Textured Gaussians for Enhanced 3D Scene Appearance Modeling

Appendix

A. Results from Original 3DGS Paper

We show the quantitative results (PSNR / SSIM / LPIPS) of all 5 datasets reported in the original 3DGS paper [29] and the performance of our own 3DGS implementation in Table A.1. Since we used a modified version of 3DGS described in Gaussian Opacity Fields [68], there are slight differences in the results. We compare our textured Gaussians model with our own modified 3DGS implementation (which we label as 3DGS* in the main paper) for a fair comparison, since our algorithm is built on top of that. However, the concept of Textured Gaussians could also be easily applied to the original 3DGS model.

A note on LPIPS. The LPIPS values reported in the original 3DGS paper [29] are underestimated, leading to additional performance discrepancies compared to our method. This issue was pointed out in [6] and confirmed in private correspondence with the authors of 3DGS [29]. To avoid confusion, we underline the underestimated LPIPS values in Table A.1 and report the correct LPIPS values of our own implementation.

B. Custom 3DGS Implementation Details

Our custom 3DGS implementation closely follow the modified 3DGS algorithm described in Gaussian Opacity Fields [68]. In this section, we describe how each component in our own implementation of 3DGS differs from the original 3DGS paper [29].

Gaussian Value Calculation. In 3DGS, 3D scenes are represented with 3D Gaussians, and images are rendered using differentiable volume splatting. Specifically, 3DGS explicitly defines 3D Gaussians by their 3D covariance matrix $\mathbf{\Sigma}_i \in \mathbb{R}^{3\times3}$ and center $\boldsymbol{\mu}_i \in \mathbb{R}^3$ (the index i indicating the i^{th} Gaussian), where the 3D Gaussian function value at point $\mathbf{x} \in \mathbb{R}^3$ is defined by:

$$\mathcal{G}_i(\mathbf{x}) = \exp(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)\mathbf{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)) \tag{A.1}$$

where the covariance matrix $\mathbf{\Sigma} = \mathbf{RSS}^\top\mathbf{R}^\top$ is factorized into the rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ and the scale matrix $\mathbf{S} \in \mathbb{R}^{3\times3}$. When rendering the color of a pixel $\mathbf{p} \in \mathbb{R}^3$, 3D Gaussians are transformed from world coordinates to camera coordinates via a world-to-camera transformation matrix $\mathbf{W} \in \mathbb{R}^{3\times3}$ and projected to the 2D image plane via a local affine transformation $\mathbf{J} \in \mathbb{R}^{3\times3}$. The transformed 3D covariance $\mathbf{\Sigma}'$ can be calculated as:

$$\mathbf{\Sigma}' = \mathbf{JW\Sigma W}^\top\mathbf{J}^\top \tag{A.2}$$

The covariance Σ_{2D} of the 2D Gaussian $\mathcal{G}^{\text{2D}}$ splatted on the image plane can be *approximated* as extracting the first two rows and columns of the transformed 3D covariance, $\Sigma_{\text{2D}} = [\Sigma']_{\{1,2\},\{1,2\}} \in \mathbb{R}^{2\times2}$ using the matrix minor notation. The alpha value α_i used to perform alpha compisting at the pixel location can then be calculated by:

$$\alpha_i = \mathcal{G}_i^{\text{2D}}(\mathbf{p}) \cdot o_i \tag{A.3}$$

where o_i is the opacity of the i^{th} 3D Gaussian.

Instead of *approximating* 2D Gaussian values by splatting 3D Gaussians, we cast rays from the pixel currently being rendered and calculate the *exact* 3D Gaussian value evaluated at the ray-Gaussian intersection point. This forumlation fits nicely with our Textured Gaussians algorithm since ray-Gaussian intersection calculation is also required from texture UV-mapping. The alpha value α_i used for alpha blending at pixel $\mathbf{p}$ is therefore defined by:

$$\alpha_i = \mathcal{G}_i(\mathbf{x}) \cdot o_i \tag{A.4}$$

where $\mathbf{x}$ is the ray-Gaussian intersection point and $\mathcal{G}_i$ is the i^{th} 3D Gaussian.

Adaptive Density Control (ADC). The optimization of 3DGS starts from a sparse structure-from-motion (SfM) point cloud and progressively densifies Gaussians either through cloning or splitting. In the original 3DGS, this densification process is guided by a score that is defined by the magnitude of the view/screen-space positional gradient $\frac{dL}{d\boldsymbol{\mu}'}$ of the Gaussian:

$$\left\| \frac{dL}{d\boldsymbol{\mu}'} \right\| = \left\| \sum_j \frac{dL}{d\mathbf{p}_j} \cdot \frac{d\mathbf{p}_j}{d\boldsymbol{\mu}'} \right\| \tag{A.5}$$

where μ' is the center of the projected Gaussian and $\mathbf{p}_j$ are the pixels the Gaussian contributed to. If this score is larger than a predfined threshold τ, the Gaussian will be cloned or splitted.

As pointed out in Gaussian Opacity Fields [68], this score is not effective in identifying overly blurred areas for Gaussian densification, and the authors proposed an alternative *sum-of-magnitudes* score to replace the original magnitude-of-sums:

$$\sum_j \left\| \frac{dL}{d\mathbf{p}_j} \cdot \frac{d\mathbf{p}_j}{d\boldsymbol{\mu}'} \right\| \tag{A.6}$$

We use this revised densification score to perform adaptive density control throughout optimization.

Method	Blender [37]	Mip-NeRF 360 [37]	DTU [27]	Tanks and Temples [30]	Deep Blending [20]
Kerbl et al. [29]	33.32 / ——— / ———	27.21 / 0.815 / 0.214	——— / ——— / ———	23.14 / 0.841 / 0.183	29.41 / 0.903 / 0.243
3DGS*	33.09 / 0.967 / 0.044	27.28 / 0.832 / 0.187	33.54 / 0.970 / 0.055	24.18 / 0.854 / 0.175	28.04 / 0.894 / 0.271

Table A.1. Quantitative performance comparison (PSNR ↑ / SSIM ↑ / LPIPS ↓) between our own 3DGS implementation (labelled as 3DGS* in the main paper) and the original 3DGS paper [29]. The under-estimated LPIPS values from the original 3DGS paper are underlined.

Learning Rates and other Hyperparameters. We use all default learning rates and hyperparameters in the original 3DGS implementation [29]. Please refer to their released code for the specific values. For each Textured Gaussians experiment and ablation study, we use the same texture map resolution and set the texture map learning rate to be 0.001 for all datasets and do not require per-dataset tuning for texture map parameters and learning rates.

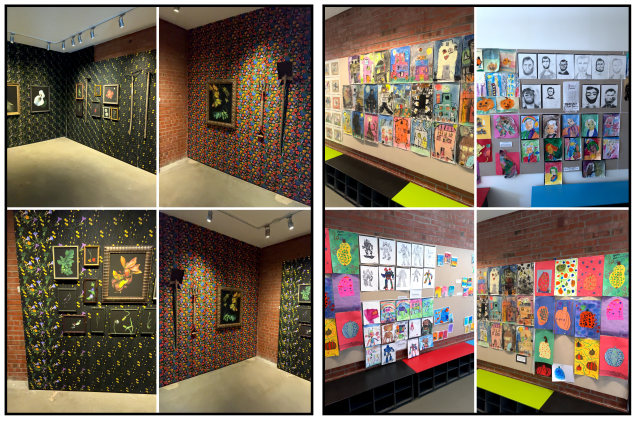
C. Additional Details on Implementation, Optimization, and Dataset Preparation

Implementation Details. We implement our algorithm using PyTorch and custom CUDA kernels that perform fast ray-Gaussian intersection calculation, UV mapping, and texture lookup. All experiments are run on clusters of Nvidia H100 GPUs.

To avoid integer overflows during UV mapping, we only intersect Gaussians within $\pm 3\sigma$ along the two major axes and map the 2D texture map to the $\pm 3\sigma$ range ($m = 3$). The color of intersection points outside of this range is simply assigned as black, that is $c_i^{\text{tex}}(u, v) = 0$. We found that this implementation greatly improves numerical stability, especially when viewing Gaussians from grazing angles. This also has very little effect on the final performance since the opacity of the Gaussian outside of the $\pm 3\sigma$ range is negligible (less than 0.01). We also found that dynamically adjusting the intersection plane based on the two longest axes of the ellipsoid leads to empirically better performance.

Optimization Details. We set the learning rate of the texture maps to 0.001 and initialize the RGB colors of the texture map to a low value ($\frac{25}{255}$) since the optimized spherical harmonic coefficients of the first stage should have already learned to reconstruct the average color of the pixels within the Gaussian extent, and the texture map should only learn to reconstruct the residual color in the second stage of optimization. The alpha channel of the texture map is initialized to 1.

Since our training procedure has two stages (3DGS pretraiing and Textured Gaussians optimization), the training time of our models are roughly two times the training time of 3DGS. Our custom CUDA kernels that perform ray-Gaussian intersection and texture lookup adds very little overhead to the original 3DGS rendering process, hence the inference time of our Textured Gaussians model is approximately the same as that of 3DGS.



"flower gallery" 161 images *"children's art"* 198 images

Figure A.1. **Sample training views and number of training images in our custom-captured datasets.** We captured room-level scenes containing highly-detailed artworks to demonstrate the effectiveness of our method.

Dataset Preparation Details. We downscale the DTU dataset to 800×600 image resolution following Xu et al. [63], and conform to the dataset preparation and evaluation protocols described in Kerbl et al. [29] for the remaining four datasets (Blender [37], Mip-NeRF 360 [4], Tanks and Temples [30], and Deep Blending [20]). Following Mildenhall et al. [37] and Xu et al. [63], we use the alpha channel of the images to create black and white backgrounds for the DTU dataset and the Blender dataset, respectively.

For our custom dataset, we capture two room-scale scenes (*"flower gallery"* and *"children's art"*) at a local art center. The images from each dataset are captured by a photographer standing at the center of the room scanning the whole room in a 360 degrees fashion. The number of captured images for each scene in shown in Figure A.1.

DTU Dataset Baseline Comparisons Details For comparisons with Mip-NeRF 360 [4] and Instant-NGP [39] on the DTU dataset, we run the open-source code provided by the authors (code for Mip-NeRF 360 and code for Instant-NGP). For Mip-NeRF 360 optimization, we slightly modified the configuration file `multinerf/configs/blender_256.gin`,

which is the default configuration for their Blender experiments. Specifically, we set parameters `Model.bg_intensity_range = (0, 0)` and `Model.opaque_background = False` because the DTU dataset scenes all have black backgrounds. The near and far clipping planes are set as $(2, 6)$, respectively, based on the input point cloud and camera positions. The capacity of this model is more than sufficient because DTU dataset has both lower image resolution and less training images than the Blender dataset (one-third to one-fourth of the training images).

For Instant-NGP optimization, we used the configuration file `instant-ngp/configs/nerf/base.json`, which is the default configuration for all Instant-NGP experiments, including experiments on scene-level datasets.

D. Additional Quantitative Results

NVS with Varying Numbers of Gaussians and Fixed Amount of Texels. We show the quantitative novel-view synthesis results of our RGBA Textured Gaussians models and 3DGS* with varying numbers of Gaussians in terms of PSNR, SSIM, and LPIPS for the 5 test datasets in Table A.3 and Figure A.4. We allocate a fixed amount of texels to each of our models. Hence, the texture map resolutions of models with more Gaussians are smaller. Our method consistently outperforms 3DGS* in terms of PSNR and LPIPS when using different numbers of Gaussians, and the performance improvement is especially large when using fewer Gaussians.

NVS with Varying Texture Map Resolution and Fixed Number of Gaussians. We show quantitative novel-view synthesis results of our RGBA Textured Gaussians models with a fixed number of Gaussians and varying texture map resolutions in terms of PSNR, SSIM, and LPIPS in Figure A.5. We optimized all models with 1% of the default optimized number of Gaussians. As expected, the quantitative performance becomes better as the texture resolution increases since the detailed appearances are reconstructed better.

NVS with Varying Number of Gaussians and Fixed Texture Map Resolution. We show quantitative novel-view synthesis results of our RGBA Textured Gaussians models with a fixed texture map resolution and varying number of Gaussians in terms of PSNR, SSIM, and LPIPS in Figure A.6. We use a 4×4 texture map for all models. As the number of Gaussians increase, the quantitative performances improve since detailed geometry and appearance are reconstructed better with more, and therefore smaller, Gaussians.

Ablation on Texture Map Variants. We ablate variants of our model that use different texture maps, namely alpha-only, RGB, and RGBA texture maps, and the same number of Gaussians. Experiments are conducted on all five standard benchmark datasets with the default optimized number (top row) and 1% of the default optimized number (bottom row) of Gaussians. We report the PSNR/SSIM/LPIPS values of the novel-view synthesis results. From Table A.4 and Figure A.7, we see that our model achieves the best performance with RGBA textures. Interestingly, using alpha-only textures already outperforms 3DGS* and our RGB textures model despite being one-third the size, striking a better balance between performance and model size. This is because alpha-textured Gaussians can represent complex shapes through spatially varying opacity and reconstruct high-frequency textures through spatially varying alpha composition. In contrast, RGB-textured Gaussians are still limited to representing ellipsoids.

Model	3DGS	Ours (w/ 4×4 RGBA textures)
Number of Gaussians (M)	2.12 M	2.12 M
Model size (MB)	125.3 MB	261.2 MB
GPU memory (GB)	2.4	4.0
Inference time per frame (s)	0.073	0.123

Table A.2. We report various efficiency metrics of 3DGS and our Textured Gaussians model. With the same number of Gaussians, the model size and GPU usage of Textured Gaussians are around $2\times$ that of 3DGS. The inference speed is also slower due to additional textured map queries.

Computational Efficiency Analysis. The total training time of Textured Gaussians is $2\times$ that of 3DGS due to the two-stage training process. In Table A.2, we report various efficiency metrics, aggregated over all 26 test scenes, of our Textured Gaussians vs. 3DGS using the same number of Gaussians and the setup of Table 1 in the main paper (100% of the default number of optimized Gaussians and 4×4 texture maps).

As for the other texture map variants of Textured Gaussians, the model sizes of using alpha-only, RGB, and RGBA textures are $1.27\times, 1.82\times, 2.08\times$ that of 3DGS with the same number of Gaussians. Note that using alpha-only textures only imposes slight memory overhead and performs better than 3DGS, according to Figure 8 in the paper.

E. Additional Qualitative Results

In this section, we show qualitative NVS results and refer the readers to the project website for novel-view **video** renderings of selected scenes for each experiment.

NVS with Varying Numbers of Gaussians and Fixed Amount of Texels. We show novel-view synthesis results from the 24 test scenes that we used for evaluation in Figures A.8 to A.12. We show results of 3DGS* and our Textured Gaussians model using the default optimized number and 1% of the default optimized number of Gaussians. Our method achieves sharper details than 3DGS* in both cases. The differences between the two methods are espe-

cially clear when using fewer Gaussians, as 3DGS* struggles to reconstruct fine-grained textures while our method achieves decent image quality.

We additionally show novel-view video renderings of selected scenes optimized using 3DGS* and our RGBA Textued Gaussians model with varying number of Gaussians (continuously varying from 1% to 100% of the default number of optimized Gaussians) in the project website. We highly recommend readers to try out the interactive website and toggle between models optimized with different numbers of Gaussians to see how novel-view synthesis quality improves as the number of Gaussians increases.

NVS with Varying Texture Map Resolution and Fixed Number of Gaussians. We show qualitative novel-view synthesis results of our RGBA Textured Gaussians models with a fixed number of Gaussians and varying texture map resolutions in Figures A.13 to A.17. Due to space constraints, we show results of our Textured Gaussians models with texture map resolutions $\mathcal{T} = 2, 5, 20, 40$.

As expected, NVS performance improves as the texture map resolution increases, since detailed appearances can be reconstructed better with smaller texture feature sizes.

NVS with Varying Number of Gaussians and Fixed Texture Map Resolution. We show qualitative novel-view synthesis results of our RGBA Textured Gaussians models with a fixed texture map resolution and varying numbers of Gaussians in Figures A.18 to A.22. Due to space constraints, we show results of our Textured Gaussians models 1%, 5%, 20%, and 100% of the default number of optimized Gaussians.

As expected, NVS performance improves as the number of Gaussians increase since details can be reconstructed better with more and therefore, smaller, Gaussians.

Ablation on Texture Map Variants. We show novel view synthesis results of our Textured Gaussians model optimized using different texture map variants, namely alpha-only, RGB, and RGBA textures, in Figures A.23 to A.27. Here, we show the results of our models and 3DGS* with 1% of the default optimized number of Gaussians to maximize the visual difference between different variants to fully demonstrate the effectiveness of our method.

From the results, we see that the alpha-textures model and the RGBA textures model achieve better qualitative performance than the RGB textures model. This is because alpha-textured Gaussians are able to both reconstruct fine-grained textures through spatially varying alpha composition and represent complex shapes through spatially varying opacity. On the other hand, RGB-textured Gaussians are still only able to represent ellipsoids. This is clearly observed in scenes that contain complex geometric structures, such as the *ship* scene in the Blender dataset (last row) and the lego truck in the *kitchen* scene in the mipNeRF360 dataset (6th row).

We additionally show novel-view video renderings of selected scenes optimized using 3DGS* and different variants (alpha, RGB, and RGBA) of our Textured Gaussians model in the project website. We highly recommend readers to try out the interactive website and toggle between models optimized with different texture map variants to see how alpha and RGB texture maps affect novel-view rendering quality.

Color Component Decompositions. We optimized our Textured Gaussians model with alpha-only, RGB, and RGBA textures with the same number of Gaussians and show the color decomposition of the final rendered color into the two color components $\mathbf{c}_{\text{base}}$ and $\mathbf{c}_{\text{tex}}$ in Figures A.28 to A.32. Specifically, we show the *alpha-modulated and composited* colors of the color components. 3DGS* models optimized with the same number of Gaussians are also shown for better comparison. We optimize all models with 1% of the default optimized number of Gaussians to maximize the visual difference between our method and 3DGS* to fully demonstrate the effectiveness of our method.

We see from the results of the alpha-only and RGBA textures model that although the color of $\mathbf{c}_{\text{base}}$ comes from spatially constant spherical harmonic coefficients, the alpha-modulated $\mathbf{c}_{\text{base}}$ is able to reconstruct high-frequency textures due to the spatially varying alpha composition. On the other hand, $\mathbf{c}_{\text{base}}$ in the RGB textures model cannot reconstruct high-frequency textures, and $\mathbf{c}_{\text{tex}}$ is used to reconstruct detailed appearance.

We additionally show novel-view video renderings of the color component decompositions of selected scenes optimized using 3DGS* and different variants (alpha, RGB, and RGBA) of our Textured Gaussians model in the project website. We highly recommend readers to try out the interactive website and toggle between models optimized with different texture map variants to see how the color component decompositions differ from one another.

3DGS vs. 2DGS-based implementation. We find that Textured Gaussians optimization naturally leads to a large proportion of flat Gaussians, as the distributions of effective rank are shown Figure A.2. Note that flat 3D Gaussians have an effective rank of 2 [26]. This observation justifies our design choice to flatten 3D Gaussians to approximate 2D Gaussians, similar to prior works [16, 68].

We also used various designs to reduce artifacts especially when rendering flat Gaussians from grazing viewpoints. As shown in Figure A.3, intersecting Gaussians only within the $\pm 3\sigma$ range and using SH base colors greatly reduce rendering artifacts.

Method	Blender [37]	Mip-NeRF 360 [37]	DTU [27]	Tanks and Temples [30]	Deep Blending [20]
3DGS* (1%)	26.89 / 0.9160 / 0.1165	22.37 / 0.6293 / 0.4774	30.88 / 0.9320 / 0.1581	19.90 / 0.6736 / 0.4406	23.97 / 0.8167 / 0.4337
Ours (1%)	28.02 / 0.9340 / 0.0847	23.75 / 0.7066 / 0.3367	32.41 / 0.9626 / 0.0696	21.08 / 0.7384 / 0.3114	24.88 / 0.8454 / 0.3550
3DGS* (2%)	28.44 / 0.9320 / 0.0973	23.43 / 0.6816 / 0.4216	31.90 / 0.9478 / 0.1317	21.05 / 0.7199 / 0.3881	25.49 / 0.8459 / 0.3894
Ours (2%)	29.68 / 0.9465 / 0.0716	24.79 / 0.7454 / 0.2945	32.77 / 0.9652 / 0.0621	22.14 / 0.7757 / 0.2726	26.29 / 0.8681 / 0.3186
3DGS* (5%)	30.09 / 0.9491 / 0.0742	24.88 / 0.7450 / 0.3393	32.63 / 0.9593 / 0.0987	22.25 / 0.7720 / 0.3189	26.55 / 0.8642 / 0.3469
Ours (5%)	30.96 / 0.9564 / 0.0571	25.87 / 0.7841 / 0.2492	32.96 / 0.9662 / 0.0584	23.05 / 0.8085 / 0.2339	27.00 / 0.8735 / 0.3039
3DGS* (10%)	31.47 / 0.9590 / 0.0594	25.77 / 0.7796 / 0.2860	32.71 / 0.9627 / 0.0811	22.82 / 0.8020 / 0.2728	27.64 / 0.8853 / 0.3101
Ours (10%)	32.12 / 0.9630 / 0.0488	26.45 / 0.8012 / 0.2298	32.72 / 0.9659 / 0.0583	23.42 / 0.8258 / 0.2123	27.98 / 0.8899 / 0.2803
3DGS* (20%)	32.21 / 0.9637 / 0.0506	26.45 / 0.8057 / 0.2408	32.93 / 0.9651 / 0.0699	23.58 / 0.8305 / 0.2298	27.99 / 0.8908 / 0.2927
Ours (20%)	32.59 / 0.9656 / 0.0453	26.83 / 0.8139 / 0.2138	32.85 / 0.9666 / 0.0584	23.99 / 0.8428 / 0.1940	28.16 / 0.8906 / 0.2784
3DGS* (50%)	32.77 / 0.9664 / 0.0452	27.02 / 0.8250 / 0.2019	33.23 / 0.9683 / 0.0603	23.96 / 0.8479 / 0.1922	28.11 / 0.8939 / 0.2762
Ours (50%)	32.98 / 0.9673 / 0.0434	27.13 / 0.8245 / 0.1973	33.19 / 0.9686 / 0.0594	24.18 / 0.8516 / 0.1801	28.27 / 0.8915 / 0.2738
3DGS* (100%)	33.09 / 0.9671 / 0.0440	27.28 / 0.8318 / 0.1871	33.54 / 0.9697 / 0.0551	24.18 / 0.8541 / 0.1754	28.04 / 0.8940 / 0.2707
Ours (100%)	33.24 / 0.9674 / 0.0428	27.35 / 0.8274 / 0.1858	33.61 / 0.9699 / 0.0556	24.26 / 0.8542 / 0.1684	28.33 / 0.8908 / 0.2699

Table A.3. **Quantitaive results on all datasets using varying numbers of Gaussians with a fixed amount of texels.** We report the quantitative NVS performance of our RGBA Textured Gaussians model and 3DGS* with the same number of Gaussians on all five test datasets in terms of PSNR (↑), SSIM (↑), and LPIPS (↓) metrics. Our Textured Gaussians model consistently outperforms 3DGS* when using different numbers of Gaussians.

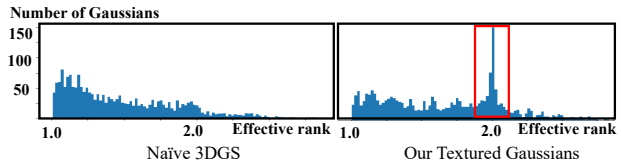


Figure A.2. We analyzed the effective rank distribution [26] of naïve Gaussian splatting and Textured Gaussians models. Our Textured Gaussians optimization naturally leads to a large proportion of flat 2D Gaussians (effective rank = 2), while naïve Gaussian splatting leads to more needle-like Gaussian strands (effective rank $\approx$ 1). This observation justifies our choice of using the 3DGS framework to approximate 2D Gaussians.

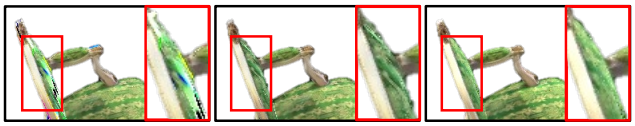


Figure A.3. We designed various heuristics to reduce artifacts caused by rendering flat 2D Gaussians from grazing viewpoints, including intersection clipping (left) and adding SH base colors (middle). Our full model achieves the best rendering results.

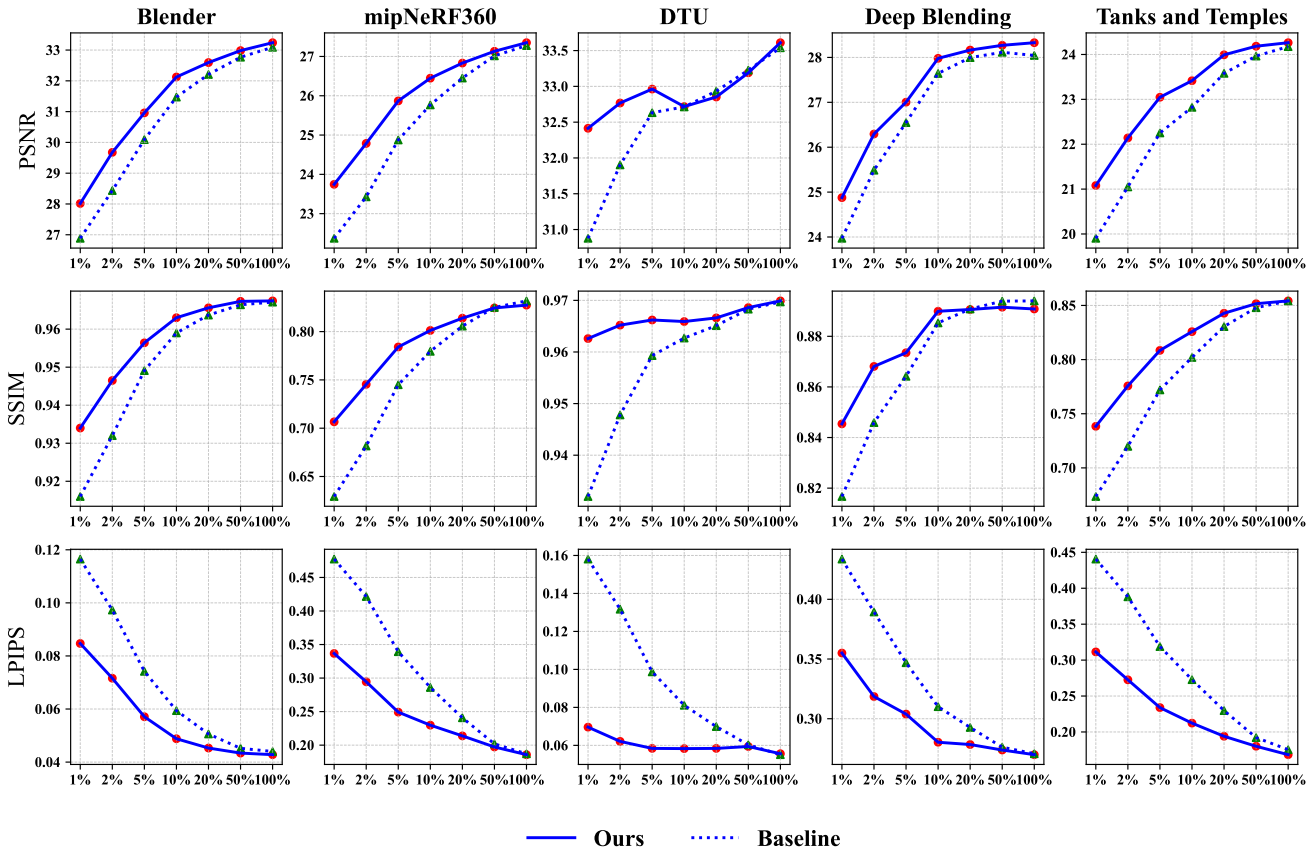


Figure A.4. **Quantitaive results on all datasets using varying numbers of Gaussians with a fixed amount of texels.** We show the NVS performance trend of our RGBA Textured Gaussians model and 3DGS* as the number of Gaussians increases. Our Textured Gaussians model greatly outperforms 3DGS* with fewer Gaussians, and the performance gap between the two methods decreases as the number of Gaussians increases.

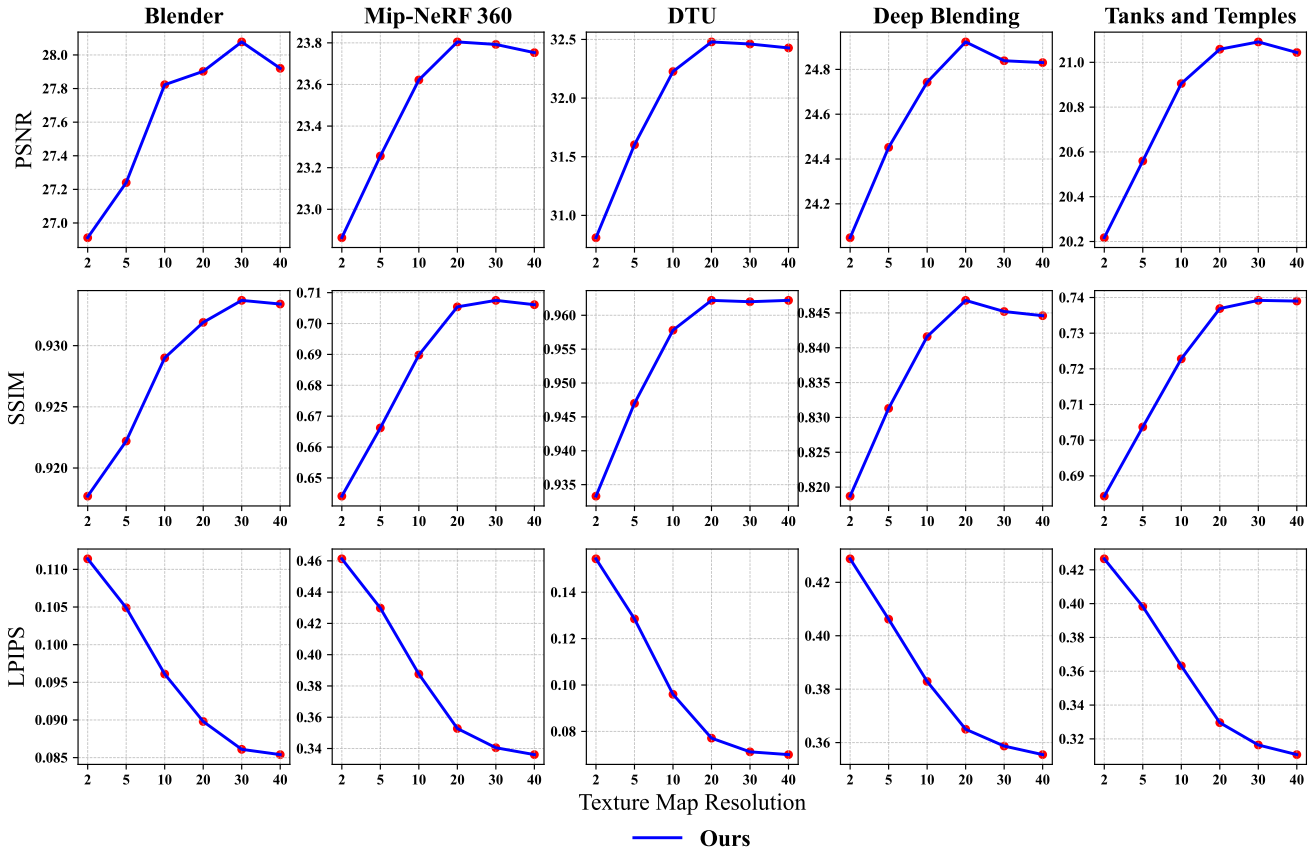


Figure A.5. **Quantitative results of ablation study on varying texture resolutions with a fixed number of Gaussians.** As the texture map resolution increases, novel-view synthesis quality improves as detailed appearances are reconstructed better. However, the performance drops slightly when the texture map resolution is too high, likely due to overfitting.

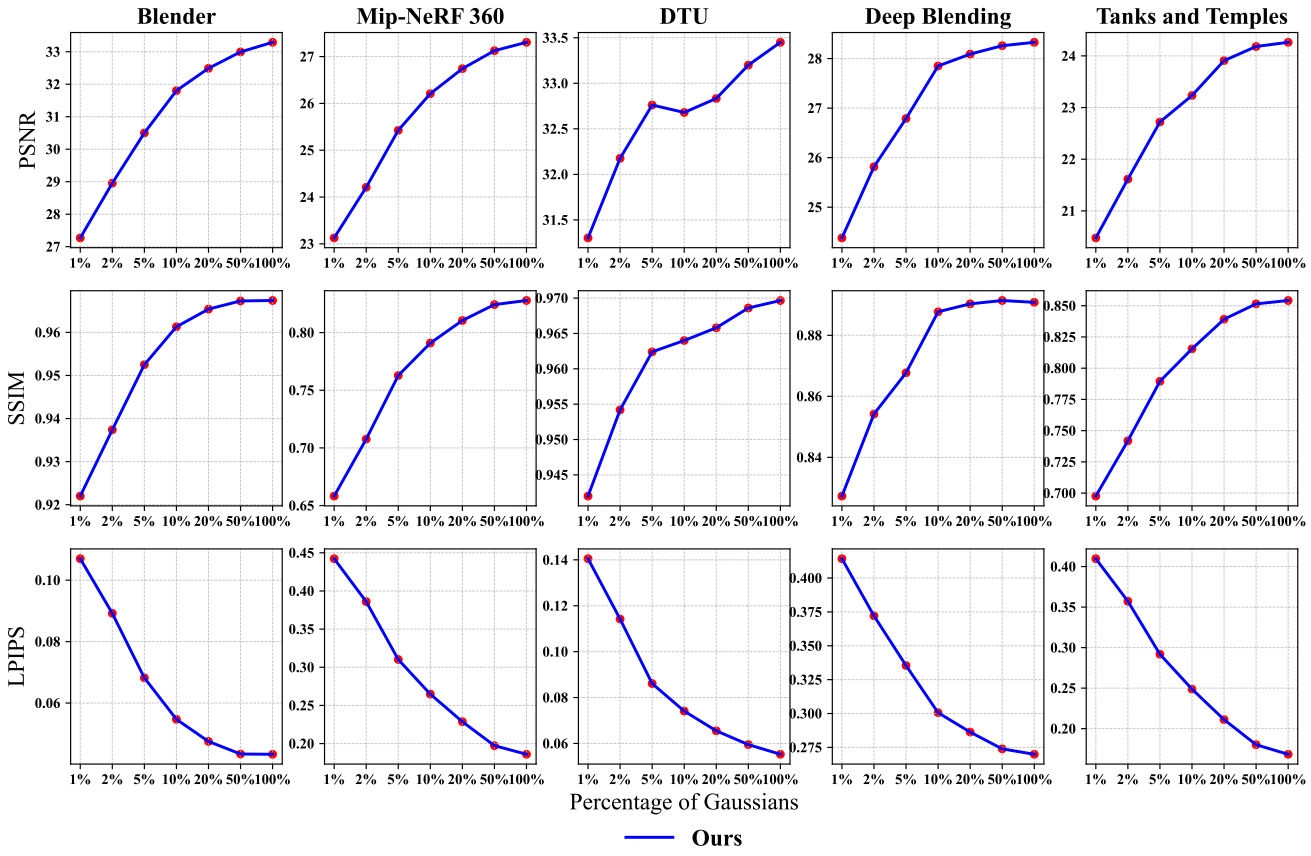


Figure A.6. **Quantitative results of ablation study on varying number of Gaussians and with a fixed texture map resolution.** As the number of Gaussians increase, novel-view synthesis performance improves since detailed appearance and geometry can be reconstructed better using smaller and more Gaussians.

Method	Blender [37]	Mip-NeRF 360 [37]	DTU [27]	Tanks and Temples [30]	Deep Blending [20]
3DGS*	33.09 / 0.9671 / 0.0440	27.28 / 0.8318 / 0.1871	33.54 / 0.9697 / 0.0551	24.18 / 0.8541 / 0.1754	28.04 / 0.8940 / 0.2707
Alpha-only	33.22 / 0.9672 / 0.0433	27.32 / 0.8259 / 0.1874	33.51 / 0.9692 / 0.0554	28.30 / 0.8888 / 0.2720	24.27 / 0.8532 / 0.1695
RGB	33.20 / 0.9673 / 0.0431	27.30 / 0.8268 / 0.1873	33.58 / 0.9697 / 0.0561	28.21 / 0.8895 / 0.2722	24.24 / 0.8536 / 0.1695
RGBA	33.24 / 0.9674 / 0.0429	27.35 / 0.8274 / 0.1859	33.60 / 0.9699 / 0.0559	28.34 / 0.8910 / 0.2699	24.26 / 0.8542 / 0.1685
3DGS* (1%)	26.89 / 0.9160 / 0.1165	22.37 / 0.6293 / 0.4774	30.88 / 0.9320 / 0.1581	19.90 / 0.6736 / 0.4406	23.97 / 0.8167 / 0.4337
Alpha-only (1%)	27.64 / 0.9304 / 0.0905	23.69 / 0.7012 / 0.3494	32.31 / 0.9604 / 0.0759	24.68 / 0.8411 / 0.3620	20.93 / 0.7335 / 0.3226
RGB (1%)	27.60 / 0.9279 / 0.0923	23.50 / 0.6971 / 0.3558	32.52 / 0.9612 / 0.0765	24.64 / 0.8403 / 0.3682	20.73 / 0.7236 / 0.3352
RGBA (1%)	28.11 / 0.9343 / 0.0849	23.73 / 0.7064 / 0.3365	32.43 / 0.9627 / 0.0694	24.83 / 0.8454 / 0.3552	21.08 / 0.7395 / 0.3107

Table A.4. **Quantitative results of ablation study on texture map variants.** We ablate different variants of our model that use alpha, RGB, and RGBA texture maps with the default optimized number and 1% of the default optimized number of Gaussians. We report the NVS performance of different models in terms of PSNR (↑), SSIM (↑), and LPIPS (↓) metrics. Models with RGBA textures achieve the best results due to the maximum expressivity of individual Gaussians. Interestingly, models with alpha-only textures achieve better results than RGB textures models while using only one-third of the model size.

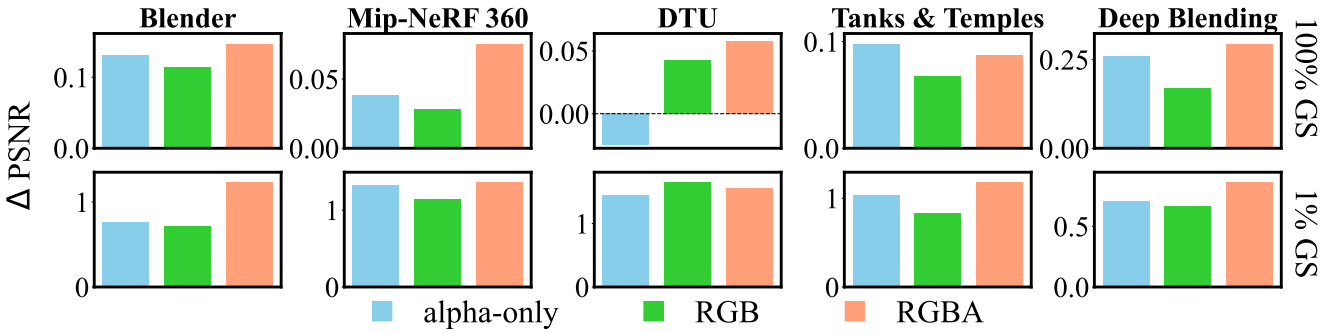


Figure A.7. **Quantitative results of ablation study on texture map variants.** We optimized our Textured Guasians model with different texture map variants (alpha-only, RGB, RGBA). We show the difference in PSNR values compared to the 3DGS* baseline for better visual comparison. We observe that the alpha-only textures models generally achieve better performance than RGB textures models. This is because alpha textures are capable of representing complex appearance and shapes due to spatially varying alpha modulation and composition. On the other hand, RGB-textured Gaussians are still only capable of representing ellipsoids. The full RGBA textures models achieve the best results since the RGBA-textured Gaussians are most expressive.

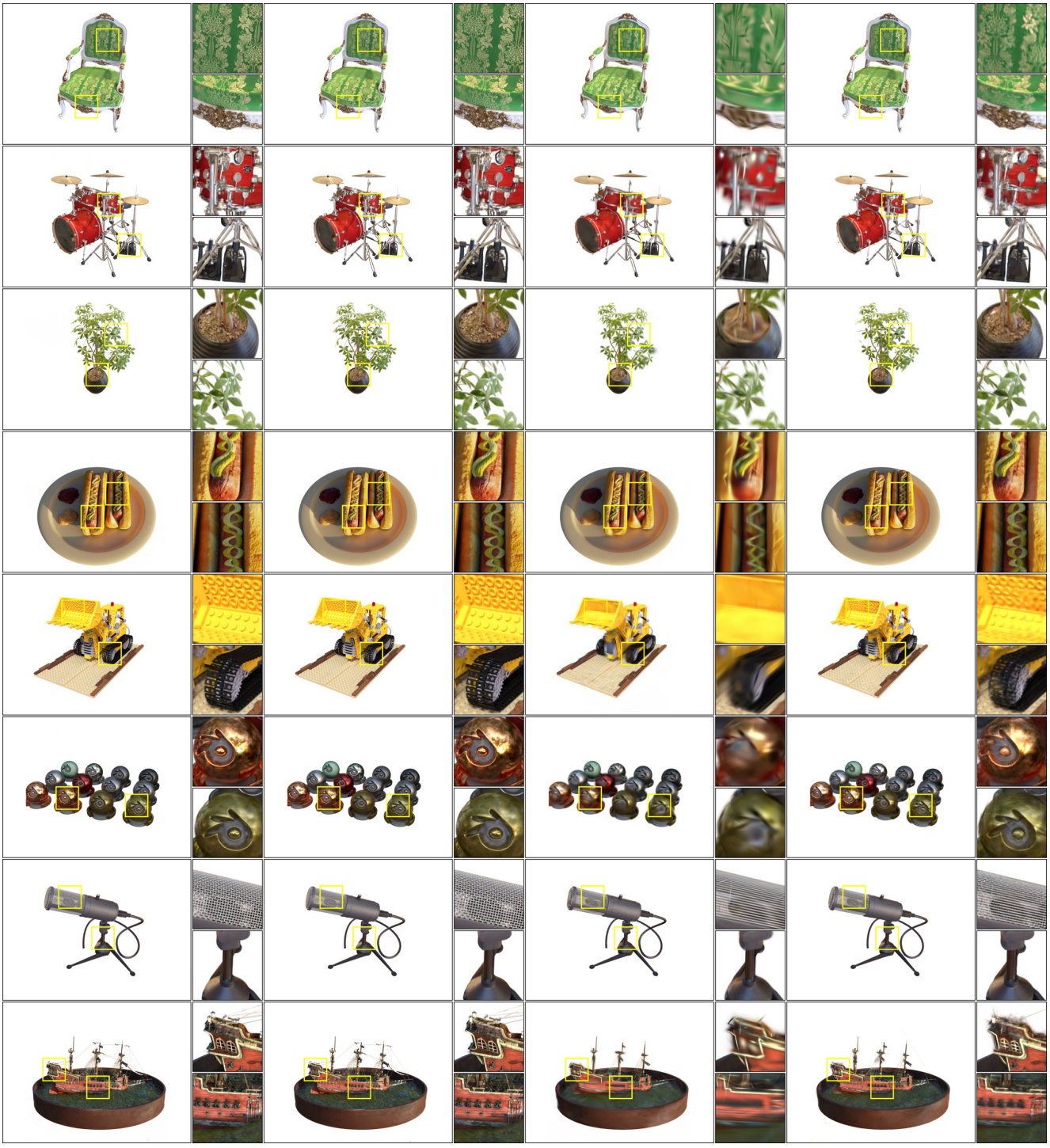


Figure A.8. **Qualitative NVS results on the Blender dataset with varying numbers of Gaussians and a fixed amount of texels.** Our RGBA Textured Gaussians model achieves better NVS quality compared to 3DGS* when using the same number of Gaussians. The visual difference is especially clear with fewer Gaussians.

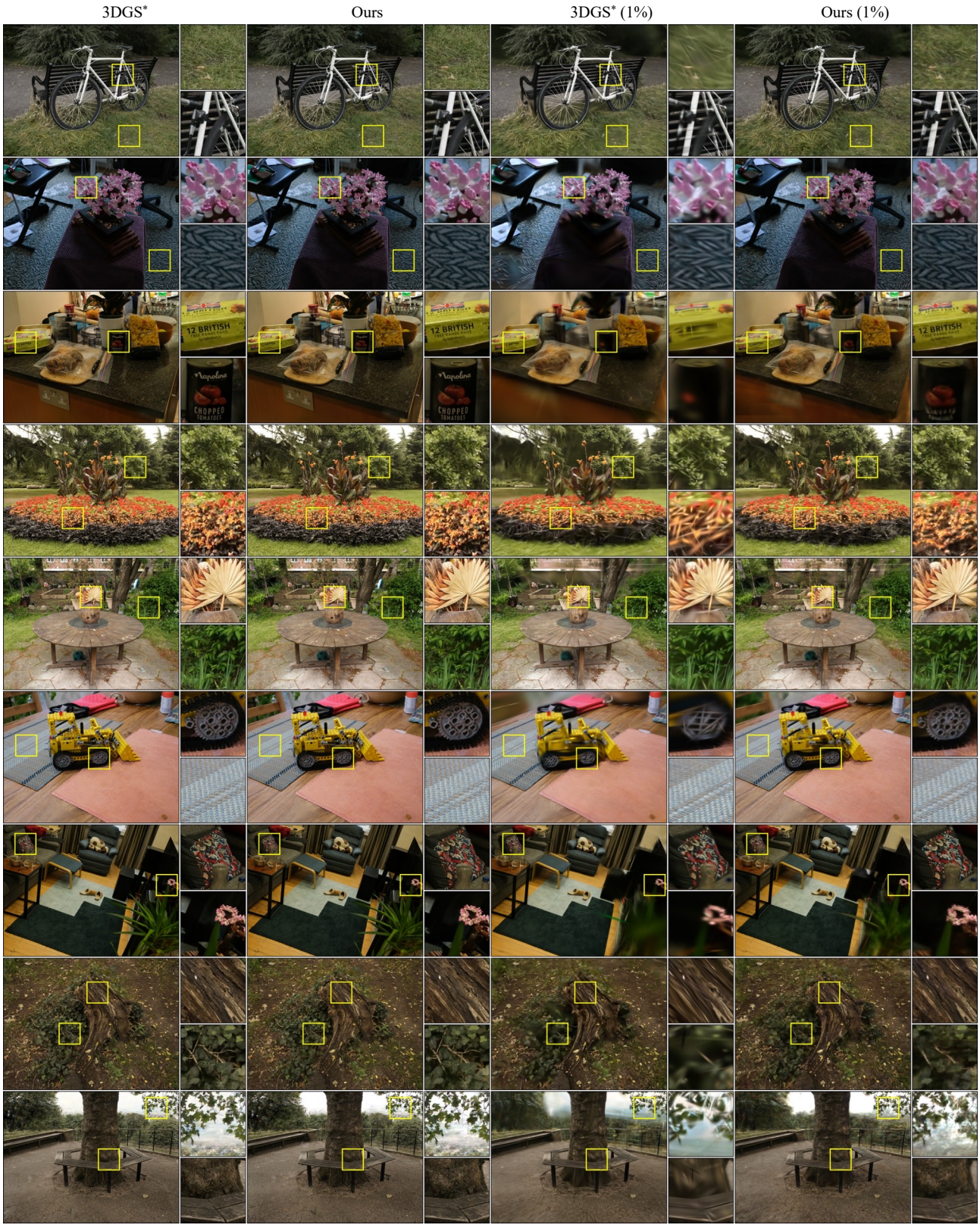


Figure A.9. **Qualitative NVS results on the Mip-NeRF 360 dataset with varying numbers of Gaussians and a fixed amount of texels.** Our RGBA Textured Gaussians model achieves better NVS quality compared to 3DGS* when using the same number of Gaussians. The visual difference is especially clear with fewer Gaussians.

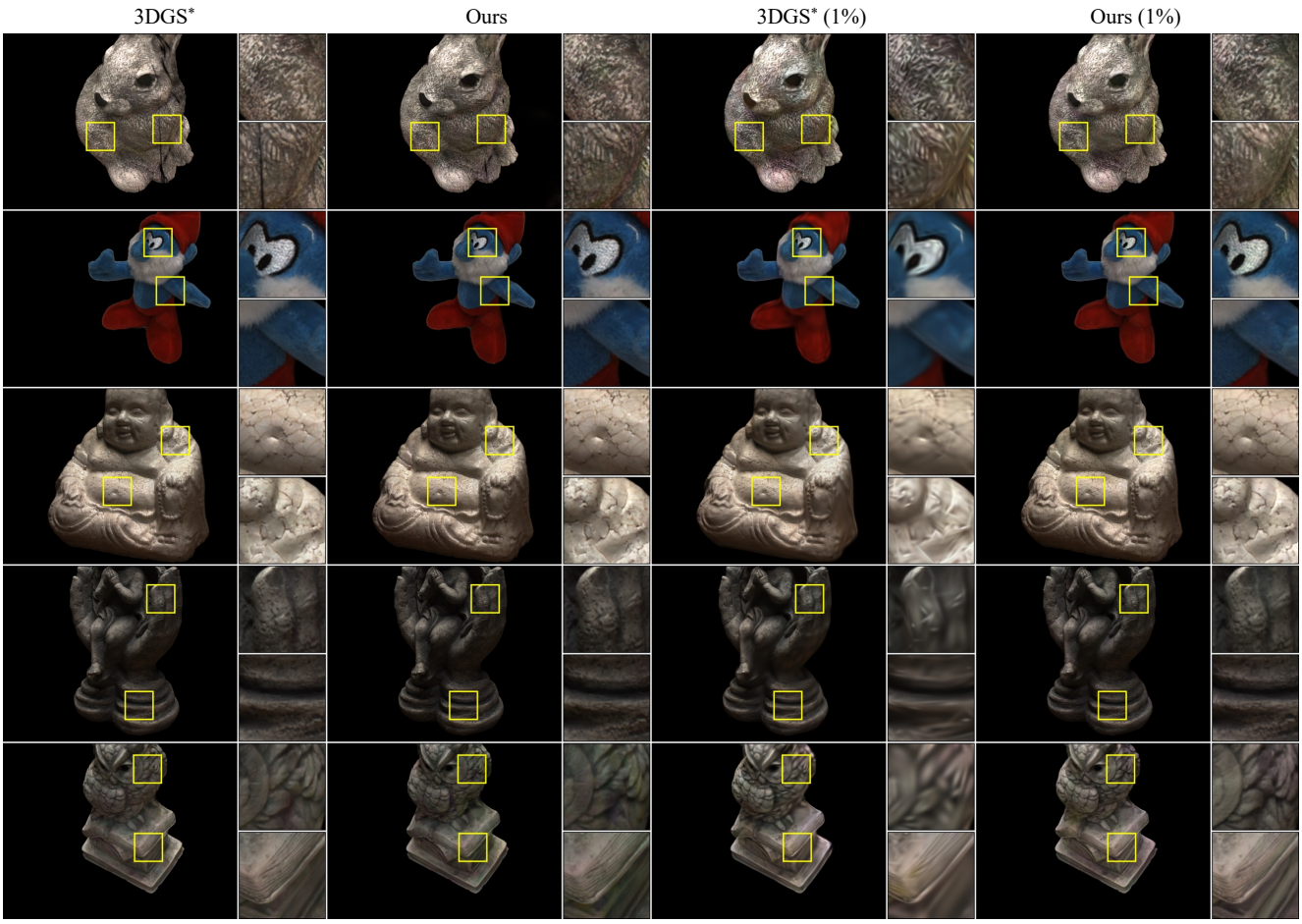


Figure A.10. **Qualitative NVS results on the DTU dataset with varying numbers of Gaussians and a fixed amount of texels.** Our RGBA Textured Gaussians model achieves better NVS quality compared to 3DGS* when using the same number of Gaussians. The visual difference is especially clear with fewer Gaussians.

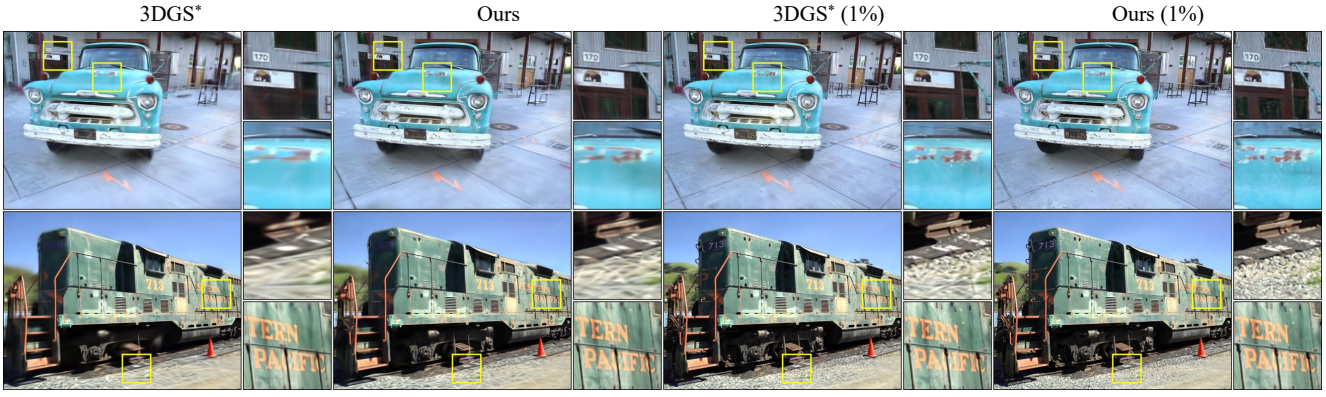


Figure A.11. **Qualitative NVS results on the Tanks and Temples dataset with varying numbers of Gaussians and a fixed amount of texels.** Our RGBA Textured Gaussians model achieves better NVS quality compared to 3DGS* when using the same number of Gaussians. The visual difference is especially clear with fewer Gaussians.

Figure A.12. **Qualitative NVS results on the Deep Blending dataset with varying numbers of Gaussians and a fixed amount of texels.** Our RGBA Textured Gaussians model achieves better NVS quality compared to 3DGS* when using the same number of Gaussians. The visual difference is especially clear with fewer Gaussians.

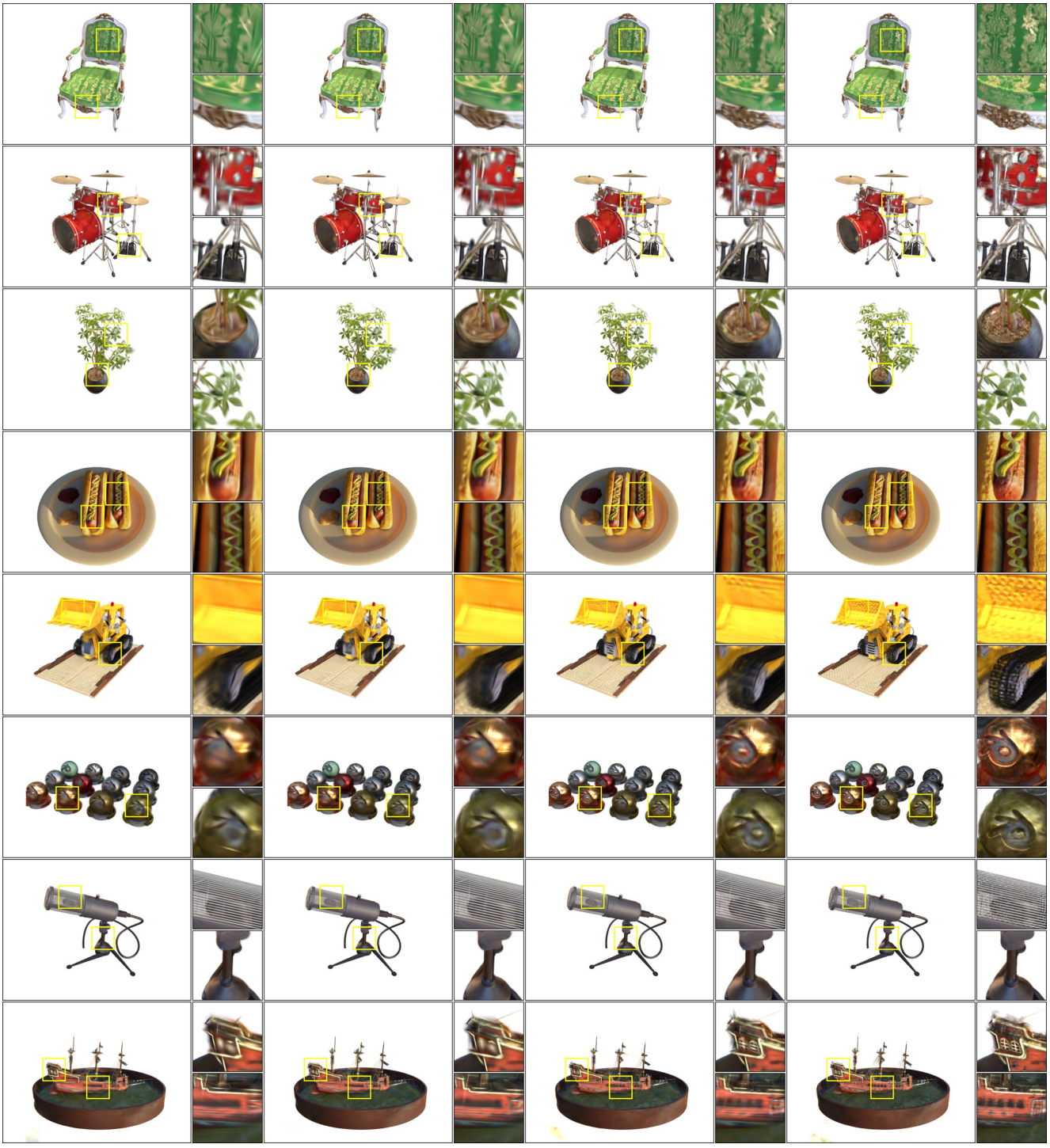


Figure A.13. **Qualitative NVS results on the Blender dataset with vayring texture map resolutions and a fixed number of Gaussians.** Given a fixed number of Gaussians, novel-view synthesis performance improves as the texture map resolution increases. Detailed appearance can be reconstructed better with higher-resolution texture maps, or smaller texel feature sizes.

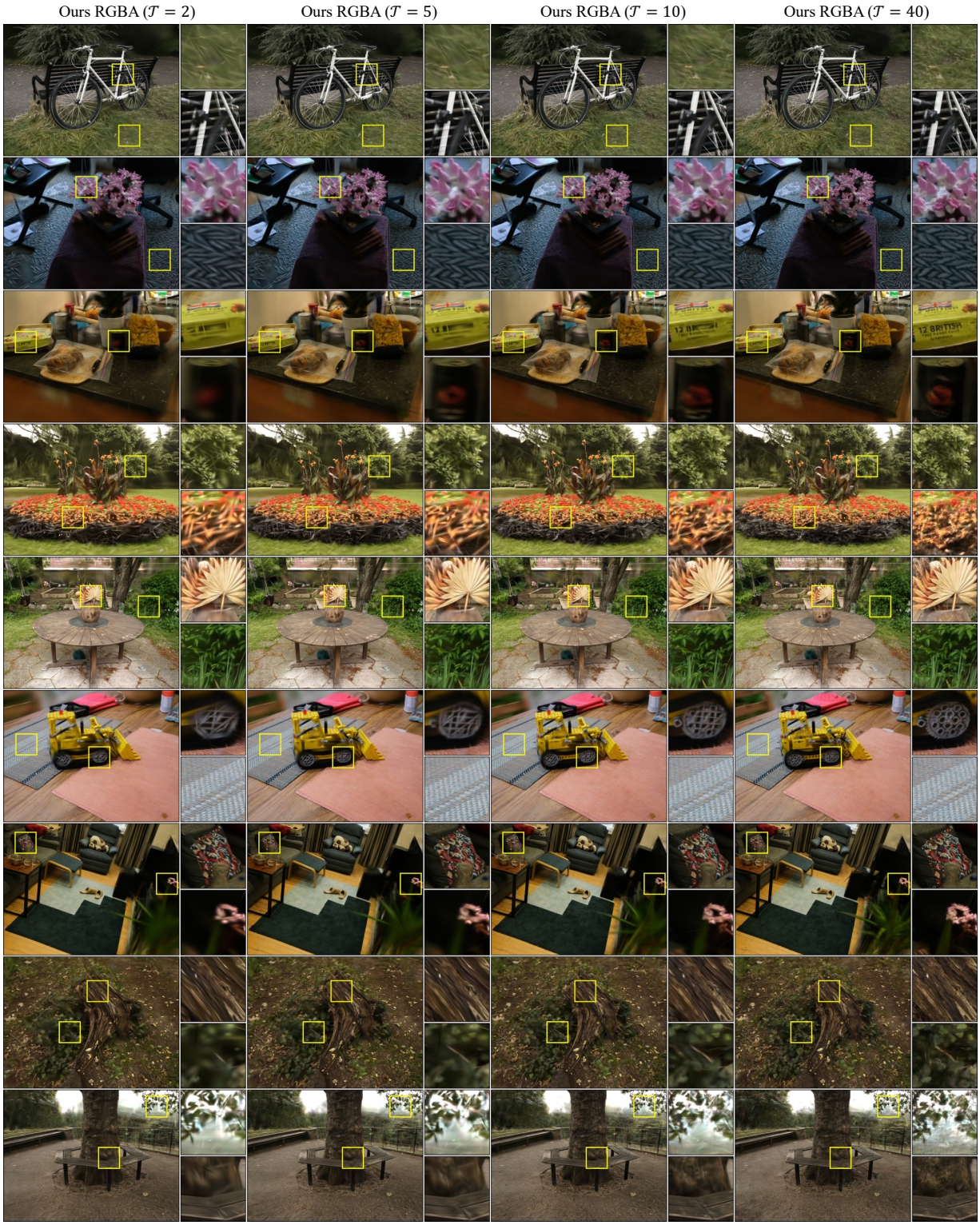


Figure A.14. **Qualitative NVS results on the Mip-NeRF 360 dataset with vayring texture map resolutions and a fixed number of Gaussians.** Given a fixed number of Gaussians, novel-view synthesis performance improves as the texture map resolution increases. Detailed appearance can be reconstructed better with higher-resolution texture maps, or smaller texel feature sizes.

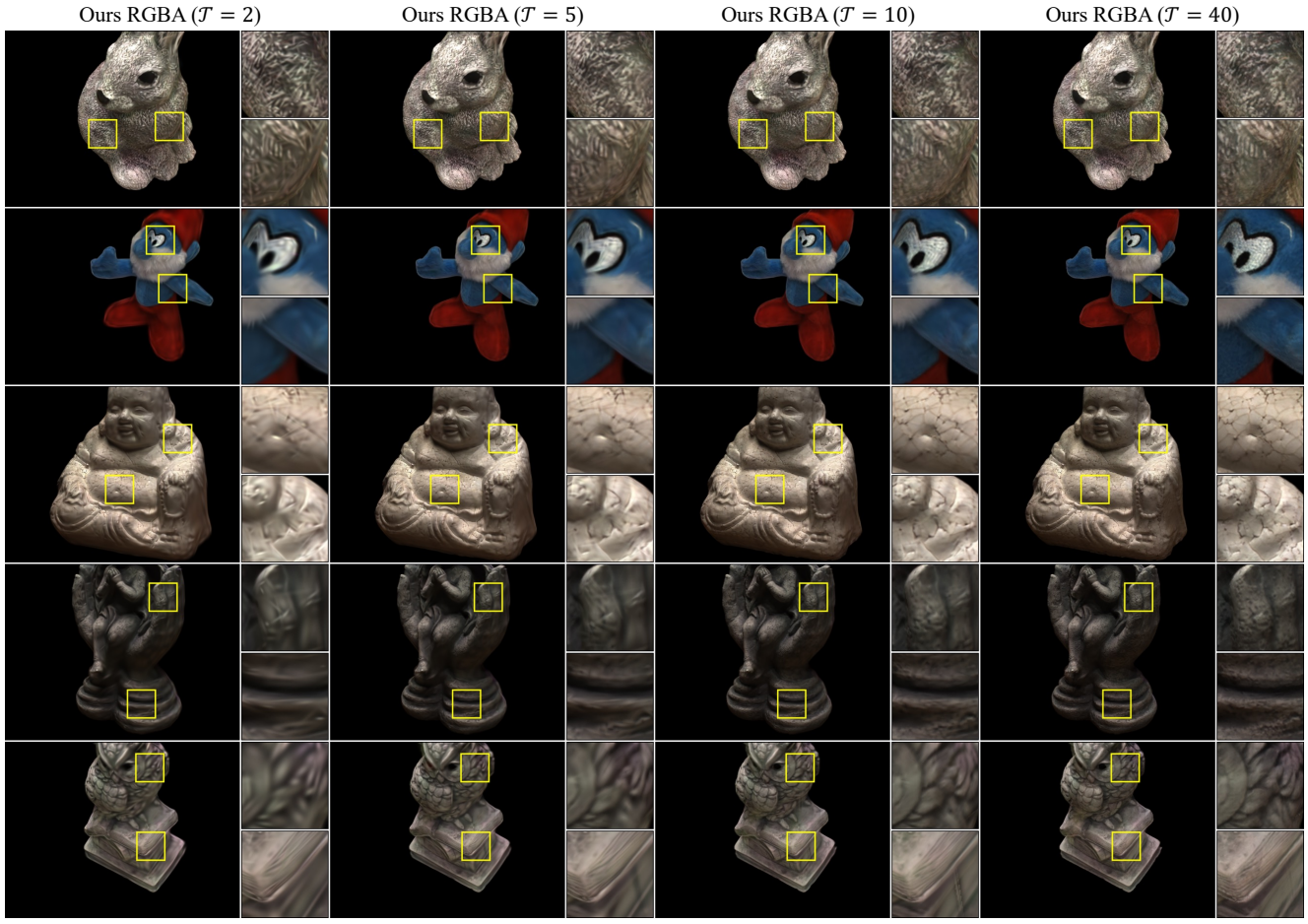


Figure A.15. **Qualitative NVS results on the DTU dataset with vayring texture map resolutions and a fixed number of Gaussians.** Given a fixed number of Gaussians, novel-view synthesis performance improves as the texture map resolution increases. Detailed appearance can be reconstructed better with higher-resolution texture maps, or smaller texel feature sizes.

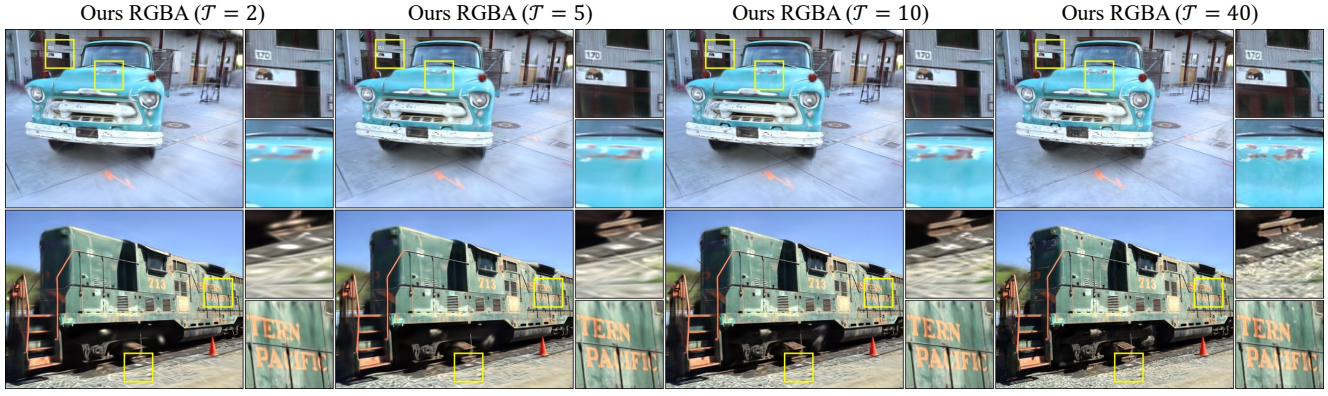


Figure A.16. **Qualitative NVS results on the Tanks and Temples dataset with vayring texture map resolutions and a fixed number of Gaussians.** Given a fixed number of Gaussians, novel-view synthesis performance improves as the texture map resolution increases. Detailed appearance can be reconstructed better with higher-resolution texture maps, or smaller texel feature sizes.

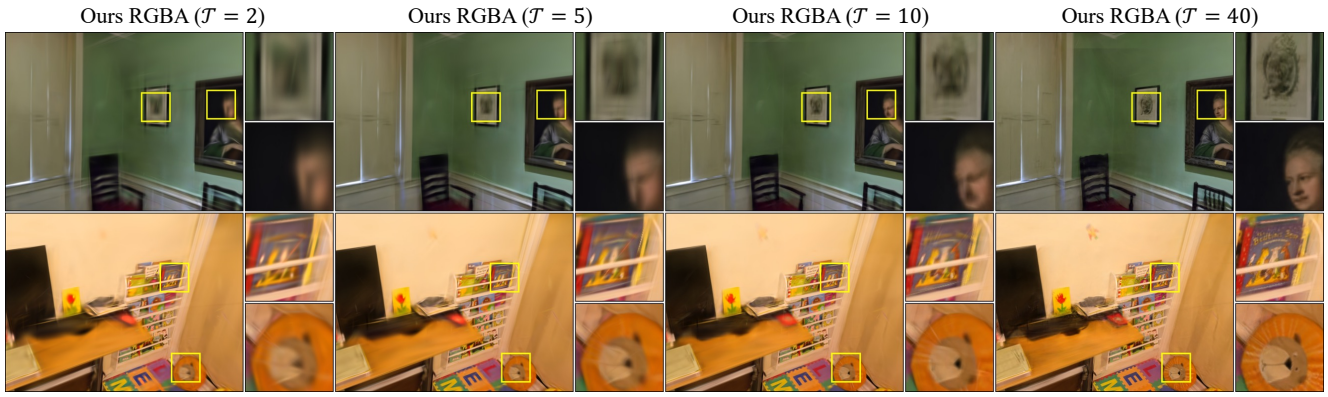


Figure A.17. **Qualitative NVS results on the Deep Blending dataset with vayring texture map resolutions and a fixed number of Gaussians.** Given a fixed number of Gaussians, novel-view synthesis performance improves as the texture map resolution increases. Detailed appearance can be reconstructed better with higher-resolution texture maps, or smaller texel feature sizes.

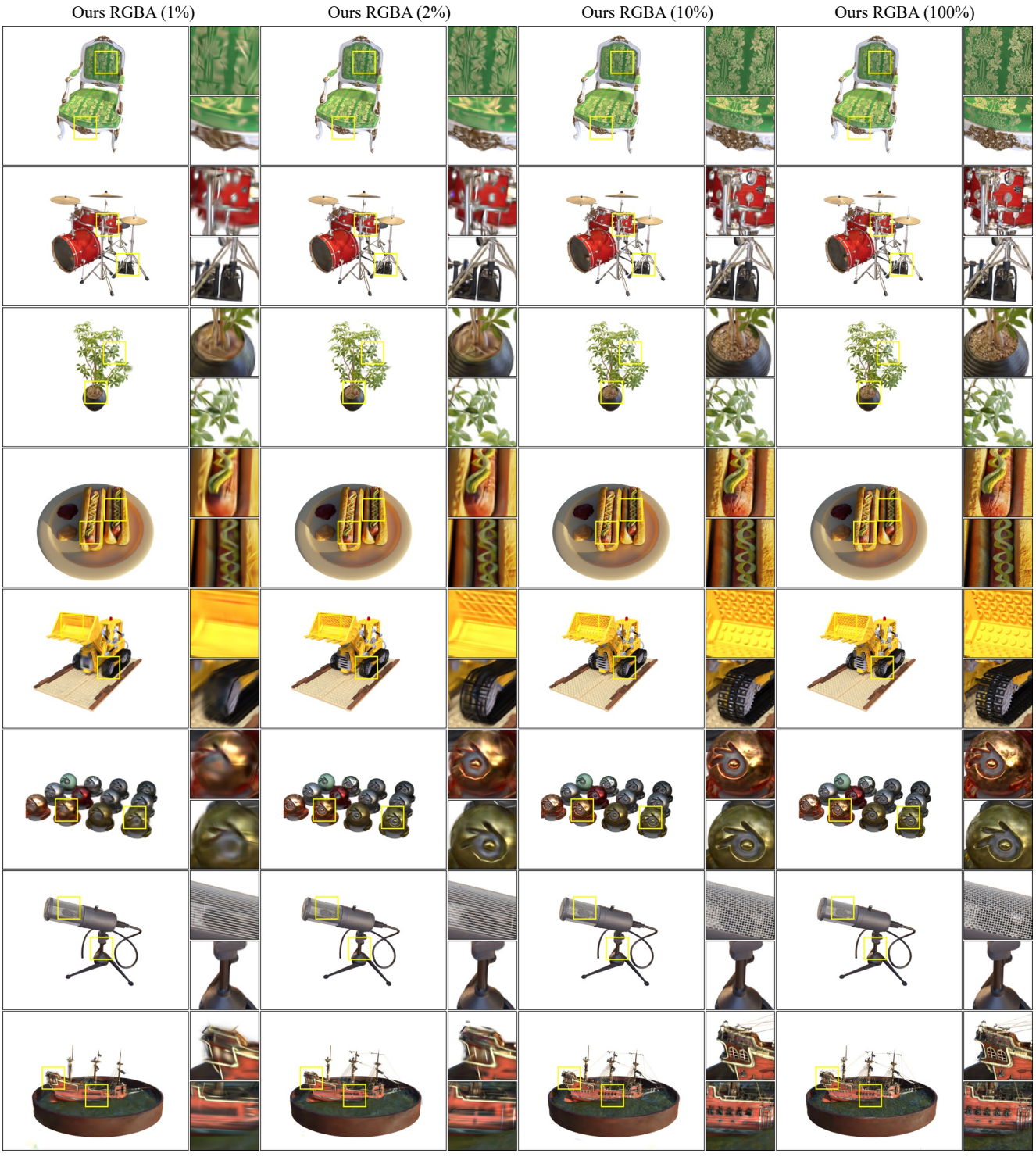


Figure A.18. **Qualitative NVS results on the Blender dataset with vayring numbers of Gaussians and a fixed texture map resolution.** Given a fixed texture map resolution, novel-view synthesis performance improves as the number of Gaussians increases. Detailed appearance and complex geometry can be reconstructed with more and therefore smaller Gaussians.

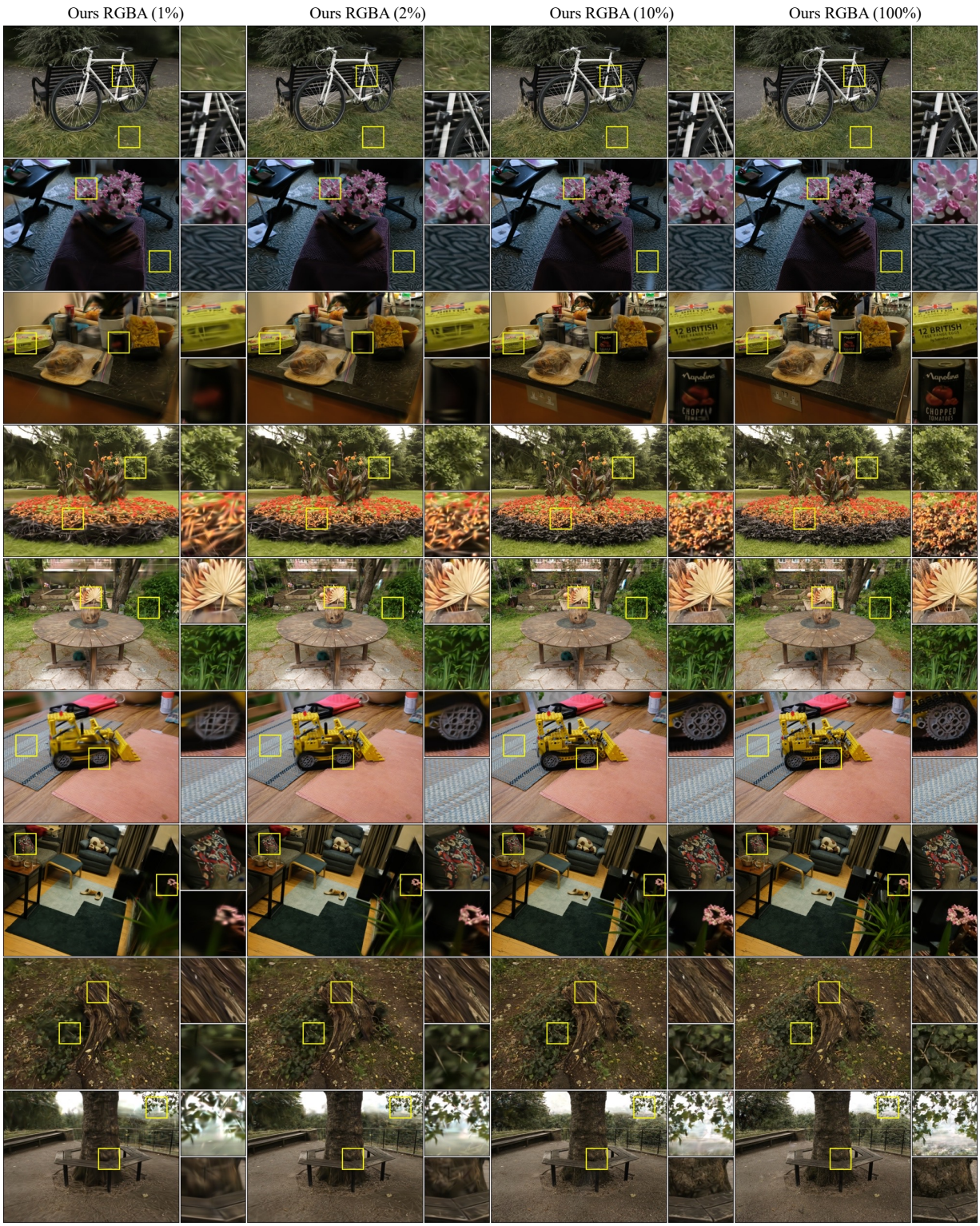


Figure A.19. **Qualitative NVS results on the Mip-NeRF 360 dataset with vayring numbers of Gaussians and a fixed texture map resolution.** Given a fixed texture map resolution, novel-view synthesis performance improves as the number of Gaussians increases. Detailed appearance and complex geometry can be reconstructed with more and therefore smaller Gaussians.

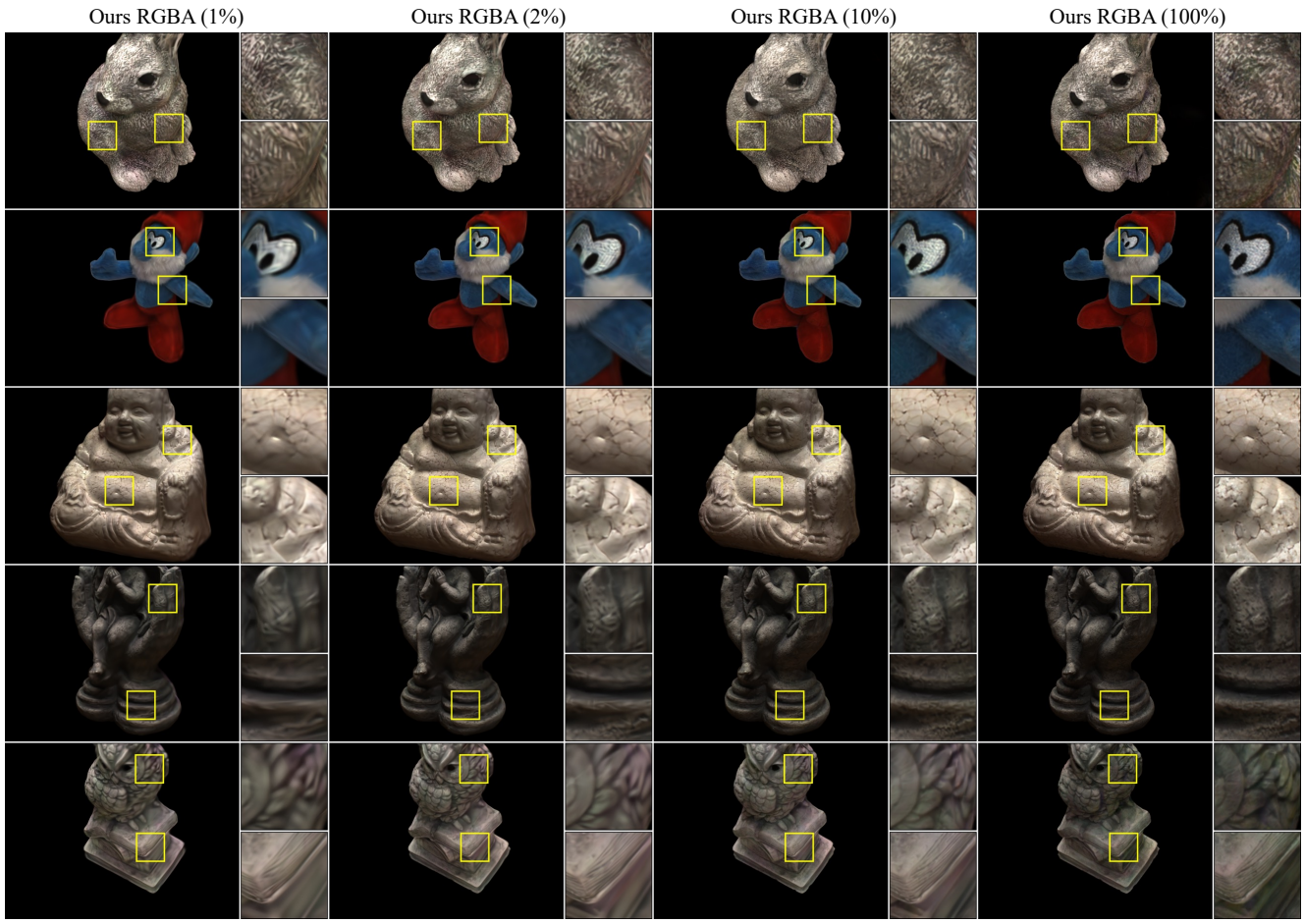


Figure A.20. **Qualitative NVS results on the DTU dataset with vayring numbers of Gaussians and a fixed texture map resolution.** Given a fixed texture map resolution, novel-view synthesis performance improves as the number of Gaussians increases. Detailed appearance and complex geometry can be reconstructed with more and therefore smaller Gaussians.

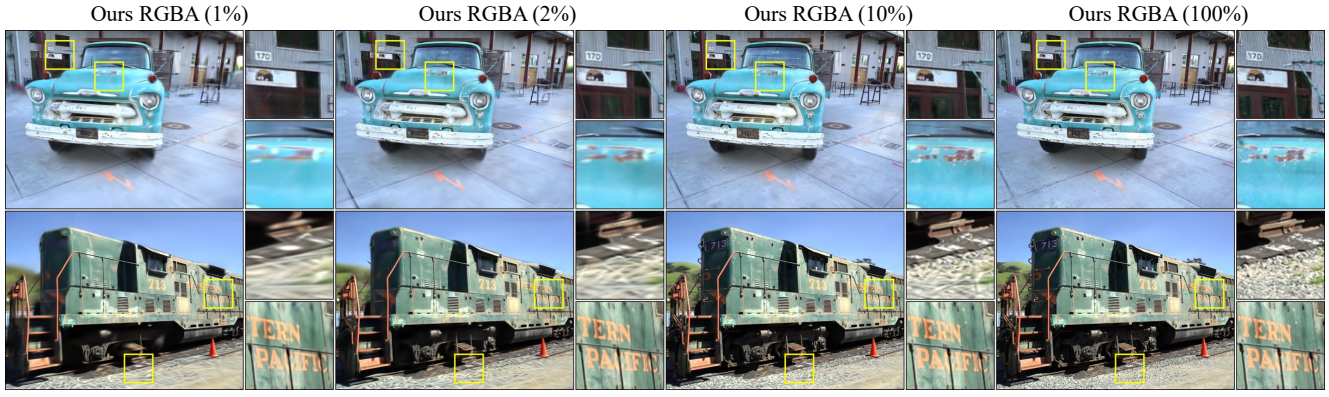


Figure A.21. **Qualitative NVS results on the Tanks and Temples dataset with vayring numbers of Gaussians and a fixed texture map resolution.** Given a fixed texture map resolution, novel-view synthesis performance improves as the number of Gaussians increases. Detailed appearance and complex geometry can be reconstructed with more and therefore smaller Gaussians.

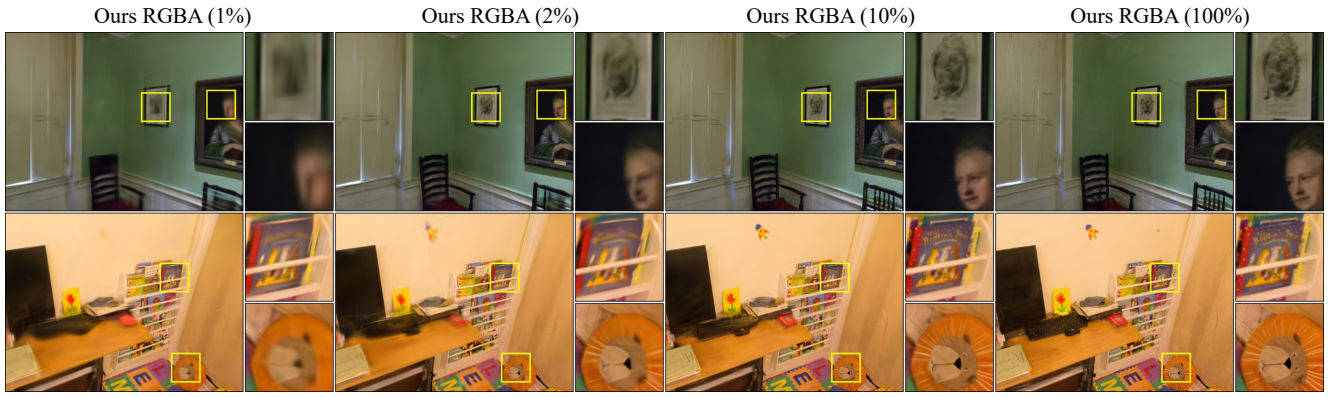


Figure A.22. **Qualitative NVS results on the Deep Blending dataset with vayring numbers of Gaussians and a fixed texture map resolution.** Given a fixed texture map resolution, novel-view synthesis performance improves as the number of Gaussians increases. Detailed appearance and complex geometry can be reconstructed with more and therefore smaller Gaussians.

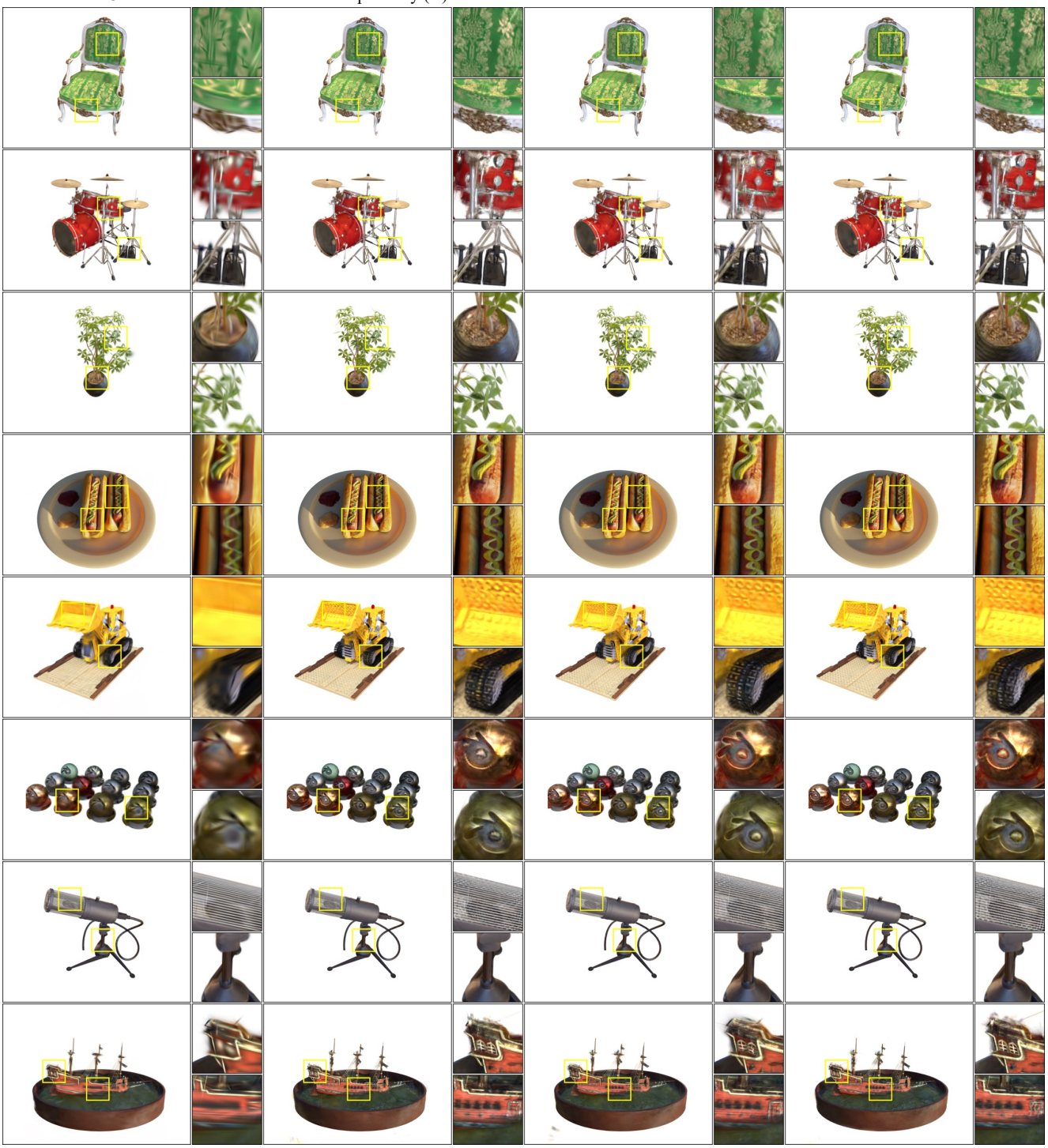


Figure A.23. **Texture map ablation results on the Blender dataset.** We see that models with alpha textures (alpha-only and RGBA models) achieves better NVS quality since *both* appearance and geometry can be reconstructed better due to spatially varying alpha modulation and composition. On the other hand, RGB Textured Gaussians models achieve worse visual quality since each Gaussian can still only represent ellipsoids.

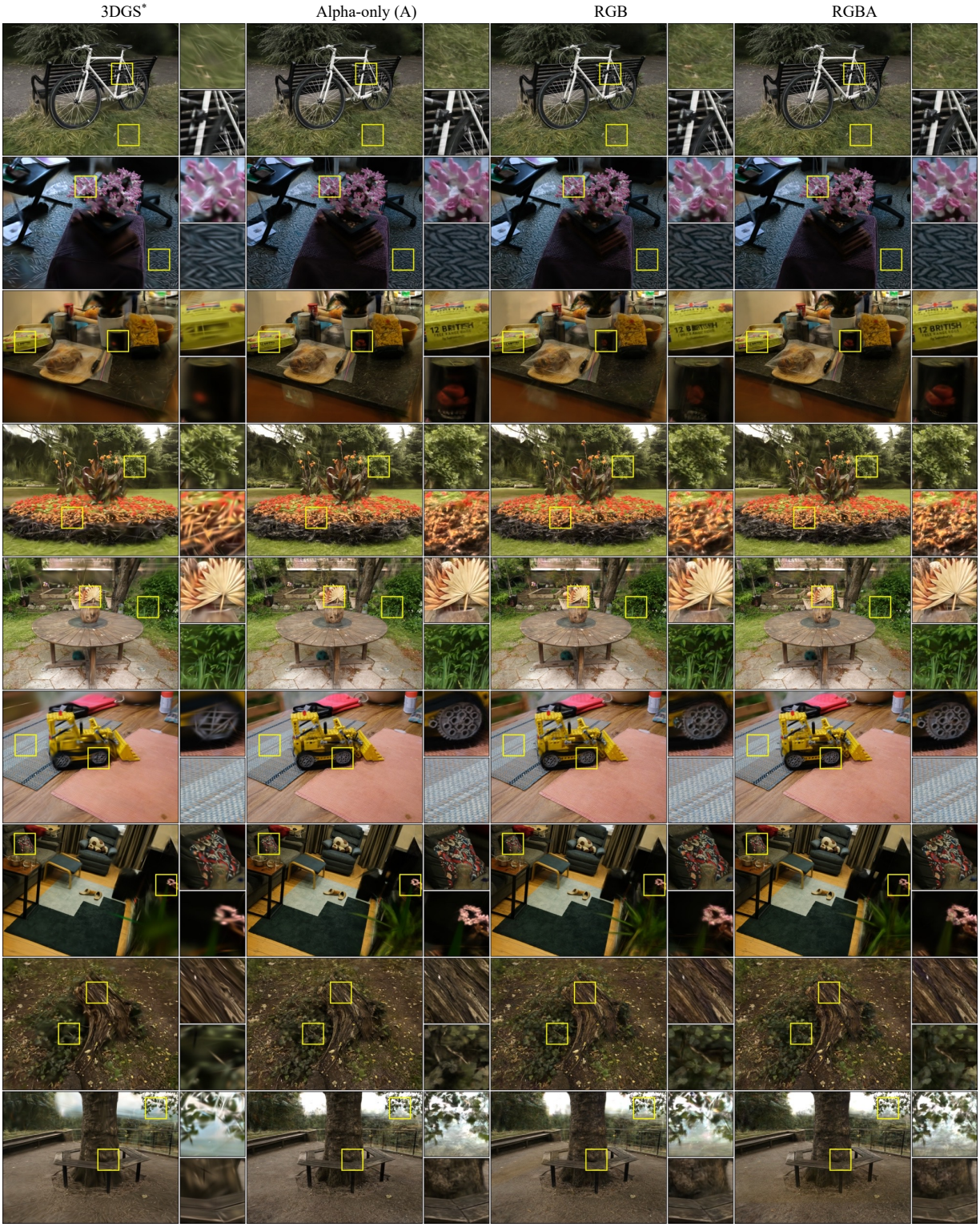


Figure A.24. **Texture map ablation results on the Mip-NeRF 360 dataset.** We see that models with alpha textures (alpha-only and RGBA models) achieves better NVS quality since *both* appearance and geometry can be reconstructed better due to spatially varying alpha modulation and composition. On the other hand, RGB Textured Gaussians models achieve worse visual quality since each Gaussian can still only represent ellipsoids.

Figure A.25. **Texture map ablation results on the DTU dataset.** We see that models with alpha textures (alpha-only and RGBA models) achieves better NVS quality since *both* appearance and geometry can be reconstructed better due to spatially varying alpha modulation and composition. On the other hand, RGB Textured Gaussians models achieve worse visual quality since each Gaussian can still only represent ellipsoids.

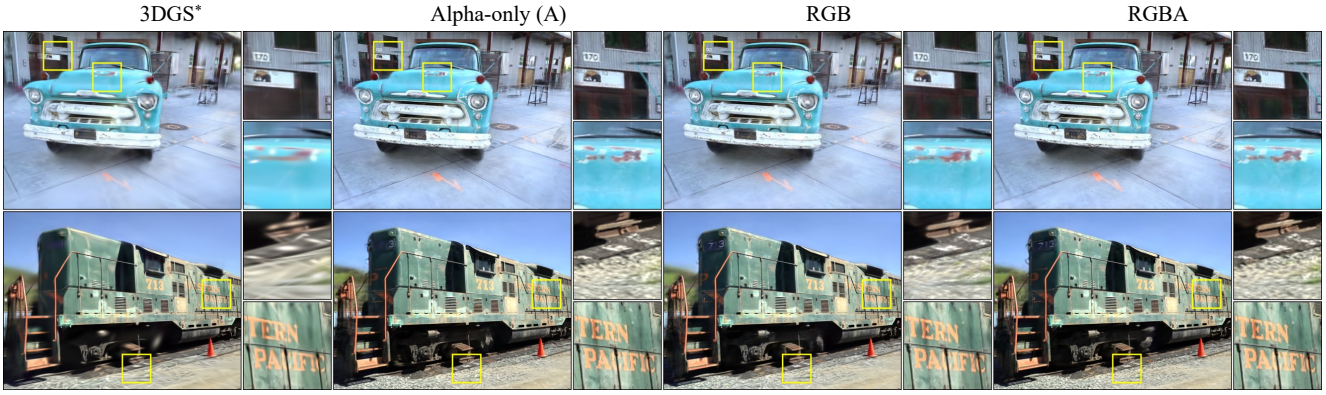


Figure A.26. **Texture map ablation results on the Tanks and Temples dataset.** We see that models with alpha textures (alpha-only and RGBA models) achieves better NVS quality since *both* appearance and geometry can be reconstructed better due to spatially varying alpha modulation and composition. On the other hand, RGB Textured Gaussians models achieve worse visual quality since each Gaussian can still only represent ellipsoids.

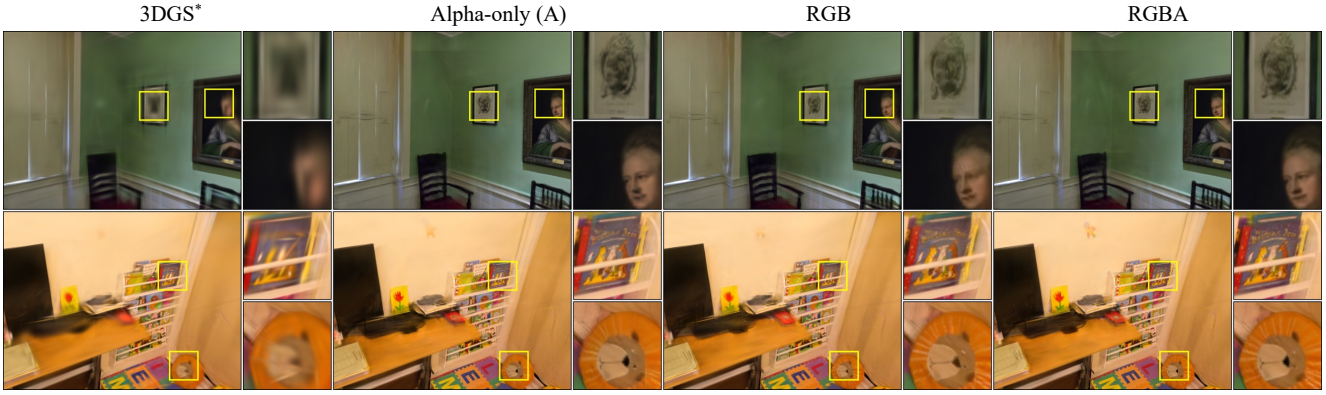


Figure A.27. **Texture map ablation results on the Deep Blending dataset.** We see that models with alpha textures (alpha-only and RGBA models) achieves better NVS quality since *both* appearance and geometry can be reconstructed better due to spatially varying alpha modulation and composition. On the other hand, RGB Textured Gaussians models achieve worse visual quality since each Gaussian can still only represent ellipsoids.

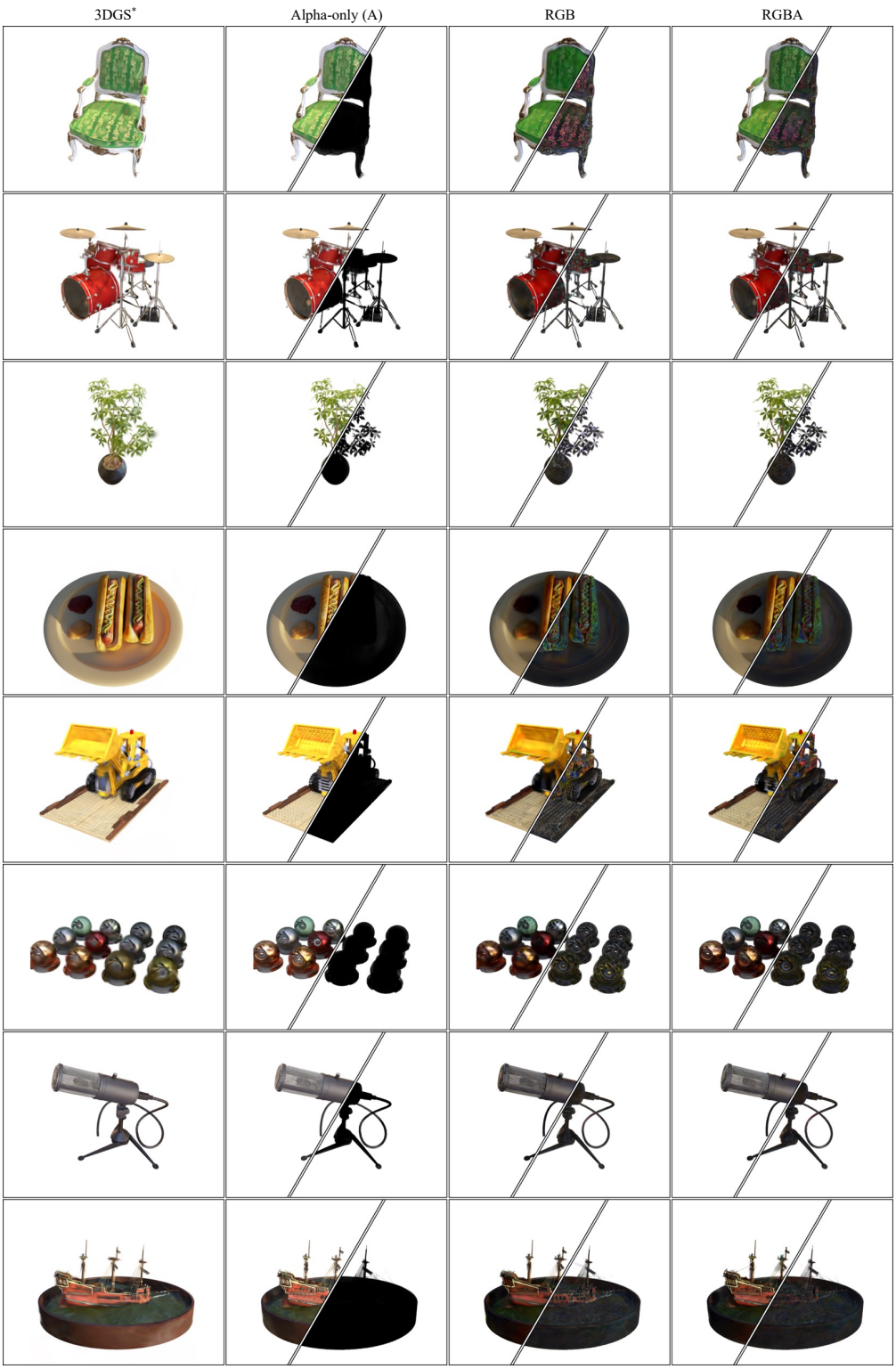


Figure A.28. **Color component decomposition visualization of the Blender dataset.** With alpha textures, the alpha-modulated and composited base color component can already recosntruct high-frequency textures, as shown in results of the alpha-only and RGBA Texuted Gaussians models. For models with only RGB textures, the base color component reconstructs lower frequency colors while the texture map color component reconstructs high frequency appearance.

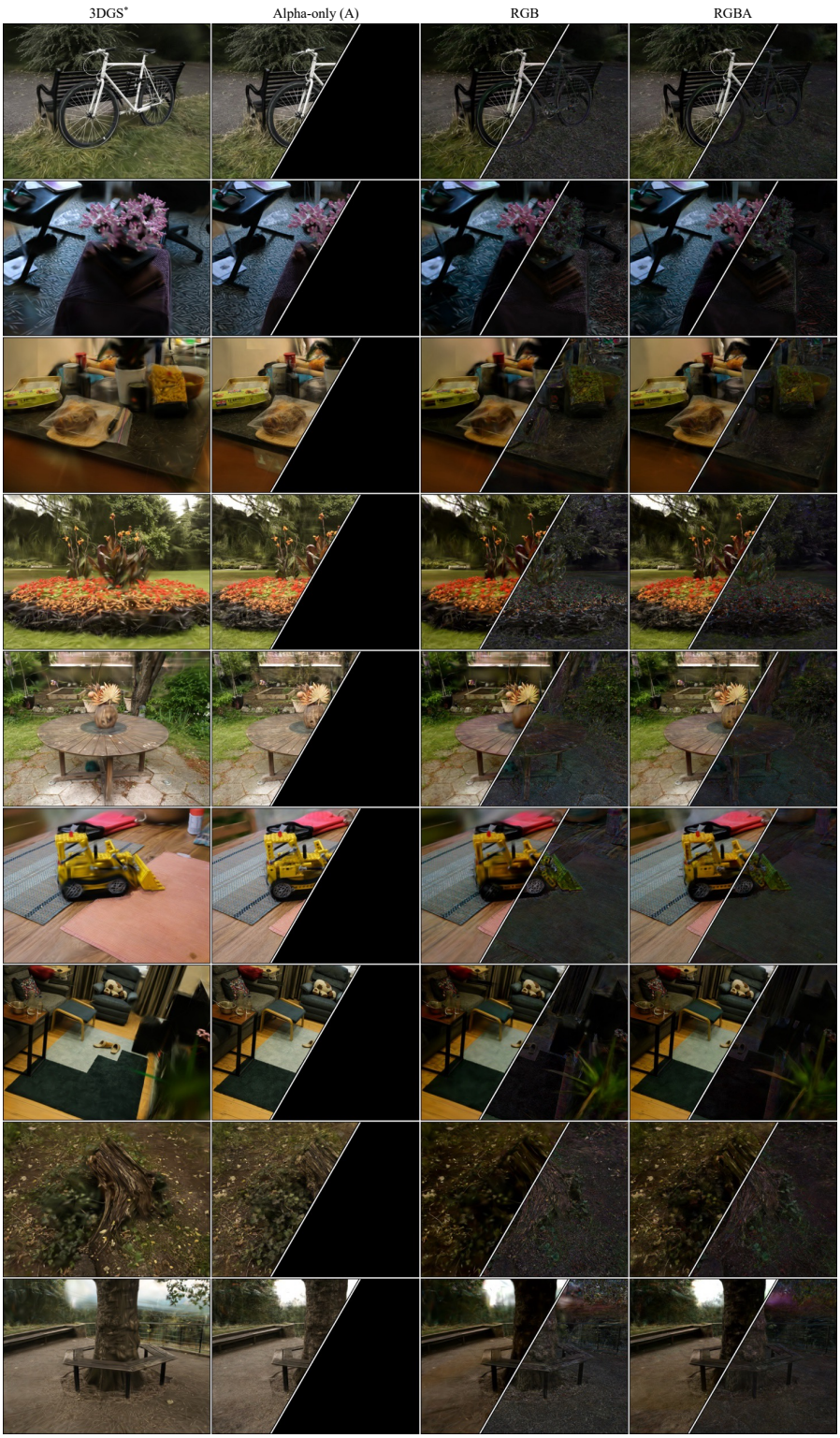


Figure A.29. **Color component decomposition visualization of the Mip-NeRF 360 dataset.** With alpha textures, the alpha-modulated and composited base color component can already recosntruct high-frequency textures, as shown in results of the alpha-only and RGBA Texuted Gaussians models. For models with only RGB textures, the base color component reconstructs lower frequency colors while the texture map color component reconstructs high frequency appearance.

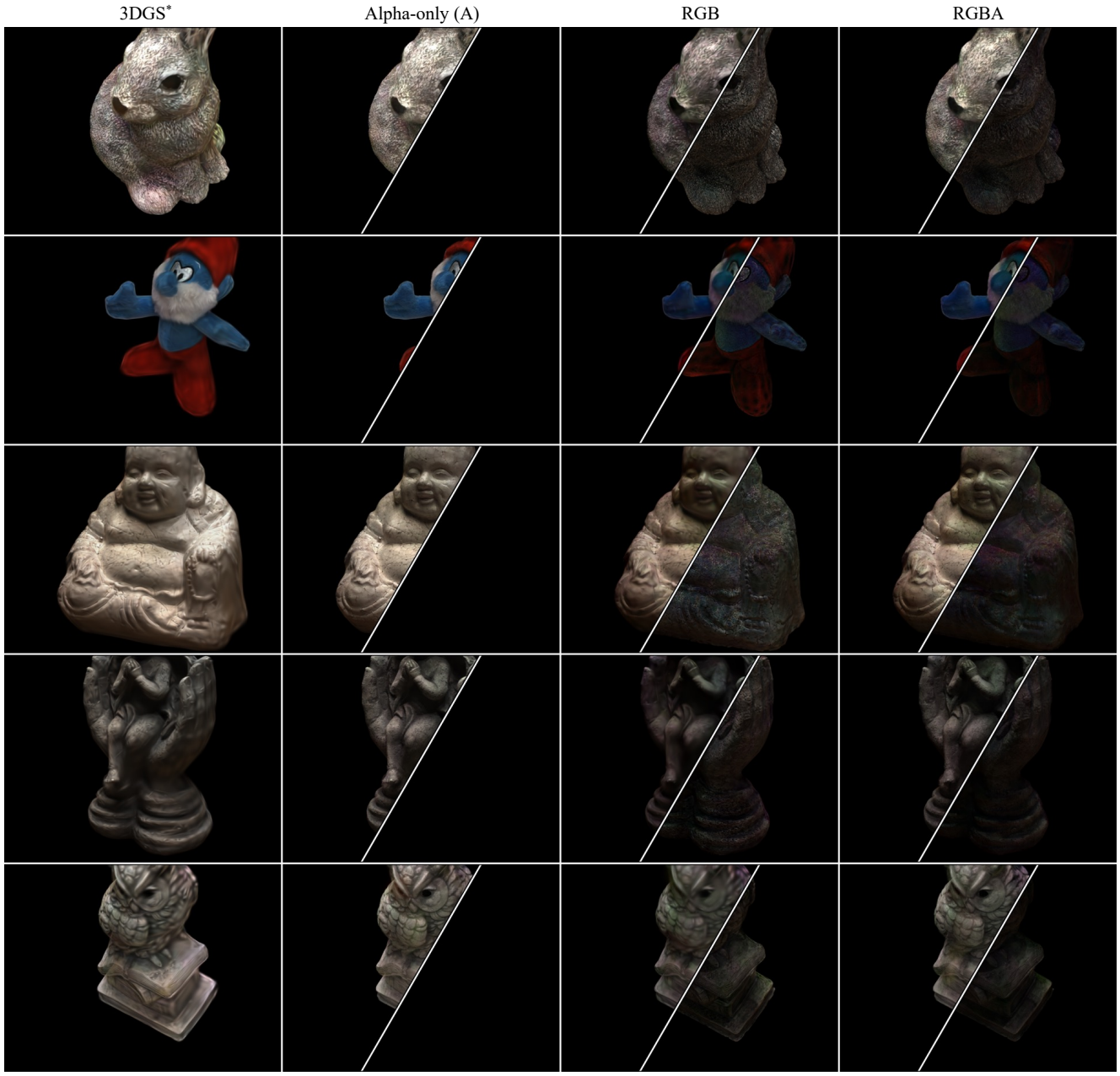


Figure A.30. **Color component decomposition visualization of the DTU dataset.** With alpha textures, the alpha-modulated and composited base color component can already recosntruct high-frequency textures, as shown in results of the alpha-only and RGBA Texuted Gaussians models. For models with only RGB textures, the base color component reconstructs lower frequency colors while the texture map color component reconstructs high frequency appearance.

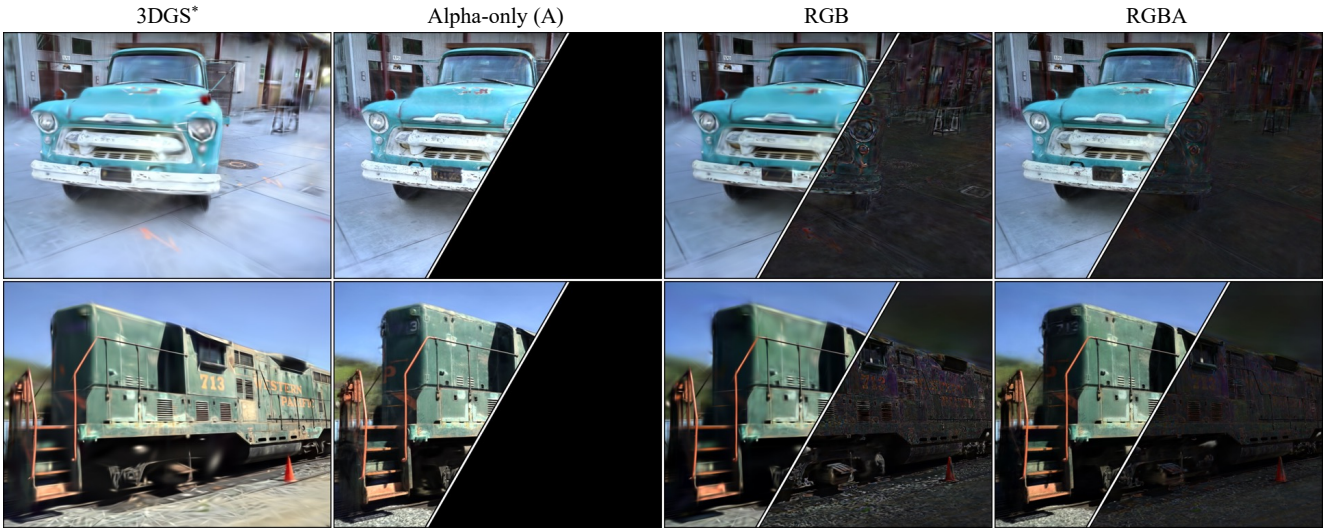


Figure A.31. **Color component decomposition visualization of the Tanks and Temples dataset.** With alpha textures, the alpha-modulated and composited base color component can already recosntruct high-frequency textures, as shown in results of the alpha-only and RGBA Texuted Gaussians models. For models with only RGB textures, the base color component reconstructs lower frequency colors while the texture map color component reconstructs high frequency appearance.

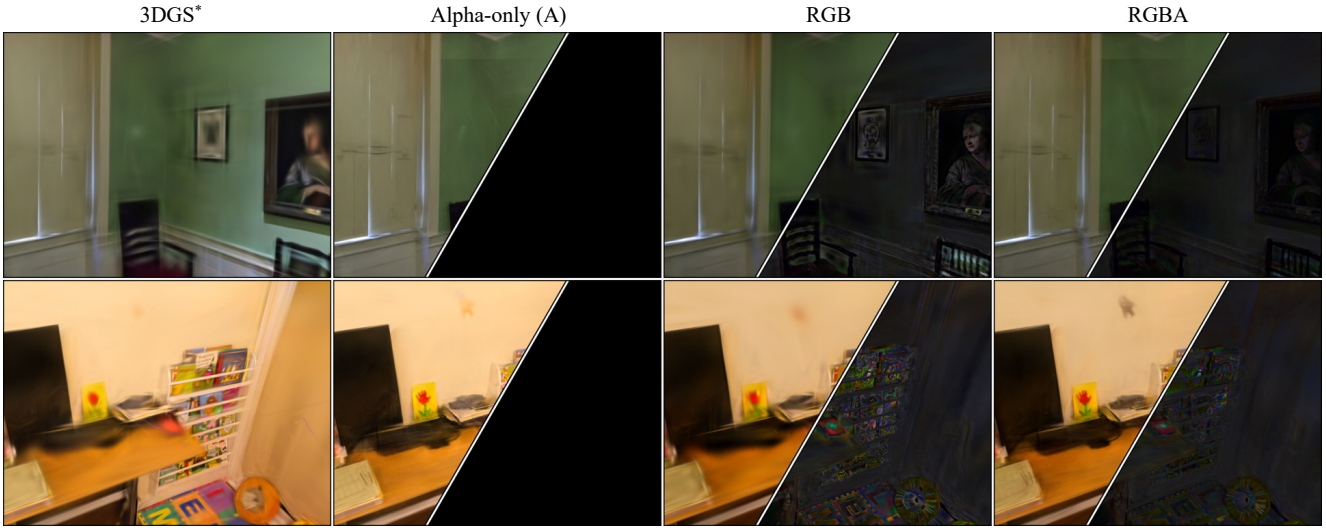


Figure A.32. **Color component decomposition visualization of the Deep Blending dataset.** With alpha textures, the alpha-modulated and composited base color component can already recosntruct high-frequency textures, as shown in results of the alpha-only and RGBA Texuted Gaussians models. For models with only RGB textures, the base color component reconstructs lower frequency colors while the texture map color component reconstructs high frequency appearance.