MultiNeRF: Multiple Watermark Embedding for Neural Radiance Fields

Supplementary Material

A. Training HiDDeN Decoder

Training. To enable robust message extraction from the rendered images, we adopt a standard deep watermarking approach HiDDeN [6], we use the training code from [1] to train the HiDDeN decoder. Fig. 1 shows the training of HiDDeN, where the encoder takes a cover image and a binary message to watermark, then outputs a watermarked image. A noise layer simulates common distortions (e.g., cropping, JPEG compression, rotation). The decoder then attempts to recover the embedded bits from the distorted watermarked image.

Because our main pipeline only requires the decoder, once training converges, we discard the encoder entirely and keep only the decoder.

Optimization. We train the decoder on MS-COCO 2014 dataset [2] keeping the image resolution at 256 x 256. The optimization is carried out on 4 GPUs, with Lamb optimizer [5]. The batch size is kept at 16, while the learning rate is $5e-3$

Whitening. We observe that the outputs could be biased or correlated when the trained decoder is exposed to non-watermarked images later. To address this, we apply PCA-whitening to the linear layer of the decoder, similar to [1]. This step de-correlates the output bits and helps eliminate systematic biases (as shown in Fig. 2 & Fig. 3), ensuring the decoder provides more reliable and unbiased watermark predictions in our method.

B. Differentiable Augmentation Layer.

We introduce a differentiable augmentation layer to strengthen our watermark decoder's robustness against image degradations. During training, the rendered image is randomly subjected to one or more lightweight augmentations (e.g., blur, JPEG compression, color jitter). The augmented image is then fed into the watermark decoder. Because these augmentations are implemented with differentiable operations (via Kornia [3] and the differentiable JPEG compression implementation comes from [4]), the decoder can learn to handle distortions in an end-to-end fashion.

This is how the augmentation is organized:

1.Augmentation Pool. We maintain a set $\{A_1, A_2, \ldots, A_m\}$ of possible transformations, such as random blur, random noise, or random brightness. Each augmentation is parameterized by a probability $p = 0.75$ that determines whether it is applied.

2. Random Selection. At every training step, we randomly select 2 augmentations from the pool and apply them sequentially. As a result, each image may be subjected to different combinations of distortions, making the model more robust.

Tab. 1 lists the parameter ranges for each augmentation used in training the model. Fig. 4 shows the visual examples of each augmentation.

Transformation	Parameter(s)
JPEG Quality	30
Brightness	(0.9, 1.1)
Contrast	(0.9, 1.1)
Color Jitter	(0.05, 0.05, 0.05, 0.01)
Gaussian Blur	Kernel: (3, 3); Sigma: (0.1, 1.0)
Gaussian Noise	Std: 0.02
Hue	$\pm$0.01
Posterize	5 bits
RGB Shift	Shift limit: 0.02
Saturation	(0.9, 1.1)
Median Blur	Kernel: (3, 3)
Box Blur	Kernel: (3, 3)
Motion Blur	Kernel: (3, 5); Angle: $\pm$25°; Direction: $\pm$0.25
Sharpness	0.5

Table 1. Augmentation Parameters while training MultiNeRF

C. Additional Details on the User Study

We conducted our user study on Amazon Mechanical Turk (MTurk), recruiting six unique participants. Each participant was presented with 180 image pairs in randomized order. In each pair, one image was the ground-truth image, and the other was the watermarked image output produced by one of the watermarking methods (e.g., MultiNeRF, WateRF, WateRF-modified, or NeRFProtector).

The participants were asked to "Compare the Ground-Truth (GT) image and the AI-processed image. and Does the processed image have blotches or color/rainbow artifacts compared to the GT image?" They then rated the overall severity of any artifacts on a five-point scale (Tab. 2).

Rating	Description
5	Severe artifacts that significantly impact image quality
4	Clearly visible artifacts
3	Some visible artifacts upon closer inspection
2	Barely noticeable artifacts
1	No visible artifacts

Table 2. Severity scale for AI-processed image artifacts

D. Augmentation-Intensity Ablation

To quantify the impact of augmentation strength and application probability on watermark persistence, we evalu-

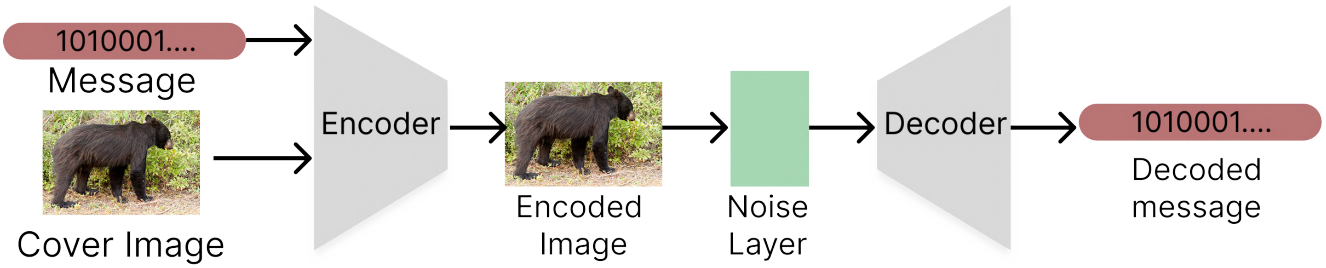

Figure 1. HiDDeN decoder

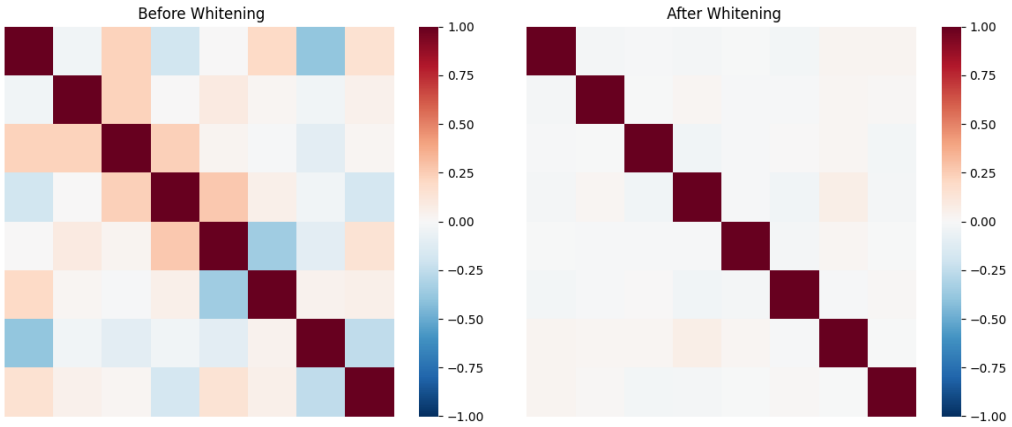

Figure 2. Covariance Matrix of the bit outputs before and after whitening

ated our best model under combinations of two simultaneous attacks drawn from the set defined in Table 3. Each augmentation was applied independently with probability p, and we swept three intensity levels—Low (L), Medium (M), and High (H)—as specified in Table 3. Results are summarized in Table 4, which reports mean bit-accuracy for $p \in \{0.5, 0.75, 1.0\}$ across all paired augmentations. We found that the Low intensity setting with $p = 0.75$ (i.e., each attack applied 75% of the time at the lowest parameter values) yielded the highest robustness, outperforming both more aggressive intensities and higher/lower application probabilities. Consequently, we adopt L($p = 0.75$) as our default augmentation configuration, balancing watermark fidelity with resistance to common image degradations.

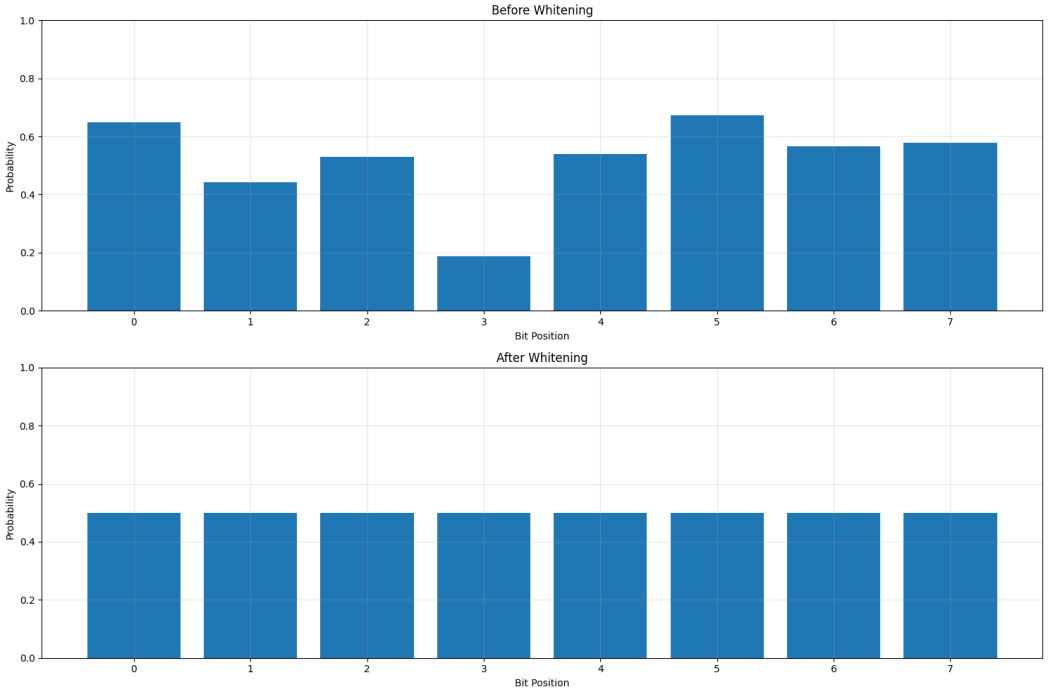

Figure 3. Probability of bit outputs from decoder before and after whitening

Attack	**Low**	**Med**	**High**
JPEG Compression	quality = 30	quality = 15	quality = 5
Brightness	(0.9, 1.1)	(0.75, 1.25)	(0.5, 1.5)
Contrast	(0.9, 1.1)	(0.75, 1.25)	(0.5, 1.5)
Color Jiggle	(0.05, 0.05, 0.05, 0.01)	(0.10, 0.10, 0.10, 0.02)	(0.10, 0.10, 0.10, 0.05)
Gaussian Blur	kernel = (3, 3), sigma = (0.1, 1.0)	kernel = (5, 5), sigma = (0.1, 1.5)	kernel = (7, 7), sigma = (0.1, 2.0)
Gaussian Noise	$\sigma = 0.02$	$\sigma = 0.04$	$\sigma = 0.08$
Hue	hue = 0.01	hue = 0.02	hue = 0.05
Posterize	bits = 5.0	bits = 4.0	bits = 3.0
RGB Shift	shift = 0.02	shift = 0.05	shift = 0.10
Saturation	(0.9, 1.1)	(0.75, 1.25)	(0.5, 1.5)
Median Blur	kernel = (3, 3)	kernel = (5, 5)	kernel = (7, 7)
Box Blur	kernel = (3, 3)	kernel = (5, 5)	kernel = (7, 7)
Motion Blur	kernel = (3, 5), angle = 25°, direction = 0.25	kernel = (3, 7), angle = 45°, direction = 0.50	kernel = (3, 9), angle = 90°, direction = 1.00
Sharpness	factor = 0.5	factor = 1.0	factor = 2.5

Table 3. Attack Intensities and Parameter Values

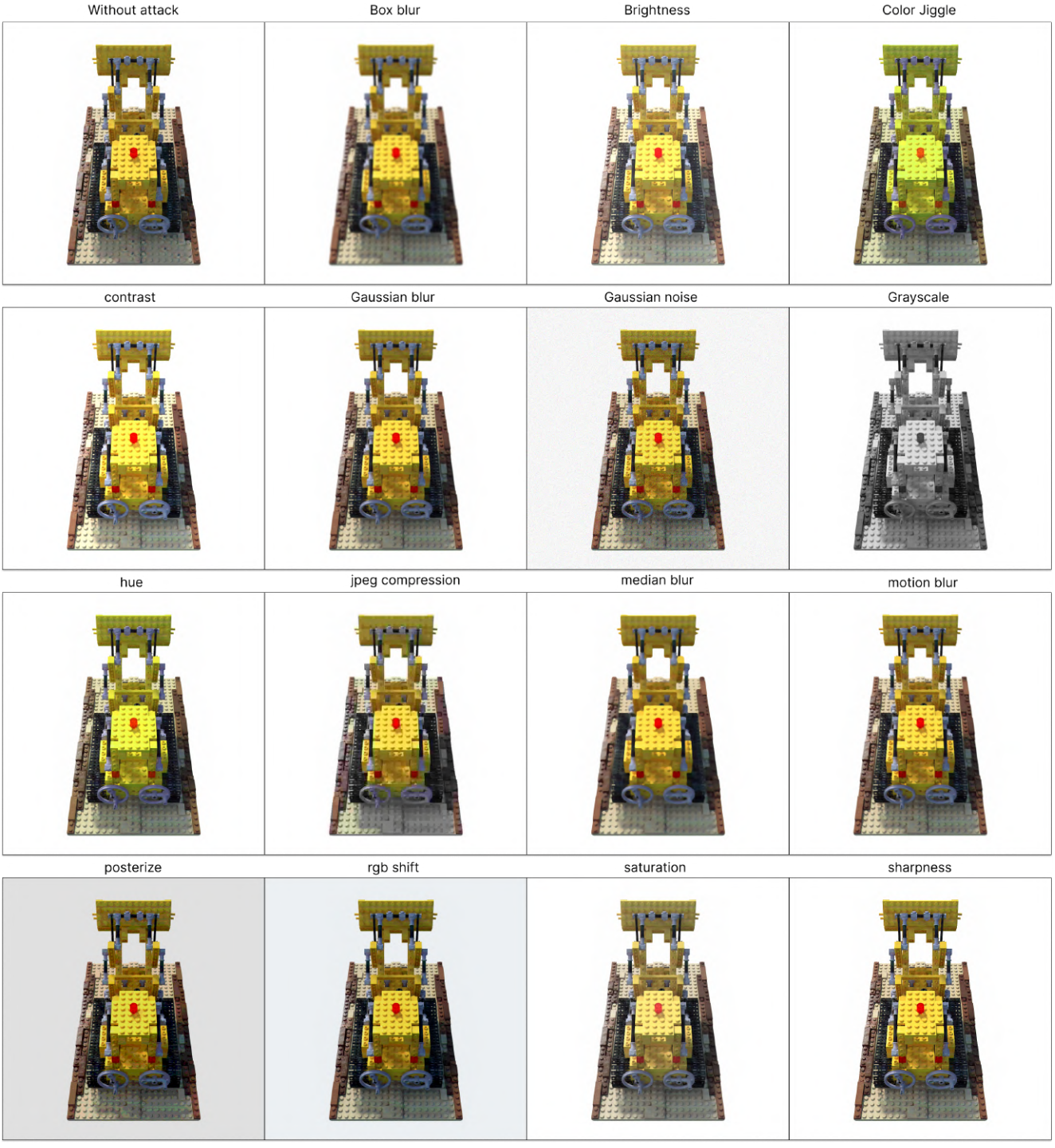

Figure 4. Examples of different augmentations applied during training

Attack	Normal	L (p=0.5)	M (p=0.5)	H (p=0.5)	L (p=0.75)	M (p=0.75)	H (p=0.75)	L (p=1.0)	M (p=1.0)	H (p=1.0)
NONE	**94.20**	94.39	94.33	92.74	**95.09**	94.26	92.74	94.98	94.39	92.43
Blur	**93.90 (L)**	94.30 (L)	94.31 (L)	92.68 (L)	**94.97**	94.16	93.86	95.01	94.38	92.52
	93.58 (M)	94.24 (M)	94.35 (M)	92.92 (M)	**94.77**	94.11	94.19	94.89	94.41	92.79
	92.90 (H)	94.06 (H)	94.39 (H)	92.81 (H)	**94.60**	94.14	94.23	94.58	94.46	92.84
Rotate	**78.18 (L)**	87.79 (L)	87.42 (L)	85.04 (L)	**88.24**	87.14	85.68	87.75	86.82	84.65
	54.58 (M)	77.14 (M)	77.27 (M)	75.06 (M)	**77.53**	76.92	75.22	78.03	76.42	75.10
	45.13 (H)	72.34 (H)	71.27 (H)	69.13 (H)	**72.51**	71.78	69.60	72.66	70.64	68.35
Crop	**89.89 (L)**	88.08 (L)	87.41 (L)	86.25 (L)	**88.19**	87.70	87.78	87.63	87.51	86.20
	82.28 (M)	82.85 (M)	81.33 (M)	80.48 (M)	**82.72**	82.11	81.49	82.77	82.27	80.60
	75.30 (H)	76.51 (H)	75.45 (H)	74.63 (H)	**77.15**	76.74	74.94	76.57	76.81	74.97
Resize	**91.58 (L)**	92.26 (L)	92.24 (L)	90.90 (L)	**92.86**	91.96	91.92	92.42	92.30	90.83
	81.61 (M)	87.78 (M)	87.98 (M)	86.41 (M)	**88.23**	87.75	87.72	87.74	88.01	86.55
	62.28 (H)	81.86 (H)	81.45 (H)	79.53 (H)	**81.63**	81.01	80.41	81.61	81.23	79.48
Noise	**94.11 (L)**	94.31 (L)	94.20 (L)	92.53 (L)	**94.82**	94.10	93.75	94.89	94.09	92.20
	94.44 (M)	79.90 (M)	78.75 (M)	76.94 (M)	**79.41**	81.08	81.01	80.57	79.73	80.26
	92.00 (H)	61.09 (H)	61.30 (H)	61.31 (H)	**62.40**	62.52	62.03	60.48	61.46	61.27
JPEG Compression	**90.39 (L)**	92.34 (L)	92.31 (L)	90.64 (L)	**92.59**	91.83	91.79	92.81	92.15	90.64
	85.11 (M)	91.54 (M)	91.13 (M)	89.48 (M)	**92.02**	90.95	90.55	91.75	91.07	88.70
	73.58 (H)	86.80 (H)	85.23 (H)	83.79 (H)	**87.35**	85.65	84.16	86.72	86.13	83.38
Contrast	**91.67 (L)**	92.42 (L)	92.13 (L)	90.36 (L)	**93.26**	92.58	92.31	93.16	92.34	91.11
	89.91 (M)	91.10 (M)	90.64 (M)	89.22 (M)	**91.99**	91.23	91.29	91.83	90.98	89.54
	88.32 (H)	89.63 (H)	89.20 (H)	88.05 (H)	**90.86**	90.22	89.97	90.85	89.86	88.29
Brightness	**66.33 (L)**	79.68 (L)	78.46 (L)	76.49 (L)	**79.32**	79.02	76.92	78.34	76.91	74.35
	71.63 (M)	83.71 (M)	82.66 (M)	80.93 (M)	**83.74**	82.56	80.71	82.59	80.94	78.46
	84.94 (H)	89.03 (H)	88.68 (H)	86.97 (H)	**89.10**	88.69	87.54	88.13	87.73	85.63
ColorJitter	**88.19 (L)**	89.17 (L)	88.58 (L)	86.68 (L)	**89.13**	88.97	88.36	89.39	88.42	87.16
	79.68 (M)	83.52 (M)	83.61 (M)	81.82 (M)	**84.01**	83.31	83.13	84.06	83.29	82.29
	72.04 (H)	79.41 (H)	78.85 (H)	77.40 (H)	**79.83**	79.30	78.71	79.34	78.81	77.56
Grayscale	**83.83 (L)**	89.72 (L)	89.32 (L)	87.49 (L)	**89.93**	89.34	88.24	89.19	89.08	87.33
	83.83 (M)	89.72 (M)	89.32 (M)	87.49 (M)	**89.93**	89.34	88.24	89.19	89.08	87.33
	83.83 (H)	89.72 (H)	89.32 (H)	87.49 (H)	**89.93**	89.34	88.24	89.19	89.08	87.33
Hue	**90.75 (L)**	90.82 (L)	90.46 (L)	88.90 (L)	**91.33**	91.03	90.31	91.34	90.54	88.69
	85.69 (M)	84.80 (M)	83.94 (M)	83.00 (M)	**84.68**	84.04	84.02	84.38	83.32	82.35
	81.70 (H)	81.10 (H)	79.82 (H)	78.58 (H)	**81.09**	80.79	80.53	80.58	79.99	79.03
RGB Shift	**93.21 (L)**	93.11 (L)	92.72 (L)	90.75 (L)	**93.55**	92.53	91.92	93.75	93.03	91.07
	90.02 (M)	90.44 (M)	90.22 (M)	88.83 (M)	**91.11**	90.29	90.02	90.96	90.78	89.27
	70.80 (H)	82.89 (H)	82.79 (H)	82.20 (H)	**83.90**	83.49	83.29	83.46	83.20	82.01
Motion Blur	**86.16 (L)**	91.27 (L)	90.93 (L)	89.66 (L)	**92.17**	91.41	90.74	91.91	91.51	90.03
	78.35 (M)	86.94 (M)	86.56 (M)	85.28 (M)	**87.35**	86.74	86.32	86.81	86.53	85.35
	72.22 (H)	82.22 (H)	82.04 (H)	80.91 (H)	**82.86**	82.92	81.82	82.68	82.44	80.99

Table 4. Comparison of Model Robustness against Different Attacks with varying Intensity Levels (p=0.5, 0.75, and 1.0)

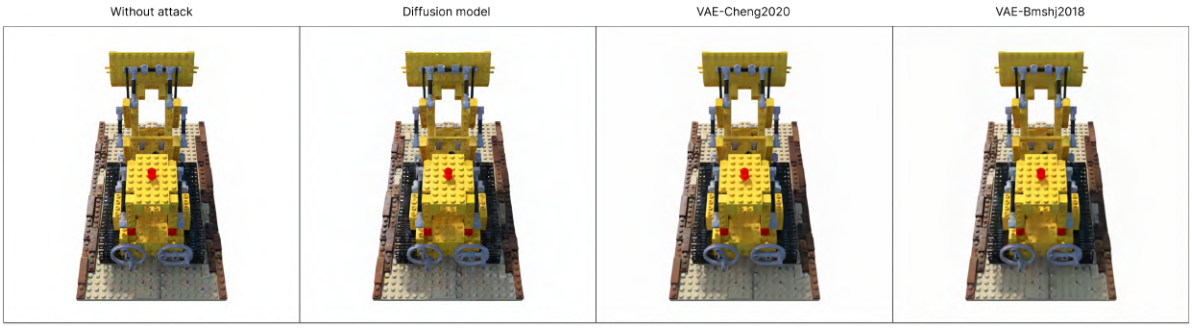

Figure 5. Regeneration attacks

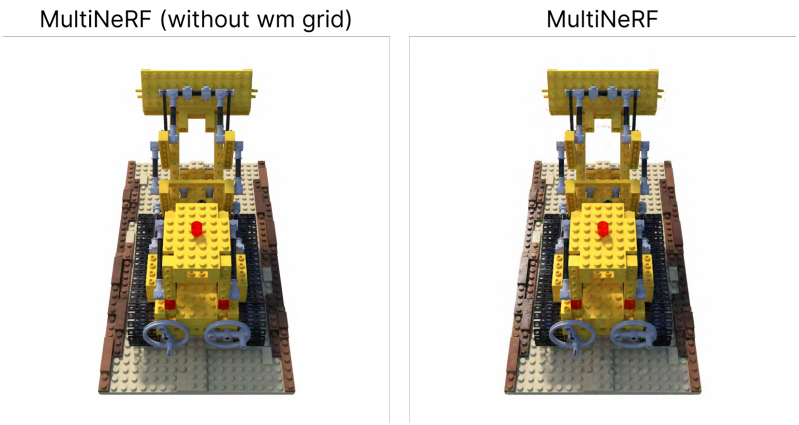

Figure 6. MultiNeRF (without watermark grid) vs MultiNeRF (Full method) image quality. Without the grid, we see that we mostly lose reflection information.

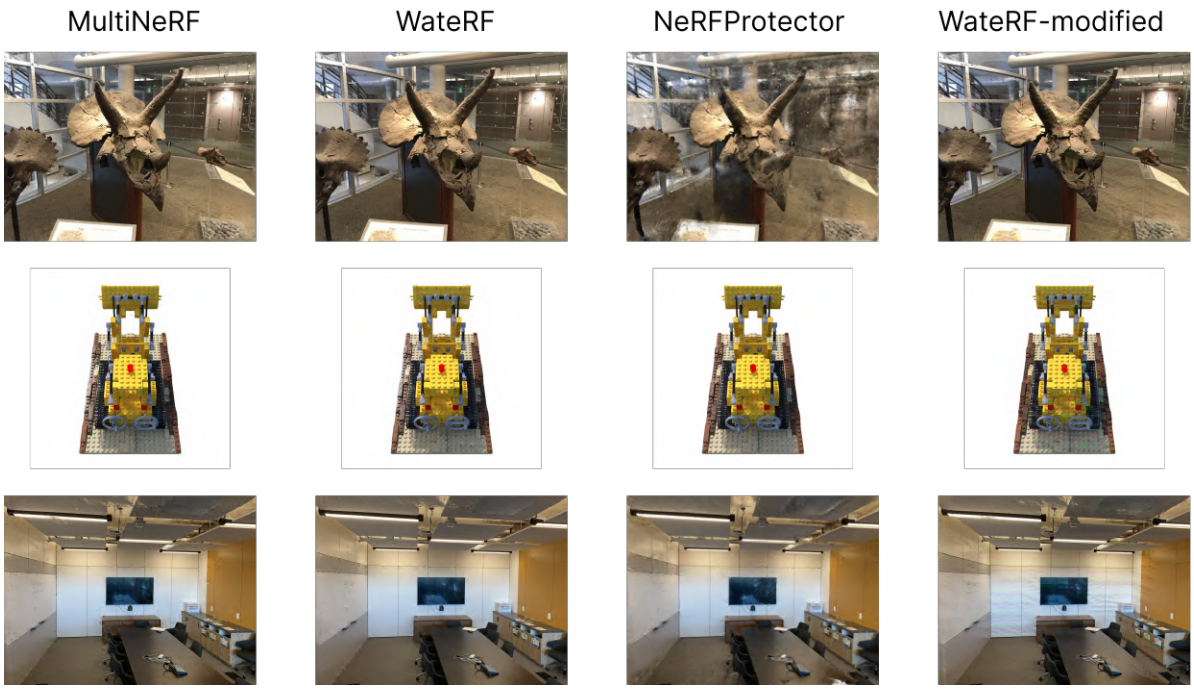

Figure 7. Examples of the images used for user study

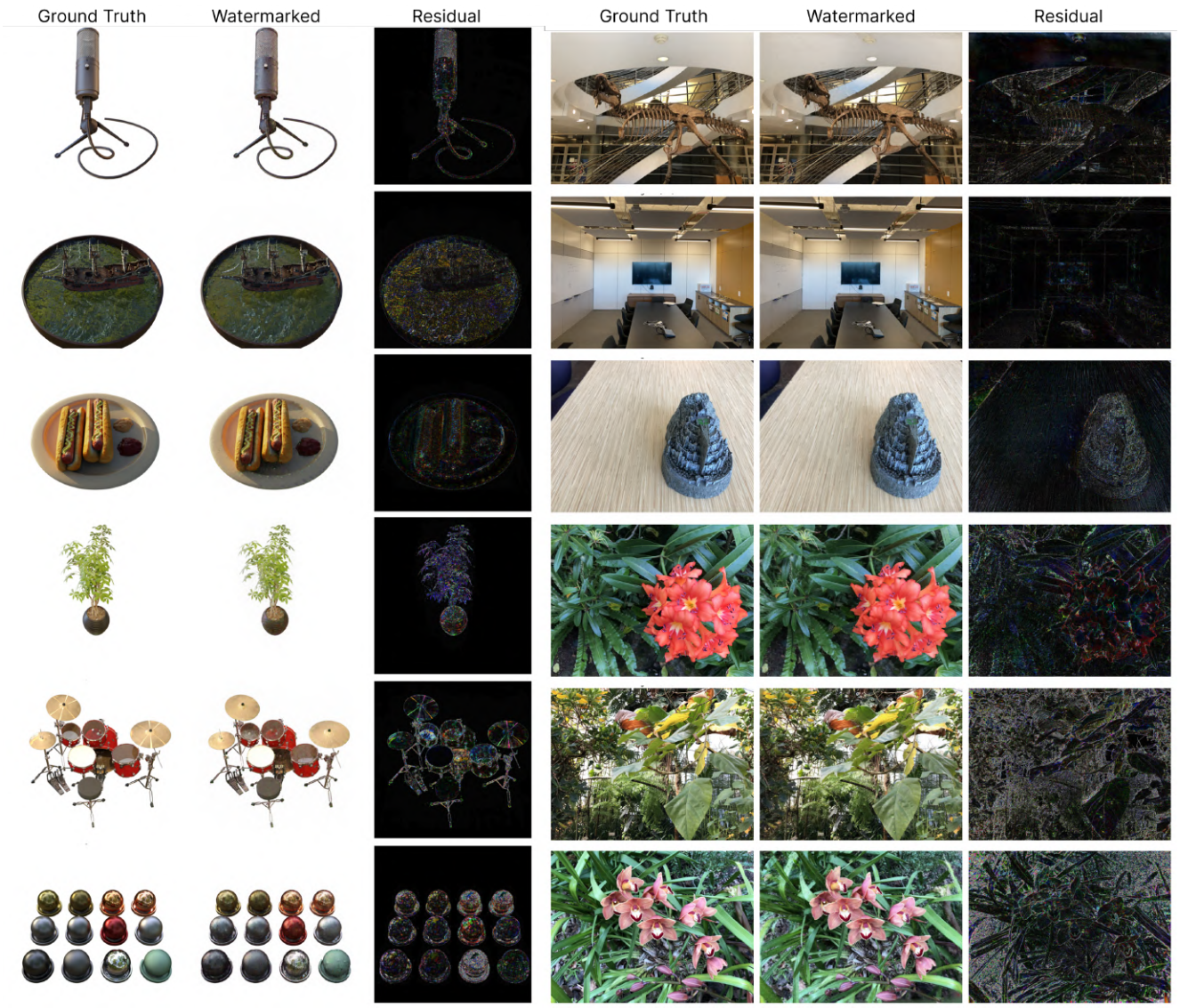

Figure 8. MultiNeRF results on the other synthetic and LLFF dataset

References

[1] 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

[2] 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

[3] Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3674–3683, 2020. 1

[4] Richard Shin and Dawn Song. Jpeg-resistant adversarial images. In *NeurIPS W. ML Comp. Security*, page 8, 2017. 1

[5] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019. 1

[6] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *Proc. ECCV*, pages 657–672, 2018. 1