3D-GSW: 3D Gaussian Splatting for Robust Watermarking

Supplementary Material

Overview

This supplementary material is organized as follows:

Section 1 explains the implementation details of our watermark decoder. Section 2 elaborates on the additional details of Frequency-Guided Densification (FGD). Section 3 provides the further analysis of the gradient mask. Section 4 presents the details of the wavelet-subband loss. Section 5 shows further experimental results, including the comparison with the baselines, the challenging scenario, the robustness test, the failure case, and qualitative results of all message bits 32, 48, and 64.

1. Pre-training Decoder

In digital watermarking, the primary goal is to ensure that the message is decoded only from images where the message is embedded. In the fine-tuning method, a challenge arises: the model is typically trained with a unique message. This often leads to the decoder overfitting to a specific message, decoding the message from non-embedded rendered images. (See Tab. 1)

Method	Bit Acc↑ (W/ M)	Bit Acc↓ (W/O M)
Train decoder with 3D-GS	**99.53**	99.53
Ours (pre-train decoder)	97.37	**53.92**

Table 1. Bit Acc↑(W/ M) is the bit accuracy from images with a message. Bit Acc↑(W/O M) is the bit accuracy from images without a message. The results are conducted on Blender, LLFF, and Mip-NeRF 360 datasets with watermarked message 32bits.

To address the issue of the decoder memorizing the unique message, StegaNeRF [8] incorporates a classifier that determines the presence of a message. Furthermore, StegaNeRF [8] employs the specific patterns in the difference between the original image and the image containing the message, ensuring that the decoder cannot memorize the message. However, this method requires prior knowledge of the specific type of attack applied to the image to decode the message. Since identifying the specific type of distortion used by unauthorized users is challenging, there is a need for a method capable of decoding messages directly from a single image, independent of the applied attack. In our approach, we utilize a pre-trained decoder to prevent it from memorizing the unique message and to decode the message from a single image, ensuring robustness and reliability in the message decoding process.

1.1. Architecture and Implementation Details

Architecture. We utilize the HiDDeN architecture [16], which consists of the encoder, decoder, and distortion layer. The encoder has 4 Conv-BN-ReLU blocks, kernel size 3, stride 1, and padding 1. The decoder consists of 7 Conv-BN-ReLU blocks with 64 filters. Average pooling is performed over all spatial dimensions, and a final $L \times L$ linear layer produces the decoded message, where L is the number of bits in the watermark message. The distortion layer has crop, scaling, and JPEG compression.

Implementation Details. We train the full HiDDeN architecture [16] on $\{32,48,64\}$ bits using the MS-COCO dataset [11]. This process is only required once per bit length. The optimization is performed with Adam [6] and a learning rate of 10^{-4}. We employ Mean Squared Error as an image loss and a message loss. To focus on decoder performance, we set the image loss hyper-parameter to 0 and the message loss hyper-parameter to 1 in the pre-training decoder process. By following the strategies from Stable Signature [3], we incorporate PCA whitening into the decoder to avoid bias in the decoded message, enhancing the robustness and reliability of the decoder.

Input of Decoder	Bit Acc(%)↑	PSNR ↑	SSIM ↑	LPIPS ↓
Pixel	70.93	31.63	0.973	0.050
DFT	50.10	**40.36**	0.993	0.024
DCT	50.08	39.97	**0.994**	**0.016**
LL Subband (Level 1)	94.17	30.86	0.972	0.051
LL Subband (Level 2)	**97.37**	35.08	0.978	0.043
LL Subband (Level 3)	94.95	33.19	0.973	0.047
LL Subband (Level 4)	90.06	32.41	0.970	0.021

Table 2. We transform the rendered image to the frequency domain, using DFT, DCT, and DWT. We choose low-frequency components for each transformed image as an input of the decoder to fine-tune 3D-GS. Pixel indicates that the rendered image itself is the input to the decoder. Results represent the average score across Blender, LLFF, and Mip-NeRF 360 datasets using 32-bit messages.

1.2. Select the input of Decoder

JPEG compression methods [12, 15] tend to remove high-frequency of the image. WateRF [4] shows that the low-frequency of Discrete Wavelet Transform (DWT) enables robust message embedding for radiance field watermarking. Following these properties, we choose the low-frequency as the input of the decoder. As shown in Tab. 2, we observe that DWT at level 2, there is a good balance between bit accuracy and rendering quality. Notably, other transformations like Discrete Fourier Transform (DFT) and Discrete Cosine Transform (DCT) fail to embed the message. Furthermore, the pixel domain can embed messages but does not guarantee high bit accuracy. From these results, we

Dataset	# Original	# Remove	# Split	# After FGD	Removal Ratio	FPS↑ (Before/After/Ratio)
Blender	0.29M	0.07M	0.03M	0.25M	13.58%	208.05 / 231.62 /+11.33%
LLFF	0.98M	0.36M	0.06M	0.68M	30.56%	75.24 / 90.55 /+20.35%
Mip-NeRF 360	3.36M	1.06M	0.09M	2.39M	28.85%	56.65 / 72.68 /+28.30%

Table 3. The impact of Frequency-Guided Densification (FGD) on the number of 3D Gaussians and rendering speed. We show the original number of 3D Gaussians, the number of removed and split 3D Gaussians, and the final count after FGD. The results are shown separately for Blender, LLFF, and Mip-NeRF 360 datasets.

Methods	Bit Acc↑	PSNR ↑	SSIM ↑	LPIPS ↓
3D-GSW (W/O split)	96.72	33.72	0.970	0.062
3D-GSW (W/ split) (Ours)	**97.37**	**35.08**	**0.978**	**0.043**

Table 4. We compare bit accuracy and rendering quality of FGD with split (W/ split) and without split (W/O split). Results represent the average score across Blender, LLFF, and Mip-NeRF 360 datasets using 32-bit messages.

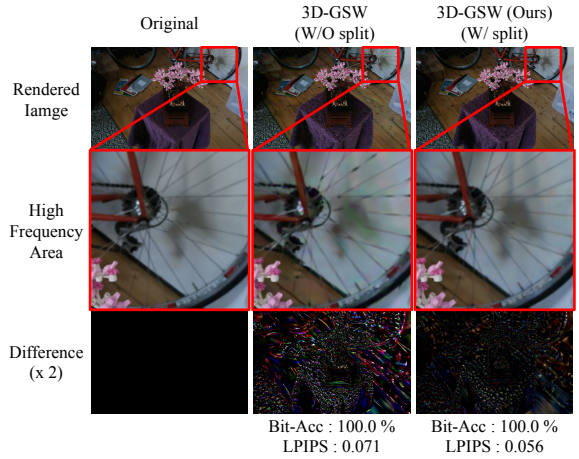

Figure 1. We compare the difference between FGD with split (W/ split) and without split (W/O split). Both images are embedded by 32-bit messages.

Methods	Bit Acc↑	PSNR ↑	SSIM ↑	LPIPS ↓
3D-GSW (FGD W/ DWT)	95.12	34.23	0.976	0.054
3D-GSW (FGD W/ DCT)	95.48	34.20	0.976	0.054
3D-GSW (FGD W/ DFT) (Ours)	**97.37**	**35.08**	**0.978**	**0.043**

Table 5. We compare the effectiveness of DFT with other methods for transforming patches into the frequency domain. Results represent the average score across Blender, LLFF, and Mip-NeRF 360 datasets using 32-bit messages.

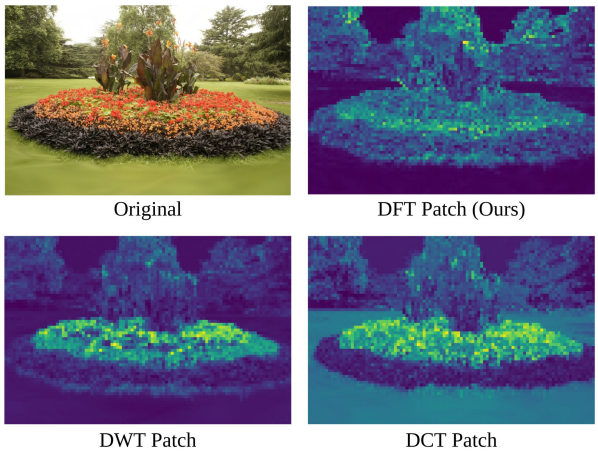

Figure 2. We visualize the patches converted by DFT, DWT, and DCT, applying a mask that emphasizes high frequency.

choose DWT at level 2 to embed the message into 3D-GS.

2. Frequency-Guided Densification

2.1. Effectiveness of Split 3D Gaussians

From 3D-GS compression work [7], small 3D Gaussians have a negligible contribution to the rendering quality due to their minimal volume. As noted in image compression work [12], human visual system is less sensitive to high-frequency components. Furthermore, as highlighted by error-based densification [1], since a small number of large 3D Gaussians are responsible for capturing high-frequency details, there are cases where 3D-GS has difficulty reconstructing high-frequency areas. To address these properties, we choose 3D Gaussians in high-frequency areas to preserve the rendering quality of the pre-trained 3D-GS. Additionally, to effectively render high-frequency details during the fine-tuning process, each selected 3D Gaussian is split into two smaller 3D Gaussians.

As shown in Tab. 4, while both methods ensure high bit accuracy, our FGD significantly enhances the rendering quality. From Fig. 1, splitting 3D Gaussians reconstructs the high-frequency components in more detail. Notably, FGD, which does not split 3D Gaussians, covers the high-frequency component by increasing the volume of 3D Gaussians. These results quantitatively and qualitatively demonstrate the effectiveness and efficiency of split 3D Gaussians. Furthermore, as shown in Tab. 3, although splitting increases the number of 3D Gaussians, the number of removed 3D Gaussians is relatively higher. Consequently, the total number of 3D Gaussians after FGD is reduced, leading to improved rendering efficiency while maintaining quality.

2.2. Emphasize High-Frequency Patch

In phase 2 of FGD, we find the patches with strong intensity of high-frequency signals to split 3D Gaussians within

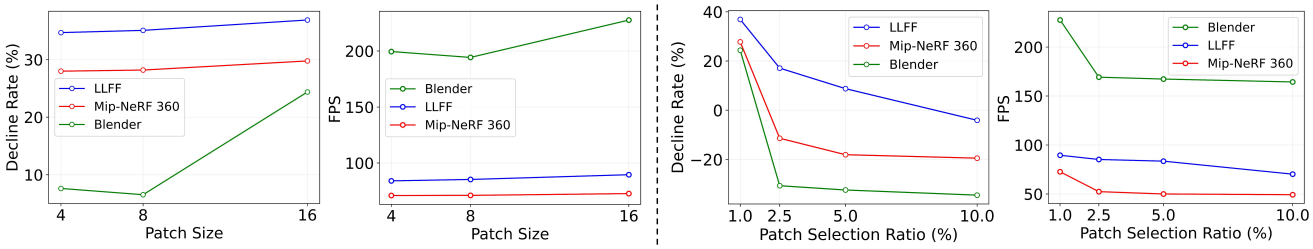

Figure 3. We compare the effects of patch size and patch selection ratio on the number of 3D Gaussians in 3D-GS after undergoing FGD. For patch size, we set patch selection ratio to 1 %. For patch selection ratio, we set patch size to 16 × 16.

Patch Size	Bit Acc↑	PSNR ↑	SSIM ↑	LPIPS ↓
4 × 4	94.12	33.13	0.975	0.054
8 × 8	95.48	34.20	0.976	0.050
16 × 16 (Ours)	**97.37**	**35.08**	**0.978**	**0.043**

Table 6. We compare performance by patch size. Results represent the average score across Blender, LLFF, and Mip-NeRF 360 datasets using 32-bit messages.

Top K %	Bit Acc↑	PSNR ↑	SSIM ↑	LPIPS ↓
10	94.21	32.34	0.975	0.061
5	95.49	33.06	0.975	0.050
2.5	97.19	33.37	**0.978**	0.044
1 (Ours)	**97.37**	**35.08**	**0.978**	**0.043**

Table 7. We compare performance by selection ratio, denoted as K. Results represent the average score across Blender, LLFF, and Mip-NeRF 360 datasets using 32-bit messages.

those patches. To measure the intensity of high-frequency signals globally across the patch, Discrete Fourier Transform (DFT) is applied. By analyzing the frequency signals of the patch through DFT, the value of the high-frequency areas is calculated to quantify their intensity through a mask to emphasize high-frequency signals in the patch (See main paper Sec.3.4).

To evaluate the effectiveness of DFT, we also utilize other frequency transforms, such as DWT and DCT. In the case of DWT, we decompose the patch into wavelet subbands and measure the intensity as the average value of the high-frequency subbands. For DCT, we transform the patch into the frequency domain and measure the intensity as the average value of the high-frequency signals in the DCT.

As shown in Tab. 5, our method enhances bit accuracy and rendering quality, which have a trade-off relationship. Furthermore, from Fig. 2, we observe that DWT detects high-frequency regions in specific image areas, leveraging its ability to analyze local information. In contrast, DCT has difficulty in detecting high-frequency areas. Notably, DFT effectively detects high-frequency areas across the entire image due to its characteristic of analyzing global information. These results show that DFT effectively evaluates the distribution of high-frequency signals within the patch.

2.3. Patch Size and Selection Patch

In this section, we explore how patch size and patch selection ratio affect performance. Since we remove 3D Gaussians to embed a robust message in phase 1 of FGD, it is crucial to avoid splitting more Gaussians than are removed during phase 2 of FGD. As shown in Fig. 3, we observe that increasing the patch size split 3D Gaussians without disturbing the results of removal 3D Gaussians in phase 1, enhancing real-time rendering. Tab. 6 shows that a large patch size improves bit accuracy and rendering quality. Fig. 3 shows that selecting more patches causes more 3D Gaussians to be split, increasing the number of 3D Gaussians after FGD. Additionally, Tab. 7 shows that as the patch selection ratio increases, all performance decreases. These results show that an appropriate patch size and patch selection are effective for message embedding and rendering quality.

3. Gradient Mask

In this section, we present the additional analysis about a gradient mask. Our gradient mask is calculated as follows:

$$w = \frac{1}{e^{|\theta|^{\beta}}}, \quad z = \frac{w}{\sum_{i=1}^{N_{\mathcal{G}'_o}} w_i} \tag{1}$$

, where $\mathcal{G}'_o$, θ, $N_{\mathcal{G}'_o}$, i and $\beta > 0$ are respectively denoted as 3D-GS passed through FGD, the parameter of $\mathcal{G}'_o$, the number of 3D Gaussians in $\mathcal{G}'_o$, the index of 3D Gaussians and the strength of gradient manipulation.

3.1. Effectiveness of Exponential Function

In FGD process, we modify 3D Gaussians in the pre-trained 3D-GS to enhance the robustness of the message and rendering quality. Since 3D-GS passed through FGD provides high-quality rendering, it is crucial to preserve the rendering quality during the fine-tuning process. To achieve this, we reduce gradient size to minimize changes in the parameters of 3D-GS passed through FGD, utilizing an exponential function. As shown in Tab. 8, we observe that the exponential function maintains the parameters of 3D-GS passed through FGD. Notably, the exponential function preserves color parameters identically. Tab. 9 shows that the exponential function has superior rendering quality compared to other methods.

Methods	Cosine Similarity of Parameters ↑				
	Color	Opacity	Scale	Rotation	Position
W/O exponential	0.982	**0.997**	0.991	0.989	**0.999**
W/ exponential (Ours)	**0.999**	**0.997**	**0.999**	**0.995**	**0.999**

Table 8. We compare similarity of parameters for a exponential function. We calculate cosine similarity between 3D-GS passed through fine-tuning process and 3D-GS passed through FGD process. Results represent the average score across Blender, LLFF, and Mip-NeRF 360 datasets using 32-bit messages.

Methods	Bit Acc↑	PSNR ↑	SSIM ↑	LPIPS ↓
3D-GS After FGD (N/M)	-	**48.66**	**0.998**	**0.002**
StegaNeRF [8]+3D-GS [5]	93.15	32.68	0.953	0.049
WateRF [4]+3D-GS [5]	93.42	30.49	0.956	0.050
W/O exponential	93.99	27.54	0.898	0.093
W/ exponential (Ours)	**97.37**	35.08	0.978	0.043

Table 9. We compare bit accuracy and rendering quality for a exponential function with other methods. 3D-GS After FGD (N/M) refers to 3D-GS that has gone through FGD before embedding the message. Results represent the average score across Blender, LLFF, and Mip-NeRF 360 datasets using 32-bit messages.

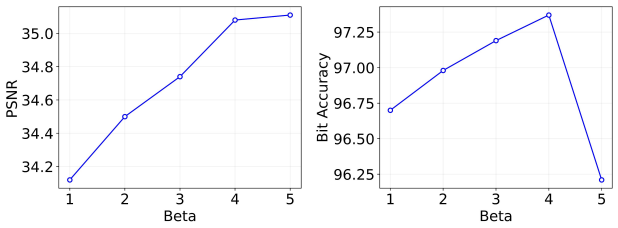

Figure 4. We compare the strength of the gradient mask. Beta controls the size of the gradient. A larger beta reduces the gradient size further.

3.2. Strength of Gradient Mask

Following Eq. 1, β controls the strength of gradient manipulation. As shown in Fig. 4, we observe that increasing β enhances the rendering quality. However, bit accuracy does not continue to increase but decreases when β becomes greater than 4. This shows the trade-off between bit accuracy and rendering quality, and we find that $\beta = 4$ achieves a good balance between bit accuracy and rendering quality.

4. Wavelet-subband loss

4.1. Effectiveness of Wavelet-subband loss

Since 3D-GS increases the volume of fewer 3D Gaussians to render high-frequency areas, there is a tendency to lose detailed information in the high-frequency areas. To address this issue, in phase 2 of FGD, we split 3D Gaussians into smaller ones. Furthermore, we propose a wavelet-subband loss to avoid losing details in the high-frequency areas during the fine-tuning process. Wavelet-subband loss is designed to focus on the high-frequency, utilizing only high-frequency subbands (LH, HL, HH).

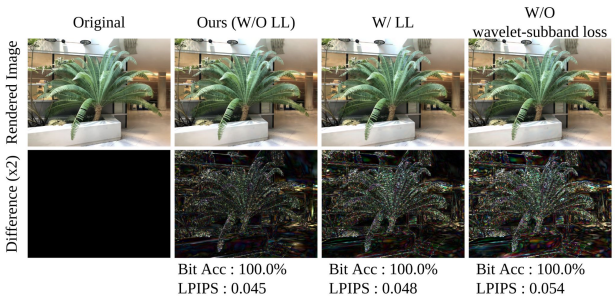

Figure 5. We show the differences(x2) between the watermarked image and the original image. Our method enhances the detail in the high-frequency areas.

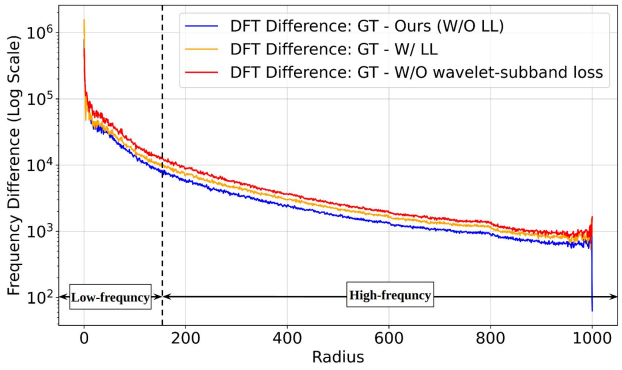

Figure 6. We show the differences in the frequency domain between the watermarked image and the original image. We transform both images to the frequency domain using DFT. Radius is the distance from the center of the image. The larger radius represents the high-frequency areas. The blue line represents our result. A lower value on the y-axis indicates greater similarity to the original image. We show this result on the LLFF dataset using 32-bit messages.

As shown in Fig. 5, we observed that the wavelet-subband loss using only high-frequency subbands achieves better rendering quality. To further analyze the rendering quality in the high-frequency areas, we convert both the original image and the rendered image to the frequency domain and calculate the difference between them. Fig. 6 shows that our method is similar to using LL subbands in the low-frequency areas, but achieves results more similar to the original in the high-frequency areas. These results quantitatively and qualitatively show the effectiveness of wavelet-subband loss in enhancing quality in high-frequency areas.

5. Additional Results

5.1. Comparison with the baselines

As shown in Fig. 7, we show the rendering quality with other methods. We observe that WateRF [4] and StegaNeRF [8] have color artifacts in the rendered image due to increasing the volume of 3D Gaussians. In contrast, our method preserves rendering quality, while achieving high bit accuracy. To illustrate the differences between the original and watermarked image, we visualize the normalized pixel intensity difference in Fig. 8, following previous watermarking methods [2, 14]. Our method achieves the smallest difference, demonstrating superior rendering quality.

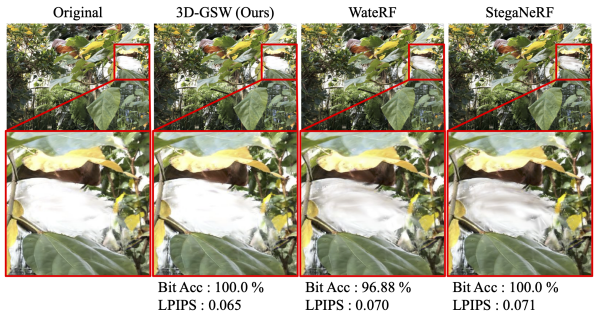

Figure 7. We compare the rendering quality with other methods. We achieve results that are most similar to the original rendering quality.

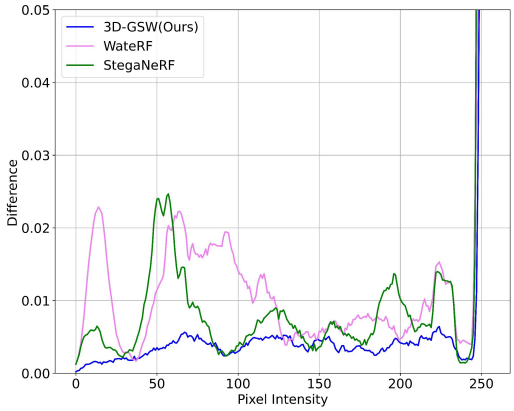

Figure 8. We show the normalized pixel intensity difference between original and the watermarked image. The blue line represents our result. A lower value on the y-axis indicates greater similarity to the original image.

5.2. Robustness Test

Fig. 9 and Fig. 10 show the bit accuracy when the distortion level is varied. We observe that our bit accuracy is consistently higher than other methods across a wide range of strengths. Notably, incorporating FGD improves performance further, highlighting its role in enhancing robustness to image and model distortions. These results show that our method is a highly effective and robust watermarking method.

5.3. Challenging Scenario

To demonstrate the practicality and scalability, we experiment on a 4D-GS task with a method named Spacetime [10] using the neural 3D Video dataset [9], which consists of six dynamic scenes with a resolution of 2028×2704. Tab. 10 shows a performance comparison between ours and baselines. Our method consistently achieves superior performance on dynamic scenes, particularly in terms of temporal consistency. This highlights its effectiveness in handling dynamic scenes while maintaining high rendering quality. Furthermore, the scalability and practicality of our method make it well-suited for various real-world applications.

Method	Bit-Acc↑	PSNR↑	SSIM↑	LPIPS↓	FVD↓
StegaNeRF [8]+Spacetime [10]	94.75%	31.26	0.940	0.0895	315.11
WateRF [4]+Spacetime [10]	95.11%	30.51	0.939	0.0909	166.15
3D-GSW +Spacetime [10]	**97.62%**	**32.51**	**0.958**	**0.0653**	**125.98**

Table 10. We show the scalability of our method to dynamic scenes. Results represent the average score across neural 3D Video dataset [9] using 32-bit messages.

5.4. Failure Case

Recently, compression works have emerged in the spotlight in 3D-GS. Thus, we can consider one scenario, in which unauthorized users compress watermarked 3D-GS. To perform the compression of pre-trained 3D-GS, the unauthorized users must have the same training data of the target 3D asset to train the 3D-GS. We assume that the unauthorized users have the same training data as the watermarked 3D-GS and proceed with compression. The compression is conducted by Simon [13]. As shown in Tab. 11, we observe that all methods lose the message in the compression process. This result is a failure case of the entire radiance field watermarking research. Although our method is robust when the unauthorized users only have 3D-GS, we will work on the case in the future when they proceed with compression.

Methods	Bit Acc↑	PSNR ↑	SSIM ↑	LPIPS ↓
StegaNeRF [8]+3D-GS [5]	52.29	38.68	**0.986**	**0.018**
WateRF [4]+3D-GS [5]	50.86	38.77	0.985	**0.018**
3D-GSW (Ours)	**55.51**	**38.83**	**0.986**	**0.018**

Table 11. We show the impact of the compression on bit accuracy and rendering quality. All methods lose the embedded message due to compression.

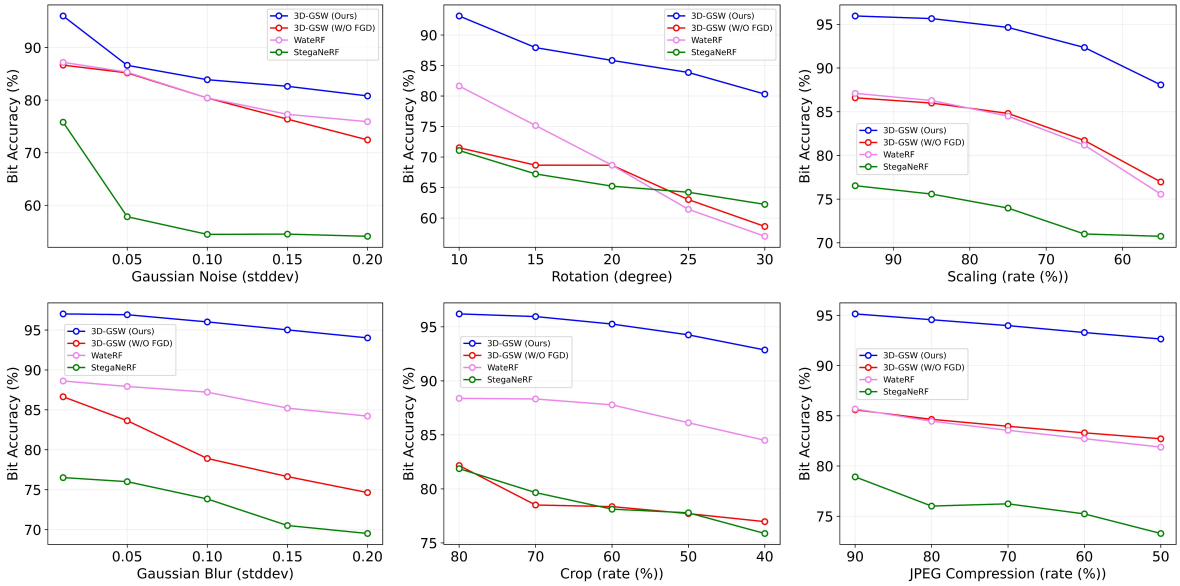

Figure 9. Bit accuracy for WateRF [4], StegaNeRF [8], 3D-GSW (W/O FGD), and our method for various image distortions and distortion strengths. The blue line represents our results. Our method outperforms other methods.

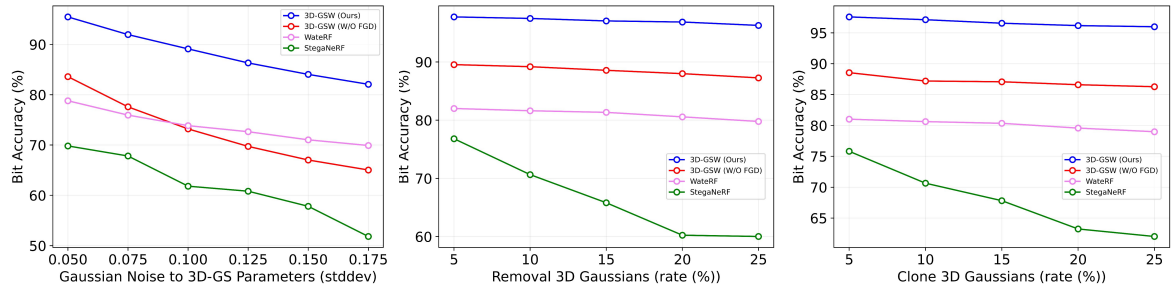

Figure 10. Bit accuracy for WateRF [4], StegaNeRF [8], 3D-GSW (W/O FGD), and our method for model distortions and distortion strengths. We conduct two model distortion: 1) We add model Gaussian noise to 3D-GS parameters. 2) We remove randomly 3D Gaussians. 3) We clone randomly 3D Gaussians. The blue line represents our results. Our method outperforms other methods.

5.5. Qualitative results

From Fig. 11 to Fig. 19, we visualize all results rendered from our method and the difference ($\times$ 2) between the original image and watermarked image.

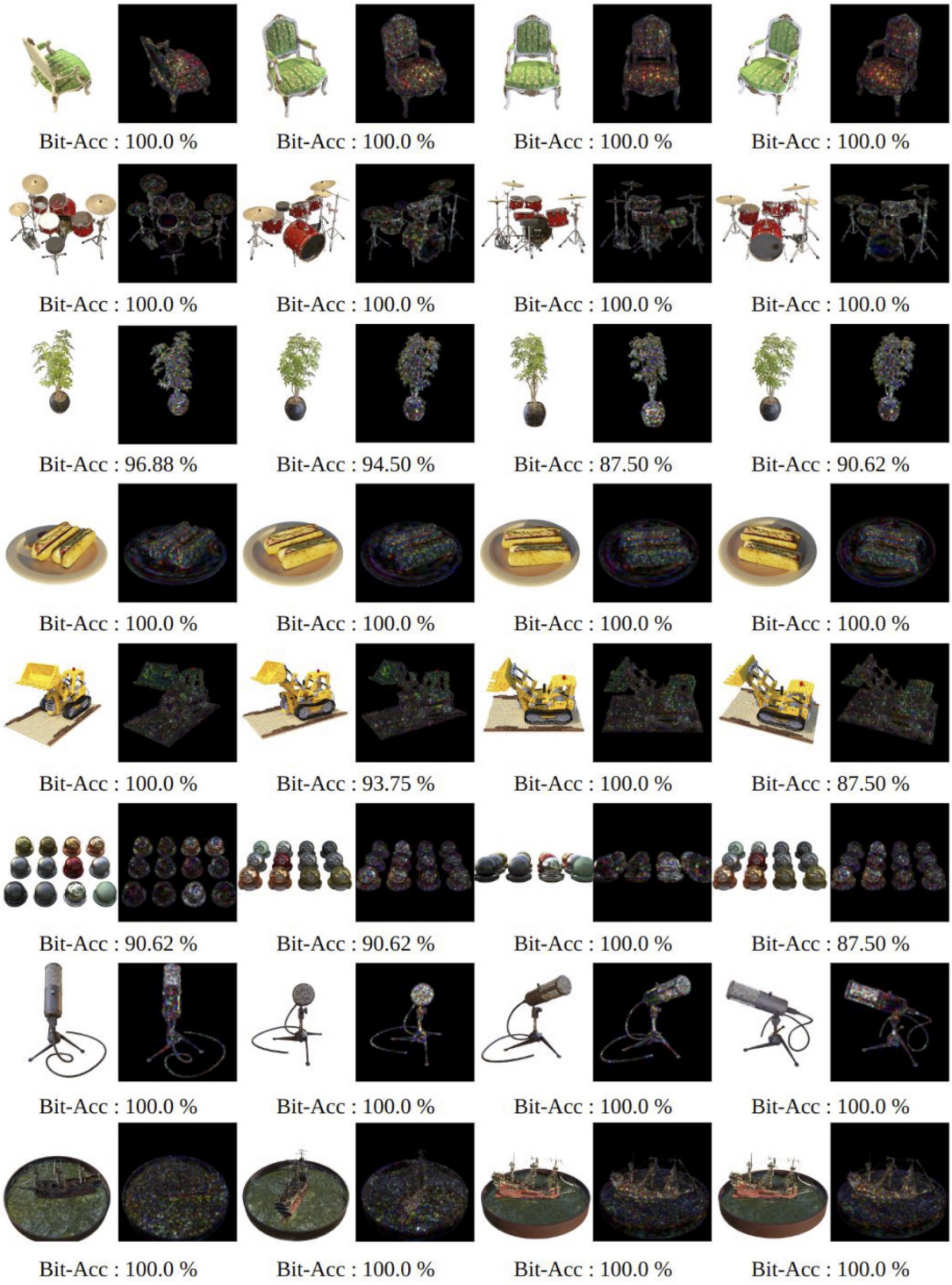

Figure 11. Rendering quality of various rendering outputs using our method on Blender dataset. We show the differences ($\times$ 2). The closer it is to white, the bigger the difference between the ground truth and the image. We show the results on 32 bits.

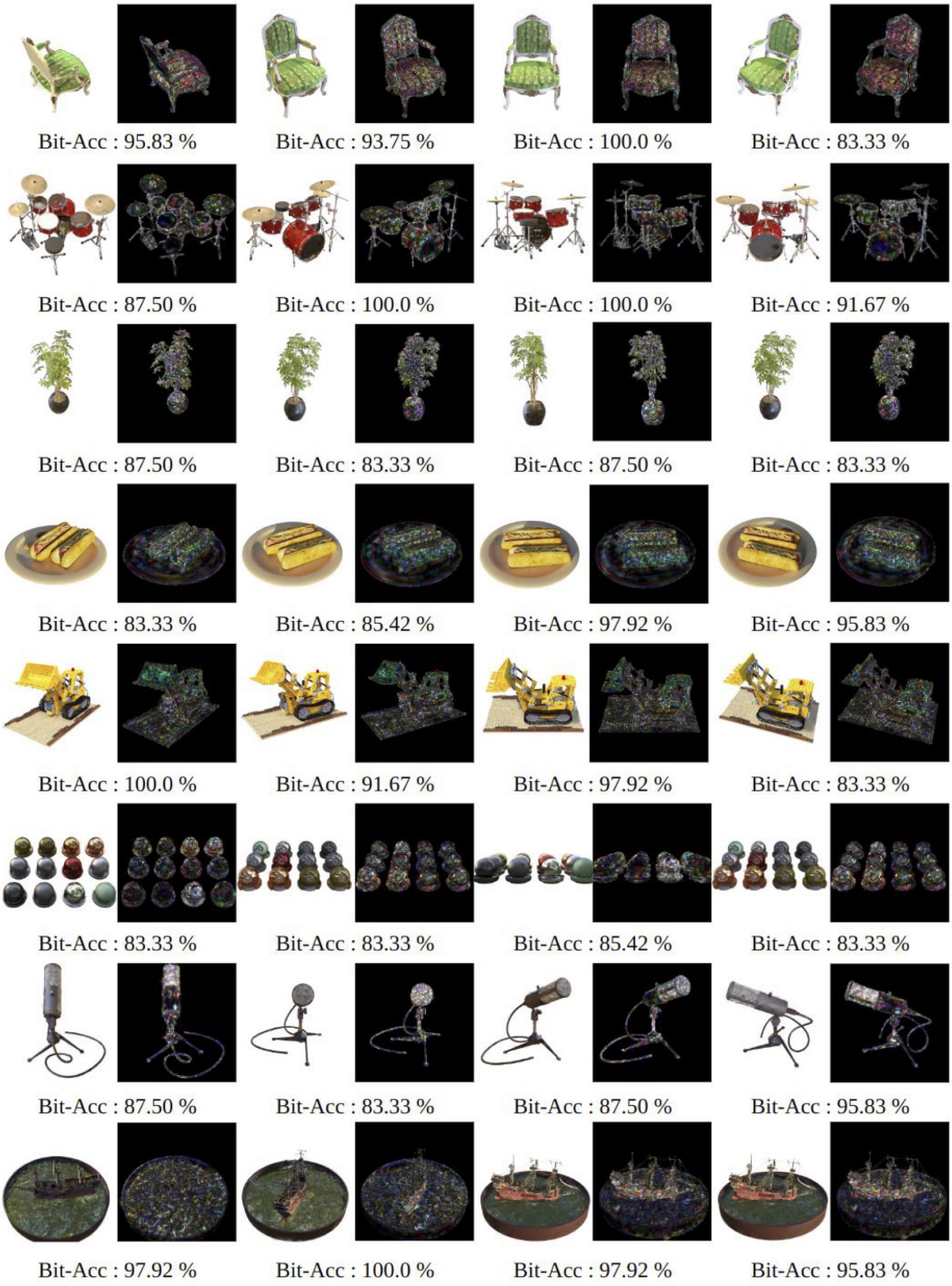

Figure 12. Rendering quality of various rendering outputs using our method on Blender dataset. We show the differences ($\times$ 2). The closer it is to white, the bigger the difference between the ground truth and the image. We show the results on 48 bits.

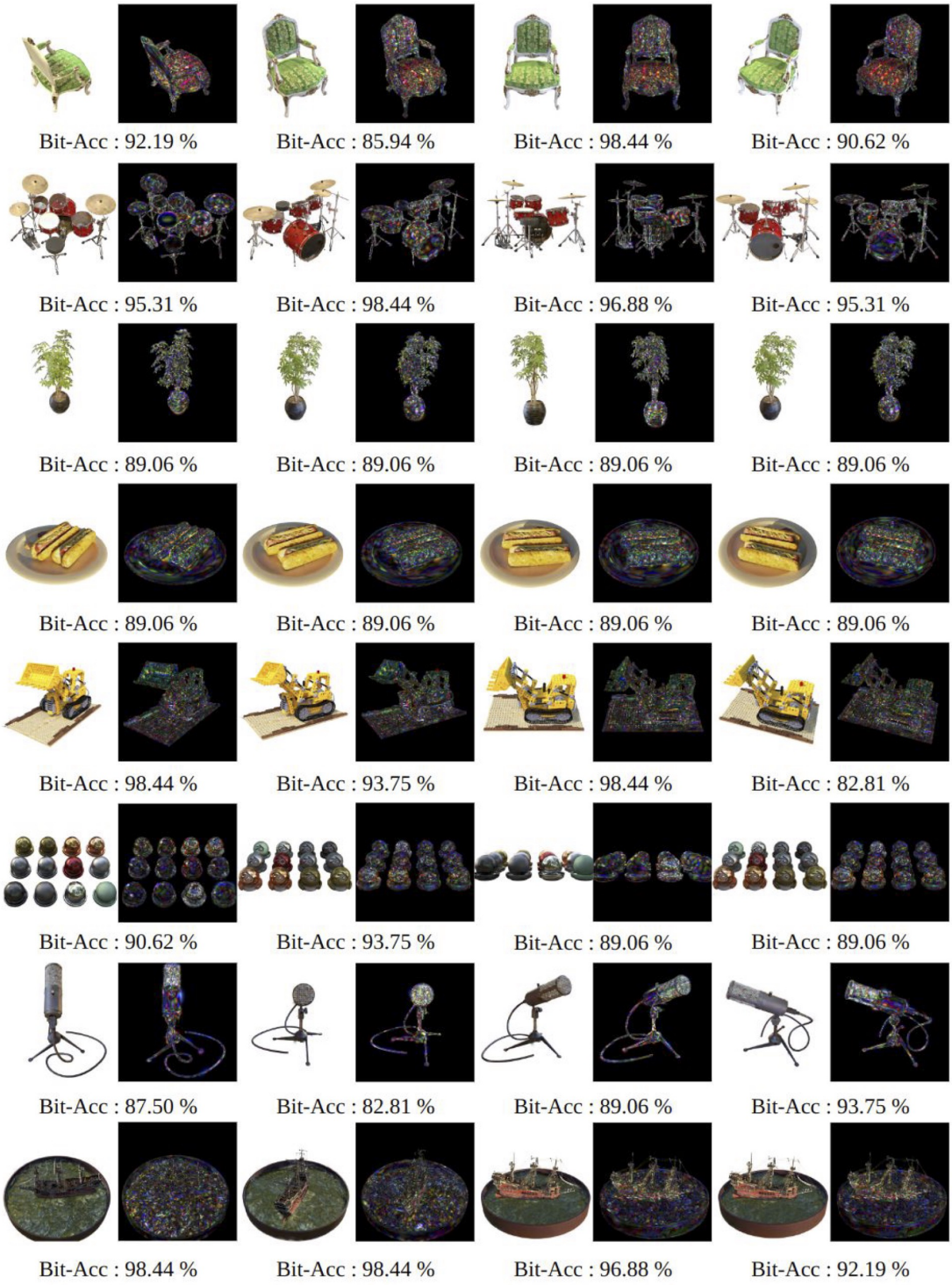

Figure 13. Rendering quality of various rendering outputs using our method on Blender dataset. We show the differences ($\times$ 2). The closer it is to white, the bigger the difference between the ground truth and the image. We show the results on 64 bits.

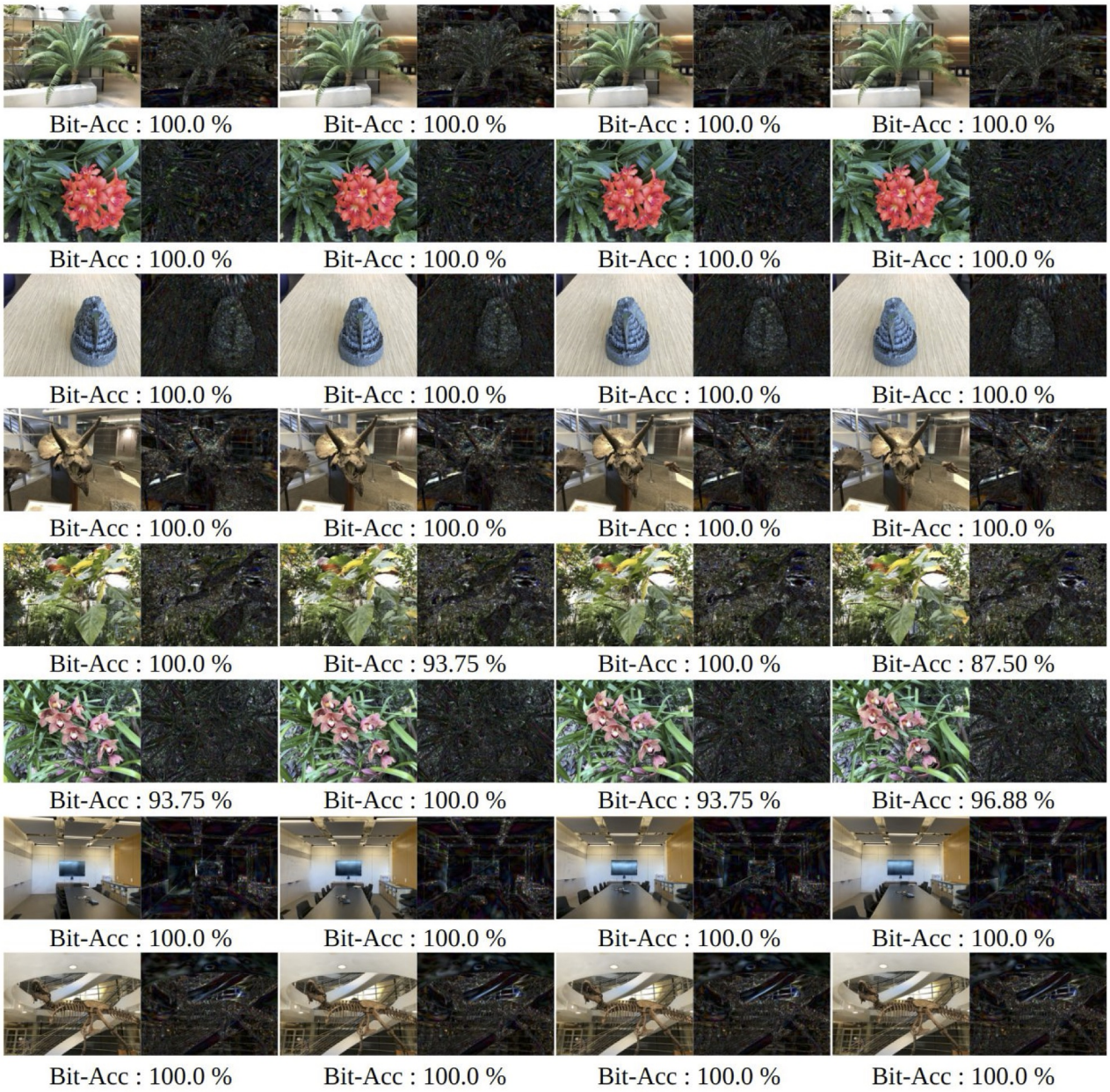

Figure 14. Rendering quality of various rendering outputs using our method on LLFF dataset. We show the differences ($\times$ 2). The closer it is to white, the bigger the difference between the ground truth and the image. We show the results on 32 bits.

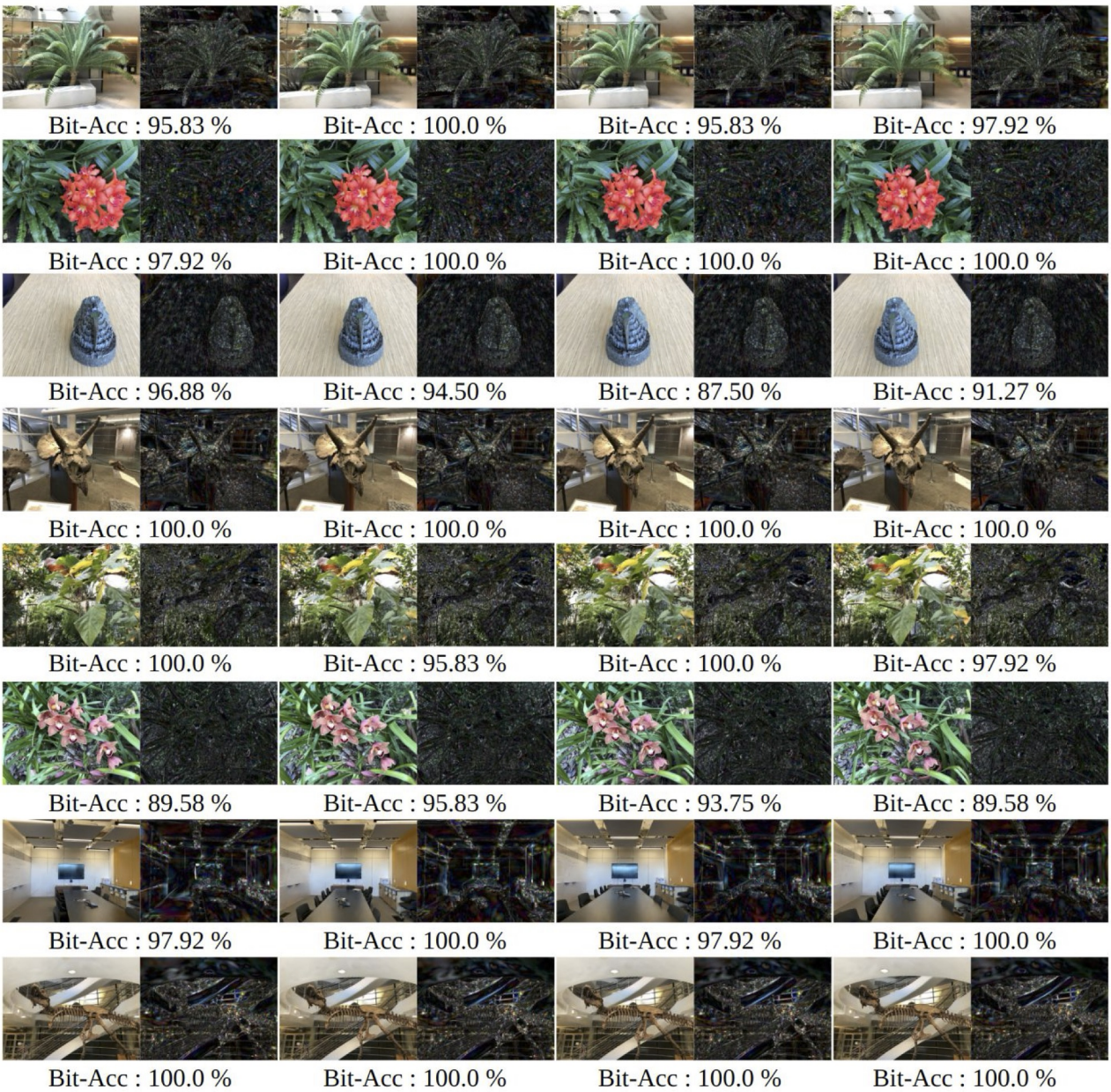

Figure 15. Rendering quality of various rendering outputs using our method on LLFF dataset. We show the differences ($\times$ 2). The closer it is to white, the bigger the difference between the ground truth and the image. We show the results on 48 bits.

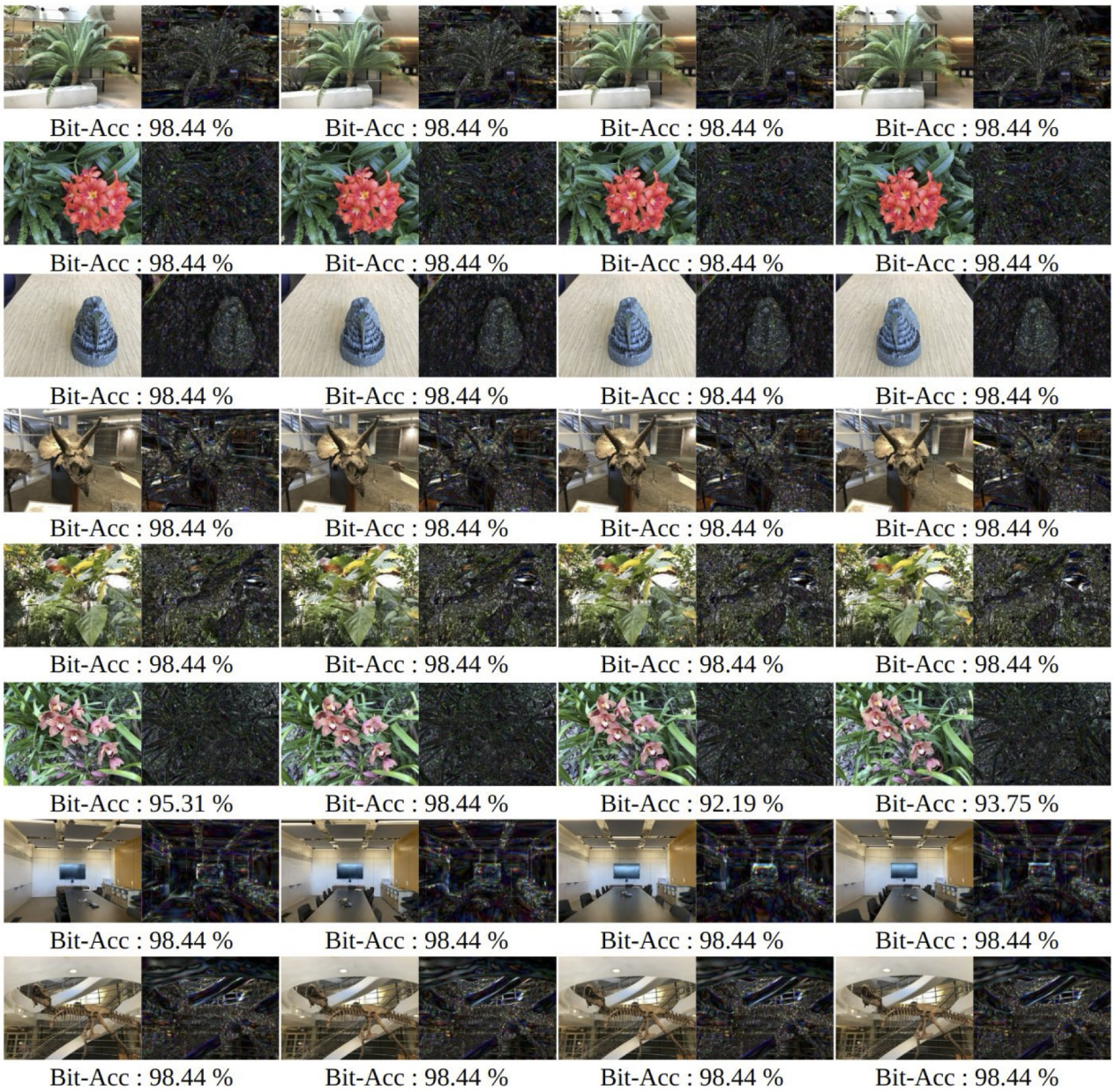

Figure 16. Rendering quality of various rendering outputs using our method on LLFF dataset. We show the differences ($\times$ 2). The closer it is to white, the bigger the difference between the ground truth and the image. We show the results on 64 bits.

Figure 17. Rendering quality of various rendering outputs using our method on Mip-NeRF 360 dataset. We show the differences ($\times$ 2). The closer it is to white, the bigger the difference between the ground truth and the image. We show the results on 32 bits.

Figure 18. Rendering quality of various rendering outputs using our method on Mip-NeRF 360 dataset. We show the differences ($\times$ 2). The closer it is to white, the bigger the difference between the ground truth and the image. We show the results on 48 bits.

Figure 19. Rendering quality of various rendering outputs using our method on Mip-NeRF 360 dataset. We show the differences ($\times$ 2). The closer it is to white, the bigger the difference between the ground truth and the image. We show the results on 64 bits.

References

[1] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. Revising densification in gaussian splatting. *arXiv preprint arXiv:2404.06109*, 2024. 2

[2] Wang Cai-Yin, Kong Xiang-Wei, and Li Chao. Process color watermarking: the use of visual masking and dot gain correction. *Multimedia Tools and Applications*, 76:16291–16314, 2017. 5

[3] Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. The stable signature: Rooting watermarks in latent diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22466–22477, 2023. 1

[4] Youngdong Jang, Dong In Lee, MinHyuk Jang, Jong Wook Kim, Feng Yang, and Sangpil Kim. Waterf: Robust watermarks in radiance fields for protection of copyrights, 2024. 1, 4, 5, 6

[5] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4):1–14, 2023. 4, 5

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

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

[8] Chenxin Li, Brandon Y Feng, Zhiwen Fan, Panwang Pan, and Zhangyang Wang. Steganerf: Embedding invisible information within neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 441–453, 2023. 1, 4, 5, 6

[9] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5521–5531, 2022. 5

[10] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8508–8520, 2024. 5

[11] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 1

[12] Zihao Liu, Tao Liu, Wujie Wen, Lei Jiang, Jie Xu, Yanzhi Wang, and Gang Quan. Deepn-jpeg: A deep neural network favorable jpeg-based image compression framework. In *Proceedings of the 55th annual design automation conference*, pages 1–6, 2018. 1, 2

[13] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10349–10358, 2024. 5

[14] Alastair Reed, Tomáš Filler, Kristyn Falkenstern, and Yang Bai. Watermarking spot colors in packaging. In *Media Watermarking, Security, and Forensics 2015*, pages 46–58. SPIE, 2015. 5

[15] Mengdi Sun, Xiaohai He, Shuhua Xiong, Chao Ren, and Xinglong Li. Reduction of jpeg compression artifacts based on dct coefficients prediction. *Neurocomputing*, 384:335–345, 2020. 1

[16] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 657–672, 2018. 1