

Table 9. **Quantitative results of SLAM.** Our method achieves an average 2.70× training speed-up.

| Method | Replica [18] RGB-D | | | | | |
|---|---|---|---|---|---|---|
| | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ | FPS↑ |
| Photo-SLAM [9] | 5.03 | **37.01** | **0.961** | **0.026** | 0.33M | 744 |
| +Ours | **1.86** | **37.01** | 0.957 | 0.042 | **0.11M** | **2700** |

**Scaffold-GS [13]:** Scaffold-GS [13] adopts an anchor-based Gaussian representation. On our accelerated 3DGS backbone, we apply VCD to control densification and maintain a low number of Gaussians. VCP is not applicable because pruning is performed at the anchor level. CB is integrated into the rasterization stage to further accelerate rendering. Adam [11] is used as the optimizer. Additional experimental results are presented in Tab. 10.

**3DCS [8]:** 3DCS [8] is a study exploring alternatives to Gaussian representations, which represents scenes using explicit 3D Smooth Convexes. We combine FastGS with 3DCS, with results shown in Tab. 11. Our method accelerates 3DCS training by over 2×, as summarized below, demonstrating its broad applicability.

## 10. Computational Overhead

We report computational resource consumption in Tab. 12. As shown, our method requires relatively low GPU memory, making it suitable for devices with limited resources.

## 11. Additional Ablation

In this section, we perform more comprehensive ablations based on 3DGS-accel [10, 14] to further demonstrate that the proposed multi-view consistency-based densification and pruning strategies, VCD and VCP, contribute most significantly to the overall acceleration.

**Component-wise Ablation.** We examine the effects of VCD, VCP, CB, and the absolute gradients from AbsGS [20] in Tab. 13. The ablation results indicate that nei-

Table 10. **Quantitative results of accelerating various backbones**.

| Method | Deep Blending [7] | | | | | | Tanks & Temples [12] | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ | FPS↑ | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ | FPS↑ |
| Mip-Splatting [21] | 23.94 | 29.35 | **0.899** | **0.241** | 3.48M | 219 | 14.57 | 23.77 | **0.856** | **0.158** | 2.36M | 300 |
| +Ours | **1.74** | **29.68** | **0.899** | 0.274 | **0.20M** | **698** | **2.01** | **24.18** | 0.843 | 0.200 | **0.36M** | **729** |
| Scaffold-GS [13] | 13.31 | **30.09** | **0.905** | **0.256** | 0.18M | 307 | 9.83 | 24.09 | **0.851** | **0.175** | 0.26M | 261 |
| +Ours | **2.82** | 30.00 | 0.900 | 0.267 | **0.08M** | **423** | **3.77** | **24.15** | 0.849 | 0.180 | **0.14M** | **332** |

Table 11. **Quantitative results of accelerating 3DCS [8].**

| Method | Mip-NeRF [1] | | | Deep Blending [7] | | | Tanks & Temples [12] | | |
|---|---|---|---|---|---|---|---|---|---|
| | Time↓ | PSNR↑ | $N_{GS}$ ↓ | Time↓ | PSNR↑ | $N_{GS}$ ↓ | Time↓ | PSNR↑ | $N_{GS}$ ↓ |
| 3DCS [8] | 67.72 | **27.22** | 2.61M | 93.93 | 29.35 | 3.07M | 58.07 | 23.76 | 2.10M |
| + Ours | **32.68** | **27.22** | **0.86M** | **36.88** | **29.39** | **0.52M** | **28.93** | **23.95** | **0.79M** |

Table 12. **Quantitative comparison of computational overhead.** We report the mean GPU memory usage (GB), peak GPU memory usage (GB), and storage size (MB).

| Method | MipNeRF-360 [1] | | | Deep Blending [7] | | | Tanks&Temples [12] | | |
|---|---|---|---|---|---|---|---|---|---|
| | mean GPU mem.↓ | peak GPU mem.↓ | Storage↓ | mean GPU mem.↓ | peak GPU mem.↓ | Storage↓ | mean GPU mem.↓ | peak GPU mem.↓ | Storage↓ |
| 3DGS [10] | 7.70 | 9.89 | 652 | 5.97 | 8.10 | 610 | 3.47 | 4.73 | 389 |
| Mini-Splatting [5] | 5.21 | 7.44 | 132 | 4.16 | 6.20 | 138 | 2.51 | 4.63 | 75 |
| Speedy-splat [6] | 4.97 | 7.03 | **74** | 3.82 | 5.03 | 61 | 2.19 | 2.66 | **45** |
| Taming-3DGS [14] | 4.78 | 5.88 | 170 | 3.39 | 4.01 | 73 | 1.94 | 2.43 | 79 |
| DashGaussian [3] | 6.71 | 9.96 | 595 | 4.79 | 7.75 | 482 | 2.81 | 4.49 | 301 |
| FastGS (Ours) | **4.58** | **5.21** | 99 | **3.34** | **3.77** | **54** | **1.91** | **2.27** | 60 |
| FastGS-Big (Ours) | 5.37 | 6.63 | 208 | 3.81 | 4.65 | 114 | 2.22 | 2.83 | 86 |

Table 13. **Ablation studies over the proposed methods.** Experiments are performed on the Mip-NeRF 360 dataset [1] with 3DGS-accel [10, 14] as the baseline.

| Method | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ |
|---|---|---|---|---|---|
| 3DGS-accel | 7.10 | 27.46 | 0.810 | 0.226 | 2.64M |
| +Abs grad | 6.85 | 27.60 | **0.817** | **0.216** | 2.29M |
| +CB. | 6.13 | 27.44 | 0.810 | 0.223 | 2.78M |
| +VCD. | 3.53 | 27.69 | 0.798 | 0.259 | 0.53M |
| +VCP. | 5.32 | **27.70** | 0.812 | 0.228 | 1.96M |
| Full | **1.93** | 27.56 | 0.797 | 0.261 | **0.40M** |

Table 14. **Ablation study on thresh $\tau_d$ on Tanks & Temples [12].**

| $\tau_d$ | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ |
|---|---|---|---|---|---|
| 1 | 1.44 | 24.17 | **0.841** | **0.205** | 0.30M |
| 2 | 1.42 | **24.18** | **0.841** | 0.207 | 0.28M |
| 5(ours) | 1.32 | 24.15 | 0.839 | 0.210 | 0.24M |
| 10 | 1.30 | 23.98 | 0.834 | 0.218 | 0.21M |
| 20 | 1.23 | 23.84 | 0.829 | 0.226 | 0.17M |
| 50 | 1.15 | 23.54 | 0.819 | 0.241 | 0.13M |
| 100 | **1.12** | 23.26 | 0.809 | 0.252 | **0.11M** |

ther absolute gradients nor CB effectively reduce the number of Gaussians, and their contribution to acceleration is limited. In contrast, **our proposed VCD and VCP achieve significantly greater speed-up, as they strictly control the Gaussian count, keeping it low throughout the entire training process, as shown in Fig. 2.**

**VCD Threshold $\tau_d$.** We study the effect of the densification threshold $\tau_d$ in VCD on Tanks & Temples [12], as shown in Tab. 14. A smaller $\tau_d$ allows more Gaussians to be densified, leading to slightly higher rendering quality but at the cost of increased training time and Gaussian count. Conversely, a larger $\tau_d$ reduces the number of Gaussians, accelerating training while slightly degrading quality. Our default choice of $\tau_d = 5$ achieves a balanced trade-off between efficiency and rendering fidelity.

**Number of sampled views $K$.** We study the effect of the number of sampled views on both training efficiency and rendering quality on the Mip-NeRF 360 [1] dataset. As shown in Tab. 15, using too few views slightly degrades quality, while sampling more views increases training time with minimal improvement. Our default choice of $K = 10$ achieves a good balance.

Table 15. **Ablation study on the number of sampled views $K$ on Mip-NeRF 360 [1].** "all" indicates using all training views.

| $K$ | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ |
|---|---|---|---|---|---|
| 5 | **1.91** | 27.47 | 0.795 | 0.265 | **0.36M** |
| 10(ours) | 1.93 | **27.56** | **0.797** | **0.261** | 0.40M |
| 20 | 2.03 | 27.55 | **0.797** | **0.261** | 0.43M |
| 50 | 2.10 | 27.55 | **0.797** | 0.262 | 0.43M |
| all | 2.29 | 27.54 | 0.796 | 0.263 | 0.42M |

Table 16. **Comparison of different optimization strategies.**

| Method | Optimizer | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ |
|---|---|---|---|---|---|---|
| FastGS | Sparse Adam [14] | **1.93** | 27.37 | 0.792 | 0.270 | **0.37M** |
| | Optimizer schedule | **1.93** | **27.56** | **0.797** | **0.261** | 0.40M |

## 12. Scene-wise Results

We present the quantitative results in Tab. 17, Tab. 18, and Tab. 19, and provide the qualitative comparisons in Fig. 7, Fig. 8 and Fig. 9.

## References


[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022. 3, 4, 5

[2] Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 2024. 2

[3] Youyu Chen, Junjun Jiang, Kui Jiang, Xiao Tang, Zhihao Li, Xianming Liu, and Yinyu Nie. Dashgaussian: Optimizing 3d gaussian splatting in 200 seconds. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 11146–11155, 2025. 3, 5, 6

[4] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12882–12891, 2022. 5

[5] Guangchi Fang and Bing Wang. Mini-splatting: Representing scenes with a constrained number of gaussians. In *European Conference on Computer Vision*, pages 165–181. Springer, 2024. 3, 5, 6

[6] Alex Hanson, Allen Tu, Geng Lin, Vasu Singla, Matthias Zwicker, and Tom Goldstein. Speedy-splat: Fast 3d gaussian splatting with sparse pixels and sparse primitives. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21537–21546, 2025. 1, 3, 5, 6

[7] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)*, 37(6):1–15, 2018. 3

[8] Jan Held, Renaud Vandeghen, Abdullah Hamdi, Adrien Deliege, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi, Bernard Ghanem, and Marc Van Droogenbroeck. 3d convex splatting: Radiance field rendering with 3d smooth convexes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21360–21369, 2025. 2, 3

[9] Huajian Huang, Longwei Li, Cheng Hui, and Sai-Kit Yeung. Photo-slam: Real-time simultaneous localization and photo-realistic mapping for monocular, stereo, and rgb-d cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 2

[10] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 1, 2, 3, 5, 6

[11] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[12] 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. 3, 6

[13] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024. 2, 3

[14] Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando De La Torre. Taming 3dgs: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024. 1, 2, 3, 4, 5, 6

[15] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (ToG)*, 38(4):1–14, 2019. 2

[16] Hyunwoo Park, Gun Ryu, and Wonjun Kim. Dropgaussian: Structural regularization for sparse-view gaussian splatting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21600–21609, 2025. 2

[17] Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. Octree-gs: Towards consistent real-time rendering with lod-structured 3d gaussians. *arXiv preprint arXiv:2403.17898*, 2024. 2

[18] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 2

[19] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20331–20341, 2024. 2

Table 17. Scene-wise quantitative results over the Mip-NeRF 360 dataset [1].

| Method | bicycle | | | | | | flowers | | | | | | garden | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ | FPS↑ | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ | FPS↑ | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ | FPS↑ |
| 3DGS [10] | 27.97 | 25.14 | 0.748 | 0.242 | 4.71M | 80 | 18.98 | 21.30 | 0.586 | 0.360 | 2.82M | 162 | 26.78 | 27.34 | 0.857 | 0.122 | 4.19M | 103 |
| Mini-Splatting [5] | 16.17 | 25.23 | 0.764 | 0.241 | 0.59M | 564 | 17.22 | 21.43 | 0.614 | 0.341 | 0.63M | 511 | 15.97 | 27.36 | 0.806 | 0.215 | 0.67M | 487 |
| Speedy-Splat [6] | 15.87 | 24.79 | 0.704 | 0.333 | 0.58M | 460 | 13.38 | 21.21 | 0.560 | 0.418 | 0.34M | 526 | 15.73 | 26.69 | 0.814 | 0.214 | 0.52M | 474 |
| Taming-3DGS [14] | 5.65 | 24.72 | 0.693 | 0.332 | 0.81M | 199 | 4.97 | 21.10 | 0.552 | 0.416 | 0.58M | 233 | 9.82 | 27.42 | 0.851 | 0.138 | 2.08M | 177 |
| DashGaussian [3] | 9.93 | 25.31 | 0.763 | 0.222 | 4.70M | 105 | 7.05 | 21.78 | 0.604 | 0.341 | 2.82M | 158 | 8.27 | 27.57 | 0.857 | 0.131 | 3.37M | 153 |
| FastGS | 1.92 | 24.84 | 0.714 | 0.310 | 0.54M | 582 | 1.95 | 21.21 | 0.560 | 0.406 | 0.49M | 555 | 2.47 | 27.20 | 0.836 | 0.174 | 0.74M | 538 |
| FastGS-Big | 2.59 | 25.26 | 0.755 | 0.245 | 1.55M | 463 | 3.22 | 21.60 | 0.602 | 0.341 | 1.14M | 468 | 6.50 | 27.56 | 0.864 | 0.110 | 2.64M | 332 |

| Method | stump | | | | | | treehill | | | | | | room | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ | FPS↑ | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ | FPS↑ | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ | FPS↑ |
| 3DGS [10] | 21.77 | 26.64 | 0.768 | 0.244 | 4.05M | 130 | 20.33 | 22.59 | 0.636 | 0.347 | 3.01M | 136 | 18.78 | 31.71 | 0.927 | 0.197 | 1.25M | 164 |
| Mini-Splatting [5] | 16.52 | 26.80 | 0.839 | 0.161 | 0.67M | 521 | 17.05 | 22.76 | 0.656 | 0.326 | 0.63M | 487 | 18.00 | 31.48 | 0.928 | 0.190 | 0.39M | 506 |
| Speedy-Splat [6] | 13.77 | 26.67 | 0.765 | 0.288 | 0.46M | 480 | 12.90 | 22.48 | 0.590 | 0.462 | 0.32M | 548 | 12.05 | 30.83 | 0.903 | 0.258 | 0.11M | 617 |
| Taming-3DGS [14] | 3.93 | 26.05 | 0.729 | 0.324 | 0.48M | 280 | 5.37 | 22.92 | 0.628 | 0.395 | 0.79M | 214 | 3.88 | 31.64 | 0.917 | 0.227 | 0.23M | 230 |
| DashGaussian [3] | 6.57 | 27.17 | 0.783 | 0.229 | 3.42M | 164 | 8.20 | 22.94 | 0.640 | 0.333 | 3.42M | 134 | 4.00 | 31.81 | 0.924 | 0.205 | 1.04M | 182 |
| FastGS | 1.72 | 26.65 | 0.756 | 0.297 | 0.39M | 576 | 1.72 | 22.94 | 0.612 | 0.429 | 0.38M | 568 | 1.62 | 31.98 | 0.920 | 0.217 | 0.21M | 632 |
| FastGS-Big | 2.88 | 27.18 | 0.786 | 0.240 | 1.06M | 489 | 2.78 | 22.83 | 0.632 | 0.378 | 1.01M | 507 | 2.38 | 32.20 | 0.929 | 0.189 | 0.57M | 577 |

| Method | counter | | | | | | kitchen | | | | | | bonsai | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ | FPS↑ | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ | FPS↑ | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ | FPS↑ |
| 3DGS [10] | 17.58 | 29.16 | 0.915 | 0.183 | 1.05M | 171 | 21.30 | 31.54 | 0.932 | 0.116 | 1.53M | 144 | 14.90 | 32.37 | 0.946 | 0.180 | 1.07M | 224 |
| Mini-Splatting [5] | 9.83 | 28.65 | 0.911 | 0.181 | 0.41M | 590 | 10.23 | 31.05 | 0.930 | 0.120 | 0.44M | 614 | 11.02 | 31.24 | 0.943 | 0.177 | 0.36M | 661 |
| Speedy-Splat [6] | 12.10 | 28.22 | 0.876 | 0.259 | 0.10M | 606 | 13.25 | 30.09 | 0.895 | 0.195 | 0.11M | 608 | 11.37 | 31.16 | 0.925 | 0.228 | 0.13M | 652 |
| Taming-3DGS [14] | 4.60 | 29.20 | 0.909 | 0.200 | 0.31M | 221 | 3.48 | 31.84 | 0.929 | 0.128 | 0.48M | 209 | 4.60 | 32.40 | 0.942 | 0.193 | 0.41M | 227 |
| DashGaussian [3] | 3.95 | 29.11 | 0.911 | 0.191 | 0.85M | 162 | 5.52 | 31.69 | 0.927 | 0.129 | 1.18M | 135 | 3.95 | 32.15 | 0.945 | 0.180 | 0.82M | 193 |
| FastGS | 1.83 | 29.15 | 0.907 | 0.204 | 0.21M | 596 | 2.42 | 31.87 | 0.929 | 0.127 | 0.38M | 543 | 1.83 | 32.19 | 0.942 | 0.191 | 0.28M | 622 |
| FastGS-Big | 2.62 | 29.57 | 0.917 | 0.177 | 0.47M | 522 | 5.15 | 32.17 | 0.938 | 0.105 | 1.18M | 395 | 3.22 | 32.97 | 0.953 | 0.161 | 0.85M | 498 |

Table 18. Scene-wise quantitative results over the Deep Blending dataset [4].

| Method | playroom | | | | | | drjohnson | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ | FPS↑ | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ | FPS↑ |
| 3DGS [10] | 16.75 | 30.14 | 0.904 | 0.243 | 1.85M | 189 | 22.78 | 29.28 | 0.902 | 0.239 | 3.07M | 126 |
| Mini-Splatting [5] | 12.30 | 30.47 | 0.908 | 0.241 | 0.51M | 618 | 14.40 | 29.51 | 0.905 | 0.246 | 0.60M | 629 |
| Speedy-Splat [6] | 9.70 | 29.77 | 0.898 | 0.274 | 0.18M | 695 | 11.80 | 29.07 | 0.898 | 0.269 | 0.31M | 633 |
| Taming-3DGS [14] | 3.35 | 29.96 | 0.901 | 0.264 | 0.40M | 354 | 2.77 | 29.04 | 0.888 | 0.292 | 0.19M | 349 |
| DashGaussian [3] | 4.70 | 30.17 | 0.909 | 0.243 | 2.38M | 233 | 3.62 | 29.13 | 0.903 | 0.250 | 1.51M | 182 |
| FastGS | 1.22 | 30.57 | 0.905 | 0.266 | 0.19M | 727 | 1.35 | 29.50 | 0.898 | 0.275 | 0.25M | 700 |
| FastGS-Big | 1.93 | 30.55 | 0.909 | 0.239 | 0.60M | 619 | 2.07 | 29.69 | 0.905 | 0.247 | 0.70M | 595 |


[20] Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. Absgs: Recovering fine details in 3d gaussian splatting. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 1053–1061, 2024. 1, 2

[21] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19447–19456, 2024. 2, 3

Table 19. Scene-wise quantitative results over the Tanks & Temples dataset [12].

| Method | truck | | | | | | train | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ | FPS↑ | Time↓ | PSNR↑ | SSIM↑ | LPIPS↓ | $N_{GS}$ ↓ | FPS↑ |
| 3DGS [10] | 12.50 | 25.39 | 0.881 | 0.142 | 2.05M | 186 | 10.18 | 22.03 | 0.818 | 0.198 | 1.09M | 203 |
| Mini-Splatting [5] | 9.08 | 25.32 | 0.879 | 0.139 | 0.32M | 716 | 9.03 | 21.60 | 0.809 | 0.223 | 0.28M | 795 |
| Speedy-Splat [6] | 7.22 | 25.18 | 0.863 | 0.192 | 0.26M | 648 | 5.42 | 21.59 | 0.768 | 0.292 | 0.11M | 733 |
| Taming-3DGS [14] | 2.38 | 25.27 | 0.865 | 0.187 | 0.27M | 409 | 3.03 | 22.50 | 0.802 | 0.240 | 0.37M | 348 |
| DashGaussian [3] | 4.25 | 25.80 | 0.886 | 0.150 | 1.43M | 257 | 4.32 | 22.19 | 0.819 | 0.206 | 1.00M | 222 |
| FastGS | 1.30 | 25.73 | 0.872 | 0.178 | 0.25M | 666 | 1.33 | 22.57 | 0.805 | 0.242 | 0.23M | 644 |
| FastGS-Big | 2.13 | 26.09 | 0.886 | 0.140 | 0.63M | 579 | 1.93 | 22.68 | 0.824 | 0.210 | 0.46M | 558 |

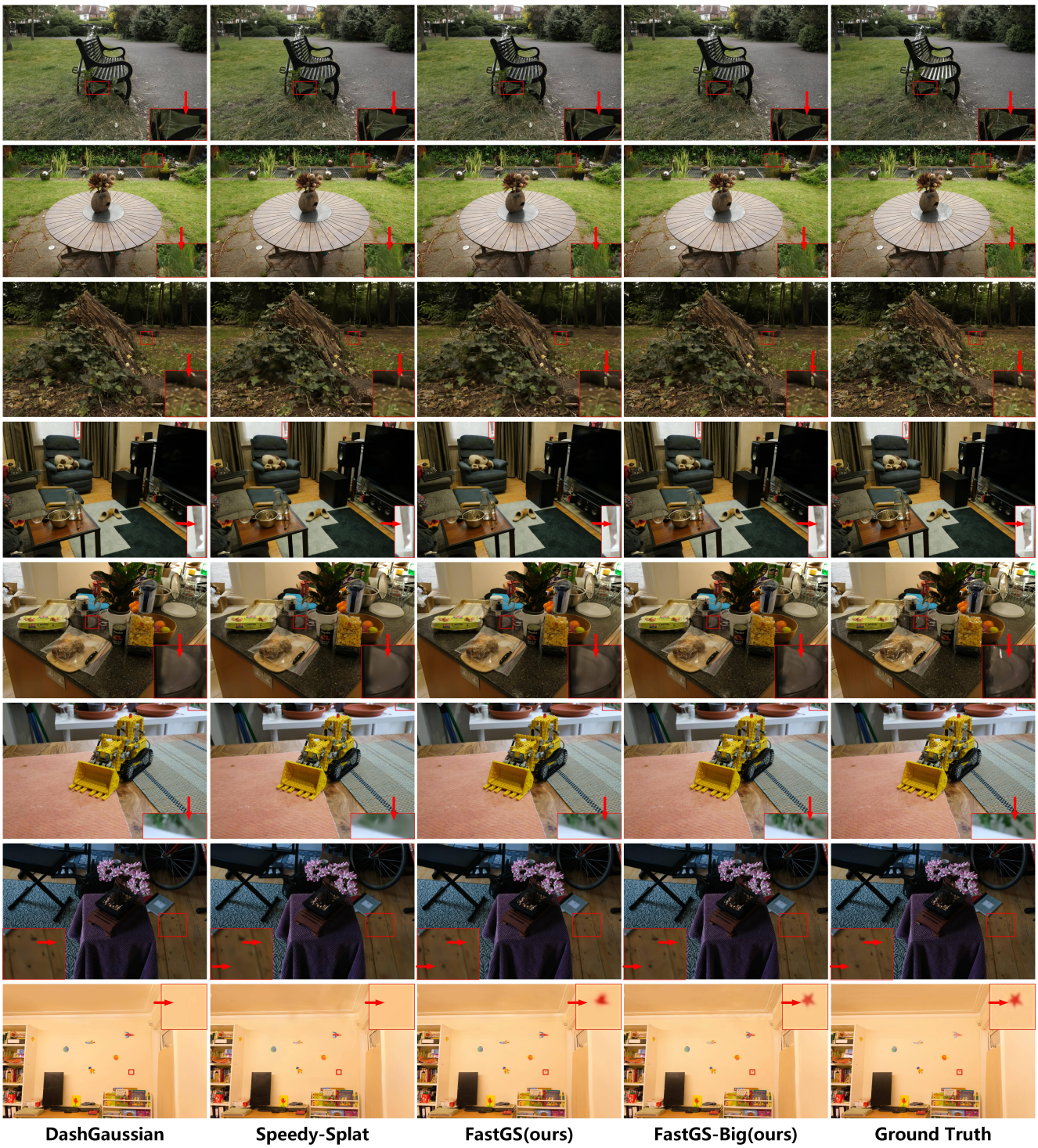


Figure 7. Additional visual comparisons on the *bicycle*, *garden*, *stump*, *room*, *counter*, *kitchen*, *bonsai*, and *playroom* scenes.

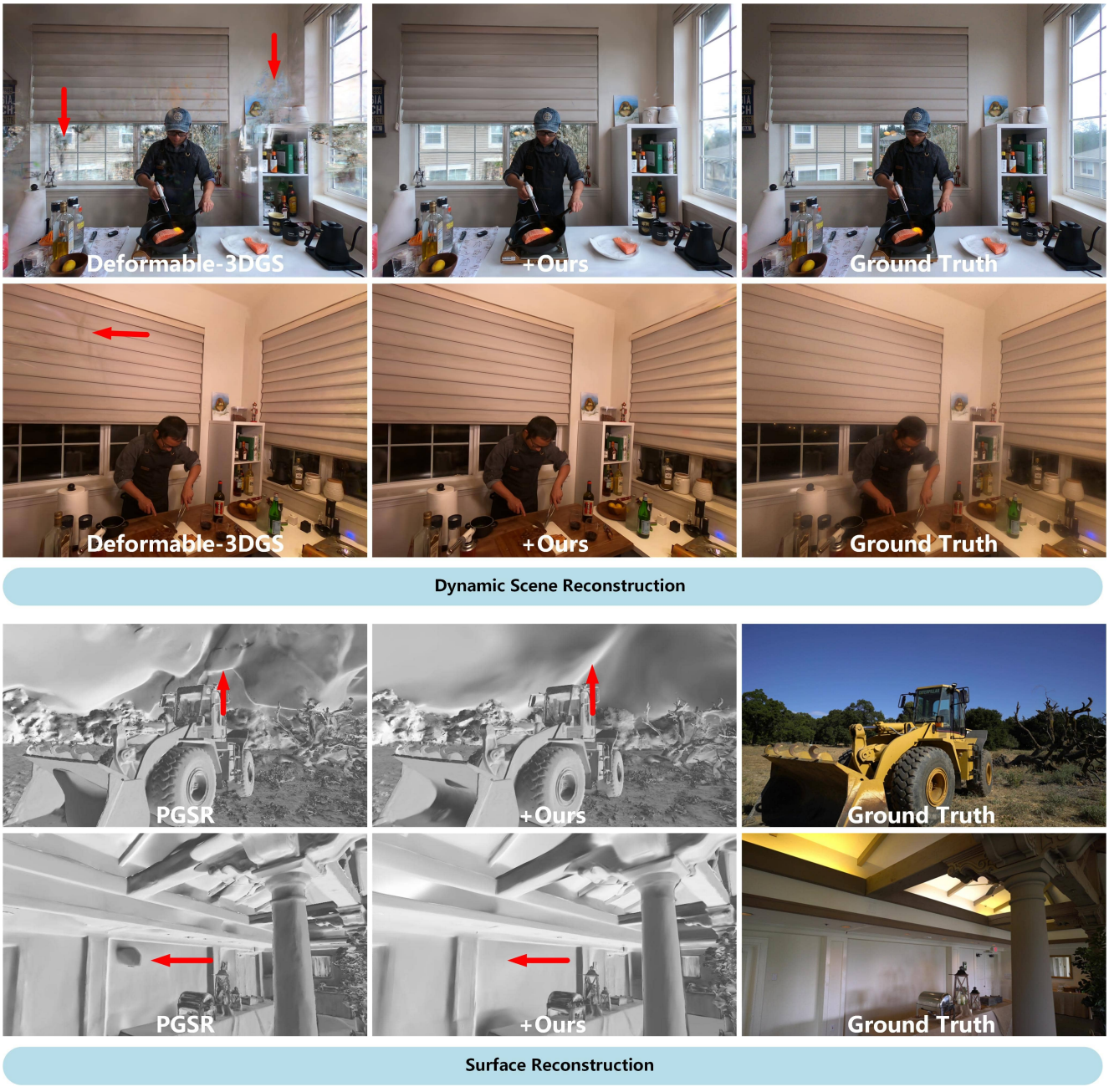


Figure 8. Additional visual comparisons on different tasks, including the *flame_salmon*, *cut_roasted_beef*, *Caterpillar*, *Meetingroom* scenes.

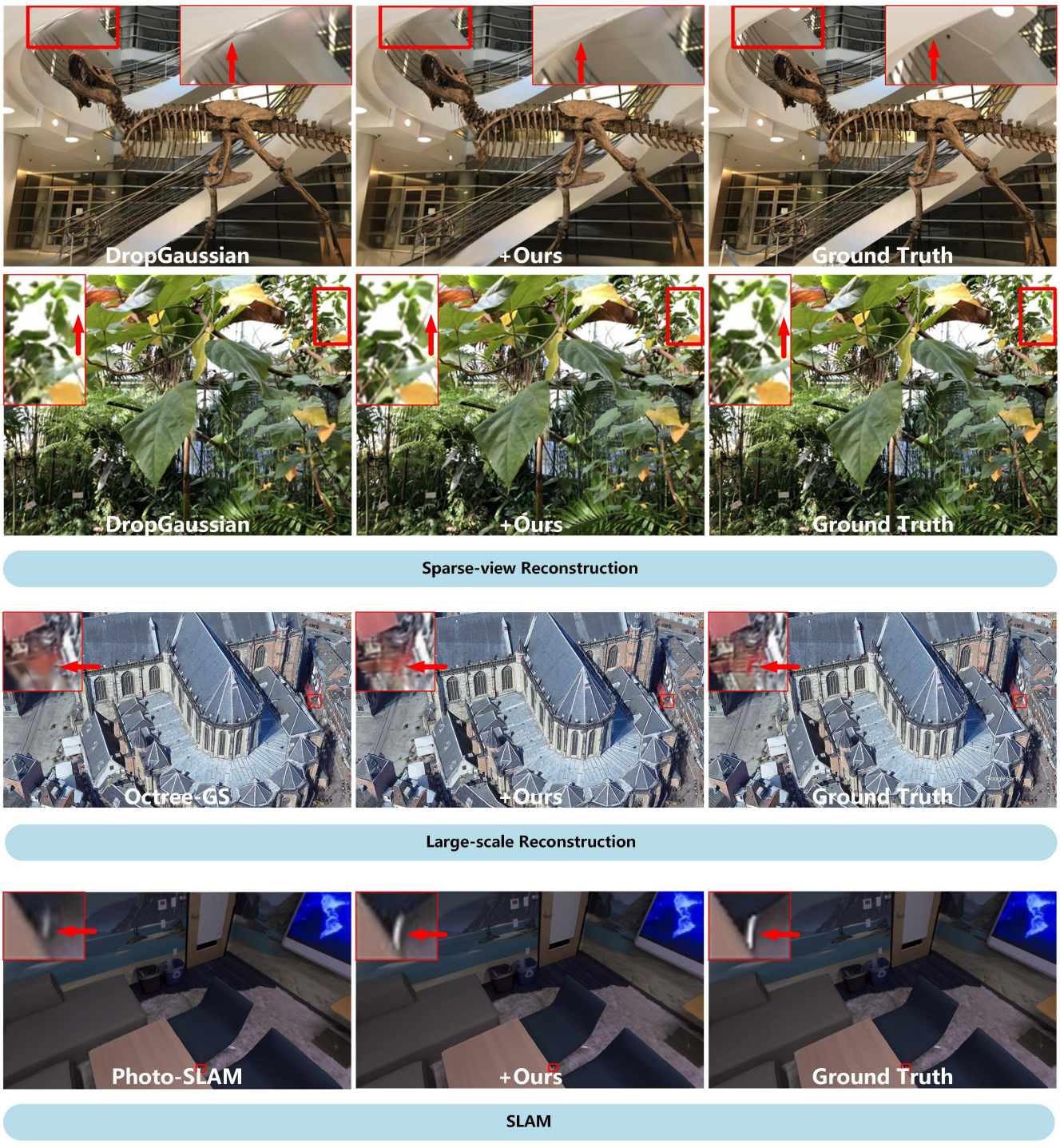


Figure 9. Additional visual comparisons on different tasks, including the *trex*, *leaves*, *amsterdam*, and *office0* scenes.