SpecTRe-GS: Modeling Highly Specular Surfaces with Reflected Nearby Objects by Tracing Rays in 3D Gaussian Splatting

Supplementary Material

Jiajun Tang[1,2] Fan Fei[1,2] Zhihao Li[3] Xiao Tang[3] Shiyong Liu[3]
Youyu Chen[4] Binxiao Huang[5] Zhenyu Chen[3] Xiaofei Wu[3] Boxin Shi[1,2#]
[1]State Key Laboratory for Multimedia Information Processing, School of Computer Science, Peking University
[2]National Engineering Research Center of Visual Technology, School of Computer Science, Peking University
[3]Huawei Noah's Ark Lab [4]Harbin Institute of Technology [5]University of Hong Kong

In this supplementary material, we provide additional implementation details for SpecTRe-GS (Sec. 6), the creation details of synthetic and real-world data (Sec. 7), and further qualitative/quantitative results and experimental analyses (Sec. 8). The videos on the project page[1] showcase qualitative view interpolation and scene editing results.

6. Additional Implementation Details

6.1. Implementation of Ray Tracing

The ray tracer described in Sec. 3.3 in the main paper is implemented in OptiX 7 [7] and integrated into the 3DGS framework with the PyOptiX package as the Python bindings for the OptiX host API. According to the OptiX 7 specification, the OptiX pipeline consists of user-programmable entry points (programs) for different stages in ray tracing, we implement ray tracing in the Gaussians point cloud with custom *ray-gen* and *any-hit* programs:

- we initialize ray origin and direction, set and read the per-ray payloads, evaluate responses of Gaussians, and aggregate the volumetric radiance in the *ray-gen* program;
- we calculate the precise hit point of the mesh-bounded Gaussians and store the hit information into the per-ray buffer in the *any-hit* program.

The programs are described in Proc. 1 and Proc. 2. In our implementation, we construct a max heap of size $k = 256$ to store the closest hits with the complexity of $\mathcal{O}(N \log k)$, where N is the total number of hits.

6.2. Alignment of Rasteriazion and Ray Tracing

We align the radiance aggregation process of our ray tracer to that of the rasterizer. We use the rasterizer in GOF [14] as GOF calculates the maximum response of Gaussians along the rays. However, GOF still uses projected center depths to sort Gaussians in volumetric rendering. Therefore, we deliberately calculate projected depths as $t_{\text{hit},i}$ in our ray tracer and use $t_{\text{hit},i}$ instead of $t_{\max,i}$ for the ordering in Eq. (1). We also use the same ray termination threshold of remaining transmittance $T_{\min} = 0.001$ and max number of contributing Gaussians $K_{\max} = 256$ for our rasterizer and ray tracer. This reduces the discrepancy between the rendered appearances of the same Gaussian point cloud by these two renderers, which would cause the inconsistency between the directly observed appearance of objects and their appearance through a highly specular surface.

Procedure 1: *Ray-gen* Program

Input: ray origin $\boldsymbol{o}$, ray direction $\boldsymbol{d}$, GAS handle $\mathcal{H}$, Gaussians $\{\mathcal{G}_i\}$, min transmittance $T_{\min}$, min contribution $\alpha_{\min}$, hit buffer size k, min ray distance t_{near}, max ray distance t_{far}

Output: ray incident radiance I_{ind}, ray visibility V_{i}, ray depth D_{ind}

$I_{\text{ind}} \leftarrow (0, 0, 0)$;
$V_{\text{i}} \leftarrow 1$;
$D_{\text{ind}} \leftarrow 0$;
$t_{\text{curr}} \leftarrow t_{\text{near}}$;
while $t_{\text{curr}} < t_{\text{far}}$ **and** $V_{\text{i}} > T_{\min}$ **do**
  $c \leftarrow 0$;
  $\mathcal{B} \leftarrow \text{buffer}(k)$;
  $\text{setPayload}(\mathcal{B}, c)$;
  $\text{traceRay}(\mathcal{H}, \boldsymbol{o} + t_{\text{curr}}\boldsymbol{d}, \boldsymbol{d}, k)$;
  $\mathcal{B}, c \leftarrow \text{getPayload}()$;
  if $c = 0$ **then**
    $\text{terminateRay}()$;
  end
  $\mathcal{B} \leftarrow \text{sort}(\mathcal{B})$;
  for (t_{hit}, i) **in** $\mathcal{B}$ **do**
    $t_{\max}, \alpha_{\text{hit}} \leftarrow \text{response}(\boldsymbol{\mu}_i, \boldsymbol{s}_i, \boldsymbol{q}_i, \boldsymbol{o}, \boldsymbol{d})$;
    if $\alpha_{\text{hit}} > \alpha_{\min}$ **then**
      $\boldsymbol{c}_i = \text{SH}(\boldsymbol{\varphi}_i, \boldsymbol{d})$;
      $I_{\text{ind}} \leftarrow I_{\text{ind}} + \alpha_{\text{hit}} V_{\text{i}} \boldsymbol{c}_i$;
      $D_{\text{ind}} \leftarrow D_{\text{ind}} + \alpha_{\text{hit}} V_{\text{i}} t_{\max}$;
      $V_{\text{i}} \leftarrow (1 - \alpha_{\text{hit}}) V_{\text{i}}$;
    end
    $t_{\text{curr}} \leftarrow t_{\text{hit}}$;
  end
end

The work was done during an internship in Huawei.
[#]Corresponding author. E-mail: shiboxin@pku.edu.cn.

[1]https://spectre-gs.github.io/

Procedure 2: *Any-hit* Program

Input: ray origin $\boldsymbol{o}$, ray direction $\boldsymbol{d}$, Gaussians $\{\mathcal{G}_i\}$, hitted primitive index i, hit buffer $\mathcal{B}$, hit buffer size k, hit count c

Output: in-place modified hit buffer $\mathcal{B}$, hit count c

```
t_hit ← projectDepth(μ_i, o, d);
h ← (t_hit, i);
if c = k then
    h_max ← B.popMax();
else
    h_max ← (+∞, −1));
    c ← c + 1;
end
h_new ← findCloser(h_max, h);
B.insert(h_new);
```

6.3. Training Details

During training, we use the same color reconstruction loss as commonly adopted in 3DGS-based methods [5]:

$$\mathcal{L}_{\mathrm{c}} = 0.8 \cdot \frac{1}{|I_{\mathrm{GT}}|} \sum ||I_{\mathrm{GT}} - I||_1 - 0.2 \cdot \mathrm{SSIM}(I_{\mathrm{GT}}, I), \tag{14}$$

and we also apply this loss to I_{ss} in later training steps (>15k iterations) to encourage physics-based modeling of highly reflective regions.

We follow Eq. (1) to compute the mean depth of contributing Gaussians as the surface depth, instead of the depth of the "median" Gaussian in GOF [14], which we find is generally more noisy and inefficient in utilizing gradient signals. In the first 4k steps, we only rely on SH color modeling to quickly get a rough geometry initialization and low-frequency view-dependent radiance modeling.

Since the Fresnel reflectance is calculated from approximation (Eq. (4)), we detach $\partial A_{\mathrm{spec}}/\partial \boldsymbol{n}$ and clip A_{spec} as $\min(A_{\mathrm{spec}}, 10F_0)$ to ensure numerical stability.

We only run StableNormal [12] once for all scenes and save the estimated normals as monocular normal priors.

6.4. Tone Mapping

Our method operates in linear color space as required by physically-based rendering. We assume the gamma-corrected sRGB space of $\gamma = 2.2$ is usually used in the input images, which is closer to human perception. Thus, we can convert the images into linear color space by inversely applying the gamma correction. We convert our results back to the commonly adopted gamma-corrected sRGB space with $\gamma = 2.2$ prior to visualization or the computation of photometric losses and error metrics.

7. Data Creation Details

7.1. Synthetic Scenes

We collect 6 synthetic scenes using the Blender Cycles engine [3]: HELMET, MARBLETABLE, VASE, POT, TOASTER, and MIRROR, as described in Sec. 4.2 in the main paper. We show example images of each scene in Tab. 4 and Tab. 5 of this document.

7.2. Real-world Scenes

For real-world scenes, we use a hand-held iPhone 15 Pro and the "ProCam" app to take raw images with a linear camera response. We fix the white balance, focal length, exposure time, and ISO for all images in the same scene. We register the camera poses using COLMAP [8, 9] with SuperPoint [2] for feature extraction and LightGlue [6] for matching. After obtaining the captured raw images of the scene, we use a custom image signal processor (ISP) to process the raw image by, *e.g*., demosaicking, white balancing, transforming color space, and most importantly, applying a tone mapping with $\gamma = 2.2$ to let the processed images satisfy our assumption of availability of linear space images. We resize the images to 1440×1080 and remove the out-of-focus background regions. The highly reflective regions are manually marked. By doing so, we collect 2 real-world scenes: REALBOWL and REALPOT. We show example images of each scene in Tab. 5 of this document.

8. Additional Results

8.1. Results with Varying Roughness

Our method is designed for perfect mirror reflections. Nevertheless, the inclusion of a low-frequency component gives it the ability to model rougher surfaces to some extent. Fig. 7 shows its results on the HELMET scene with varying roughness, from highly smooth ($\rho = 0.05$) to medium rough ($\rho = 0.3$). For each roughness value, we show the rendered image and the ground truth image in a test view, alongside the decomposition of specular component $A_{\mathrm{spec}} I_{\mathrm{i}} \odot (1 - I_{\mathrm{r}})$ and low-frequency component $I_{\mathrm{diff}} \odot (1 - I_{\mathrm{r}}) + I_{\mathrm{rs}} \odot I_{\mathrm{r}}$, expanded according to the modeling in Sec. 3.2 and soft mask I_{r} in Sec. 3.4. As the roughness increases, while direct specular reflections can be approximated by blurred environment maps, indirect specular reflections in the lower half of the helmet are mimicked by brighter I_{rs} with lower A_{spec} values.

8.2. Geometry Representation

As shown in Fig. 8, our method can better capture the planar surface in MARBLETABLE scene with most points well aligned to the object, benefitting from our normal prior guidance and joint optimization of incident radiance and geometry. Without accurate geometry optimization and inci-

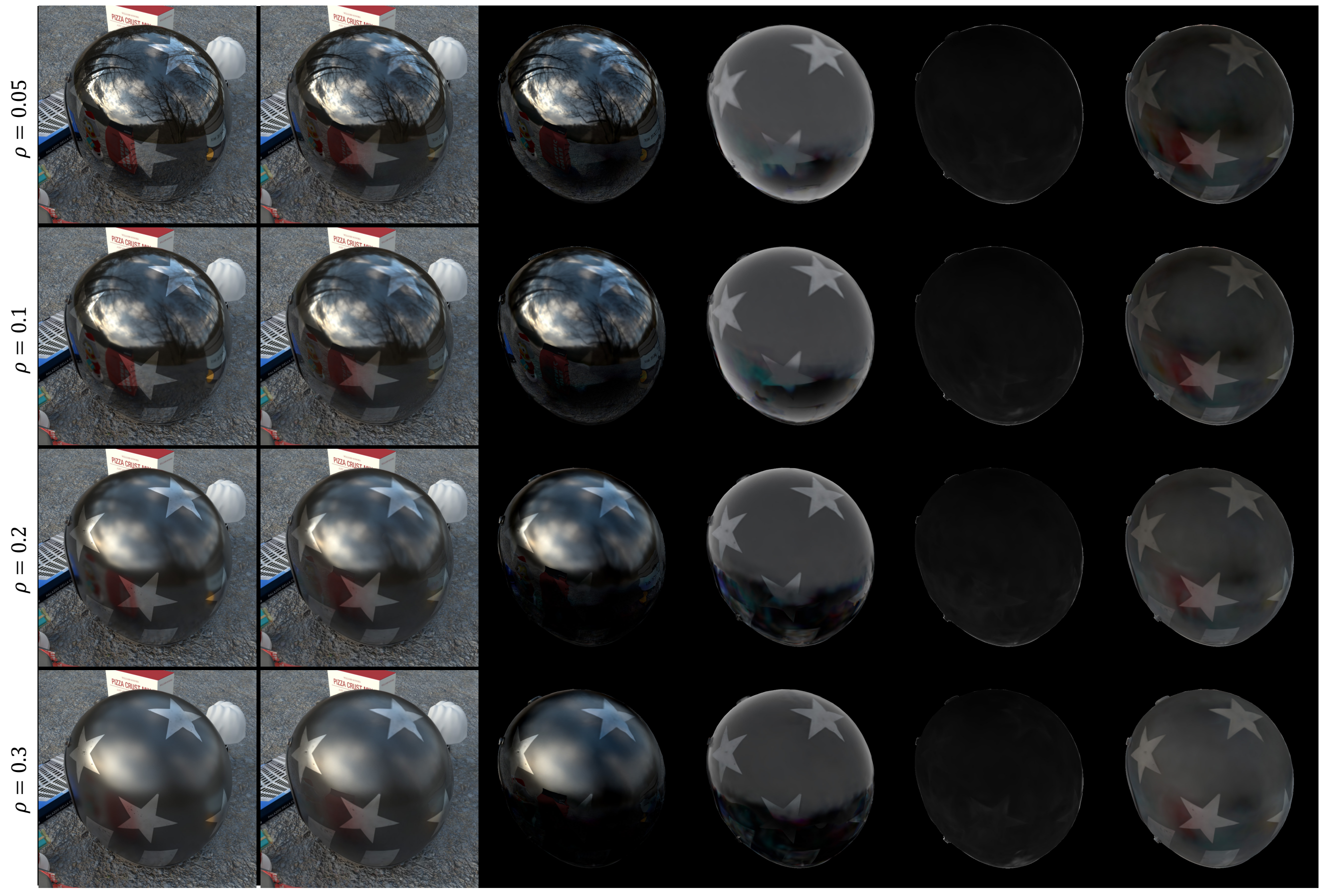


Figure 7. As surface roughness increases, our method attributes more proportion of the scene appearance to low-frequency reflection instead of perfect mirror reflection.

dent radiance reconstruction, other methods tend to fit high-frequency view-dependent specular reflection with high-frequency floaters.

8.3. Qualitative Ablation Results

Fig. 9 shows the results of the ablated variants of our method mentioned in Sec. 4.4 in the same view as Fig. 4. When the monocular normal prior guidance is absent during early training stages (Ours w/o **N**.), the training loss terms tend to overemphasize color reconstruction fidelity in observed images, causing the scene representation to converge to local minima with geometry deviating from ground truth in highly specular regions (as shown by the translucent artifacts on the left side of the helmet in column 1, indicating incomplete underlying geometric reconstruction). Conversely, when relying solely on monocular normal priors without subsequent joint optimization to refine scene geometry (Ours w/o **J**.), the inherent inaccuracies and multi-view inconsistencies in monocular normal predictions prevent the reconstruction of precise geometry required for physics-based high-frequency reflection modeling (evidenced by the missing high-frequency details in the reflections on the helmet surface, as depicted in column 2). As previously discussed regarding physics-based rendering approaches, modeling only direct illumination (Ours w/o **I**.) leads to indirect lighting effects being baked into either SH color representations or diffuse albedo, thereby compromising high-frequency component quality (manifested as missing high-frequency details and ghosting artifacts in the lower helmet region due to inter-reflections, shown in column 3). The progressive learning scheme (Ours w/o **P**.) and depth-aware ray perturbation (Ours w/o **D**.) also significantly contribute to faithful reconstruction of view-dependent high-frequency specular reflections.

8.4. Alignment of Rendering Methods

We analyze the consistency of rendering results from our rasterizer and ray tracer on the STUMP scene of the Mip-NeRF 360 dataset [1], which is a prerequisite for accurately evaluating indirect incident radiance. We show the rendered images and the corresponding PSNR scores of each renderer in Fig. 10, accompanied by the error map visualiza-

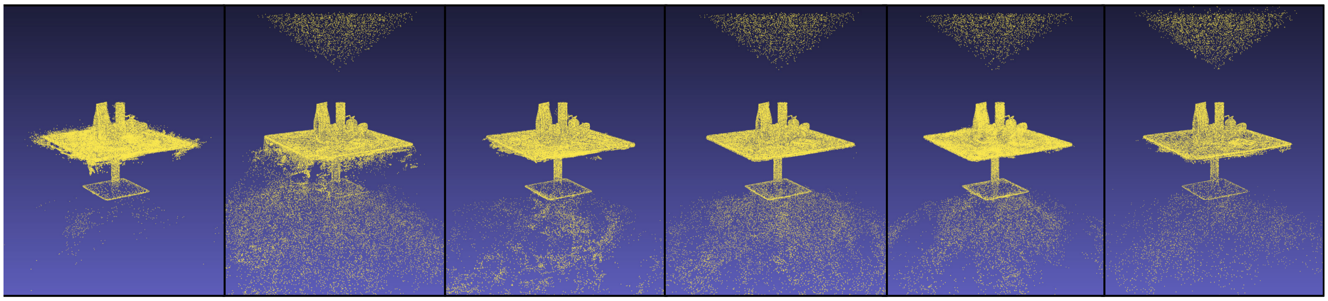


Figure 8. Visualization of the point cloud reconstructed by comparing methods [4, 5, 10, 13, 14]. Ours more faithfully captures the underlying geometry of reflective regions, while other methods disrupt their geometry to imitate highly specular reflections.

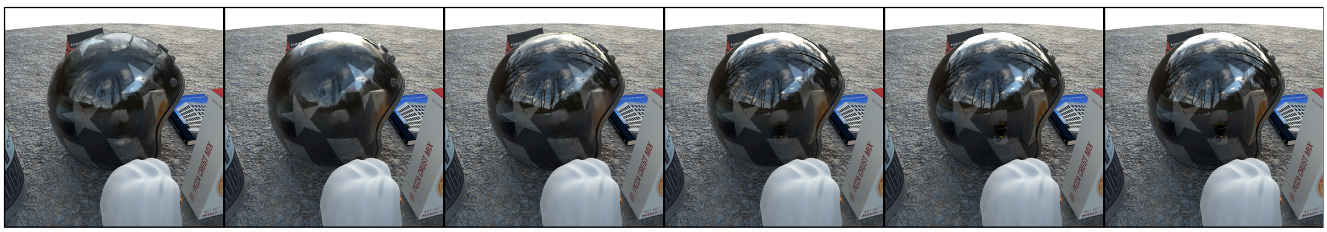


Figure 9. Qualitative ablation results with variants of our proposed method excluding: **N**ormal prior guidance, **J**oint geometry optimization, **I**ndirect incident lighting modeling, **P**rogressive learning, and **D**epth-aware ray perturbation.

Figure 10. Rasterized and ray-traced results in our proposed method are highly consistent, which ensures accurate indirect incident radiance queries from the Gaussian point cloud shared by our rasterizer and ray tracer.

tion. The rendering results from those two renderers in our pipeline remain highly consistent, as indicated by the inconspicuous visual difference, the close PSNR scores, and the colors in the error map.

8.5. More Quantitative Results

We show detailed quantitative results on each scene in Tab. 4 and Tab. 5 of this document. In general, our SpecTRe-GS consistently outperforms most compared methods on both synthetic scenes and real-world scenes, especially within reflective regions.

8.6. More Qualitative Results

We show additional qualitative comparisons with the baseline methods[2] on each scene in Fig. 11-14 of this document. For each scene, we show comprehensive visual comparison results from multiple test views.

[2]We show results of GShader* for all scenes, 3DGS-DR* for HELMET scene as their better performance indicated by quantitative evaluations.

We provide videos of view interpolation results as attached files on the project page. Compared with baseline methods, our method gives more view-consistent renderings of high-frequency reflection, which better respects the geometry of the highly reflective surfaces. In addition, we provide videos of the scene editing results.

Table 4. Quantitative comparison results with state-of-the-art methods on each of the 4 synthetic scenes (HELMET, MARBLETABLE, VASE, and POT). We show example images and the dataset splits of each scene in the leftmost column. We report the scores of PSNR, SSIM [11], and LPIPS [15] for entire images and within reflective regions. We mark the **best** and the second best results in each column. ↑ (↓) means higher (lower) is better.

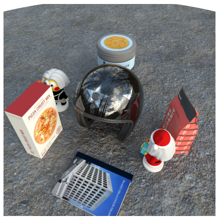
Scene	Method	Entire Image			Reflective Region		
		PSNR↓	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
HELMET (train: 201, test:100)	3DGS [5]	27.92	0.881	0.157	21.62	0.918	0.090
	GOF [14]	28.28	0.893	0.130	21.62	0.918	0.086
	GOF* [14]	27.15	0.887	0.145	20.28	0.912	0.100
	3iGS [10]	28.33	0.883	0.152	22.06	0.919	0.088
	GShader [4]	25.80	0.836	0.204	20.72	0.913	0.098
	GShader* [4]	26.51	0.846	0.196	21.28	0.915	0.097
	3DGS-DR [13]	26.32	0.841	0.233	20.76	0.914	0.098
	3DGS-DR* [13]	27.15	0.845	0.226	22.36	0.927	0.083
	Ours	**29.90**	**0.914**	**0.112**	**24.05**	**0.944**	**0.056**
MARBLETABLE (train: 233, test:123)	3DGS [5]	22.41	0.858	0.197	20.10	0.889	0.142
	GOF [14]	23.59	0.873	**0.177**	19.77	0.889	**0.137**
	GOF* [14]	24.14	0.870	0.187	19.71	0.884	0.150
	3iGS [10]	24.42	0.865	0.184	20.09	0.880	0.145
	GShader [4]	24.28	0.856	0.209	20.65	0.878	0.160
	GShader* [4]	24.89	0.865	0.198	21.27	0.886	0.152
	3DGS-DR [13]	25.37	0.857	0.223	22.02	0.882	0.164
	3DGS-DR* [13]	22.51	0.837	0.239	19.63	0.866	0.180
	Ours	**26.89**	**0.875**	0.183	**22.38**	**0.890**	0.142
VASE (train: 201, test:100)	3DGS [5]	33.16	0.944	0.093	26.42	0.975	0.042
	GOF [14]	**33.28**	0.948	0.083	26.36	0.975	0.040
	GOF* [14]	32.72	0.945	0.087	25.32	0.973	0.045
	3iGS [10]	33.02	0.943	0.091	26.60	0.975	0.042
	GShader [4]	30.33	0.912	0.129	25.14	0.973	0.046
	GShader* [4]	30.79	0.919	0.121	25.44	0.973	0.046
	3DGS-DR [13]	31.35	0.914	0.149	25.86	0.973	0.046
	3DGS-DR* [13]	30.94	0.912	0.152	25.10	0.971	0.049
	Ours	33.14	**0.949**	**0.076**	**27.20**	**0.982**	**0.027**
POT (train: 201, test:100)	3DGS [5]	29.88	0.923	0.096	23.50	0.945	0.062
	GOF [14]	29.71	0.921	0.093	23.41	0.943	0.058
	GOF* [14]	29.04	0.919	0.105	22.22	0.940	0.071
	3iGS [10]	**31.13**	0.928	0.090	23.88	0.947	0.059
	GShader [4]	29.37	0.913	0.116	22.53	0.942	0.068
	GShader* [4]	29.53	0.915	0.114	22.86	0.943	0.066
	3DGS-DR [13]	30.22	0.917	0.115	23.34	0.944	0.066
	3DGS-DR* [13]	28.87	0.909	0.125	22.23	0.940	0.074
	Ours	30.30	**0.933**	**0.075**	**24.79**	**0.959**	**0.035**

Table 5. Quantitative comparison results with state-of-the-art methods on each of the 2 synthetic scenes (TOASTER and MIRROR) and 2 real-world scenes (REALBOWL, and REALPOT). We show example images and the dataset splits of each scene in the leftmost column. We report the scores of PSNR, SSIM [11], and LPIPS [15] for entire images and within reflective regions. We mark the **best** and the second best results in each column. ↑ (↓) means higher (lower) is better.

Scene	Method	Entire Image			Reflective Region		
		PSNR↓	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
TOASTER	3DGS [5]	26.26	0.914	0.123	18.77	0.951	0.060
(train: 201, test:100)	GOF [14]	26.34	**0.924**	**0.103**	18.74	0.951	0.060
	GOF* [14]	25.61	0.918	0.115	17.99	0.945	0.070
	3iGS [10]	26.71	0.914	0.120	19.34	0.951	**0.056**
	GShader [4]	25.51	0.897	0.146	18.16	0.944	0.070
	GShader* [4]	26.07	0.900	0.143	18.77	0.946	0.068
	3DGS-DR [13]	25.68	0.885	0.175	18.42	0.949	0.064
	3DGS-DR* [13]	25.97	0.871	0.199	18.95	0.947	0.069
	Ours	**27.73**	0.918	0.115	**20.39**	**0.953**	0.062
MIRROR	3DGS [5]	26.65	0.938	0.120	18.65	0.963	0.072
(train: 201, test:100)	GOF [14]	26.65	**0.941**	0.112	18.37	0.962	0.074
	GOF* [14]	26.74	**0.941**	0.113	18.52	0.963	0.074
	3iGS [10]	27.86	0.939	0.115	19.90	**0.964**	0.067
	GShader [4]	24.61	0.913	0.156	17.52	0.961	0.075
	GShader* [4]	25.02	0.917	0.149	17.71	0.962	0.074
	3DGS-DR [13]	26.77	0.924	0.145	19.04	**0.964**	0.069
	3DGS-DR* [13]	26.40	0.920	0.152	18.83	0.963	0.075
	Ours	**28.64**	0.938	**0.097**	**20.66**	0.963	**0.056**
REALBOWL	3DGS [5]	25.75	0.832	0.212	20.73	0.967	0.041
(train: 120, test:18)	GOF [14]	25.79	0.835	0.202	20.71	0.966	0.039
	GOF* [14]	25.59	0.834	0.205	19.81	0.965	0.043
	3iGS [10]	25.34	0.819	0.216	20.80	0.967	0.040
	GShader [4]	24.76	0.817	0.240	19.68	0.966	0.045
	GShader* [4]	24.82	0.819	0.239	19.82	0.966	0.045
	3DGS-DR [13]	25.66	0.832	0.227	20.58	0.967	0.042
	3DGS-DR* [13]	25.48	0.830	0.236	19.86	0.966	0.045
	Ours	**26.16**	**0.839**	**0.195**	**22.84**	**0.973**	**0.026**
REALPOT	3DGS [5]	23.89	0.814	0.245	21.89	**0.962**	0.056
(train: 121, test:18)	GOF [14]	23.94	**0.817**	0.235	21.78	0.960	0.055
	GOF* [14]	23.80	0.816	0.240	21.07	0.959	0.059
	3iGS [10]	23.40	0.799	0.249	21.63	0.960	0.055
	GShader [4]	23.11	0.802	0.271	20.77	0.960	0.061
	GShader* [4]	23.23	0.806	0.267	20.89	0.961	0.060
	3DGS-DR [13]	23.91	0.816	0.258	21.78	**0.962**	0.058
	3DGS-DR* [13]	23.71	0.815	0.266	20.90	0.961	0.064
	Ours	**24.06**	0.816	**0.230**	**22.69**	**0.962**	**0.044**

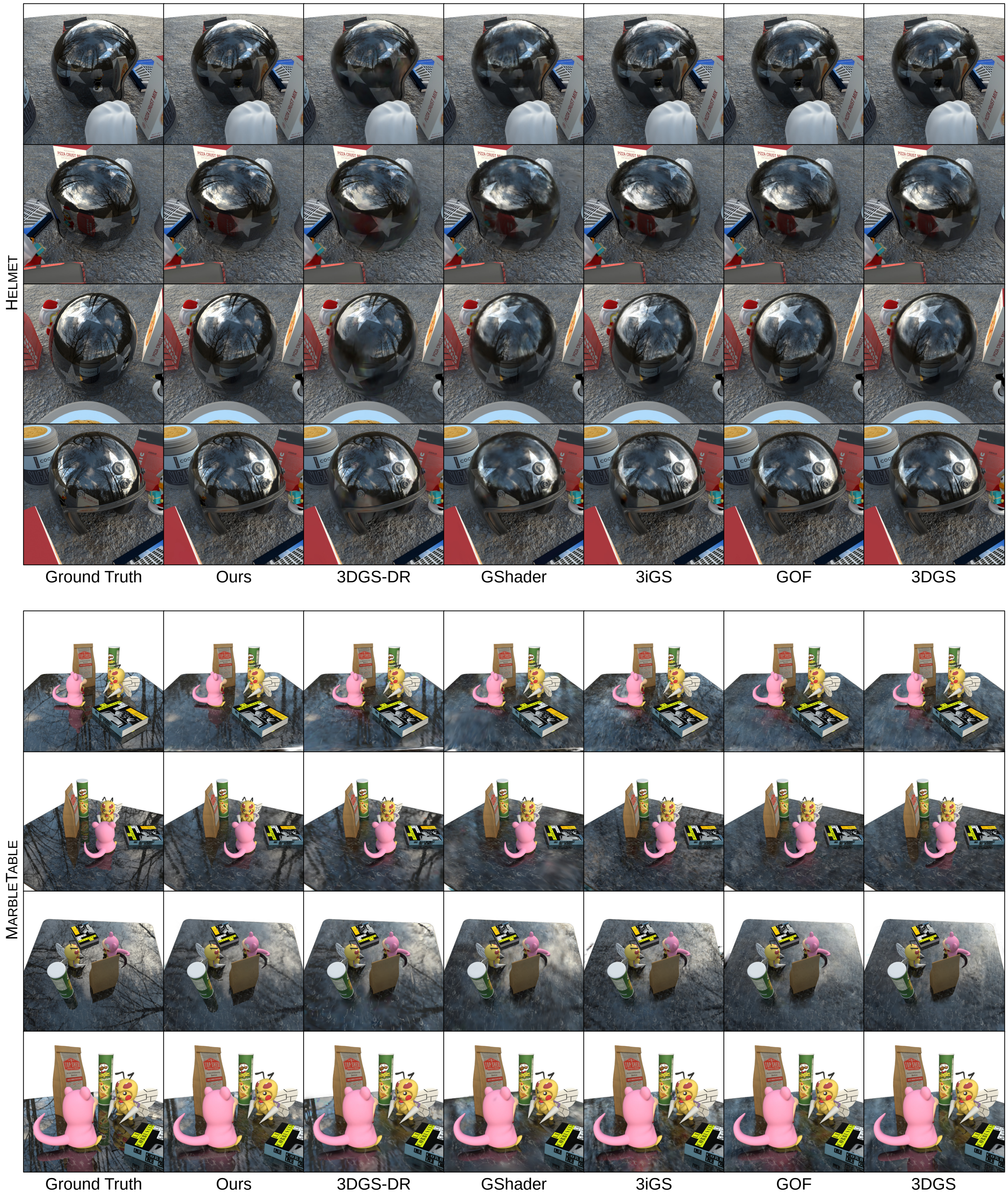


Figure 11. Comparison with state-of-the-art methods on two synthetic scenes: HELMET and MARBLETABLE.

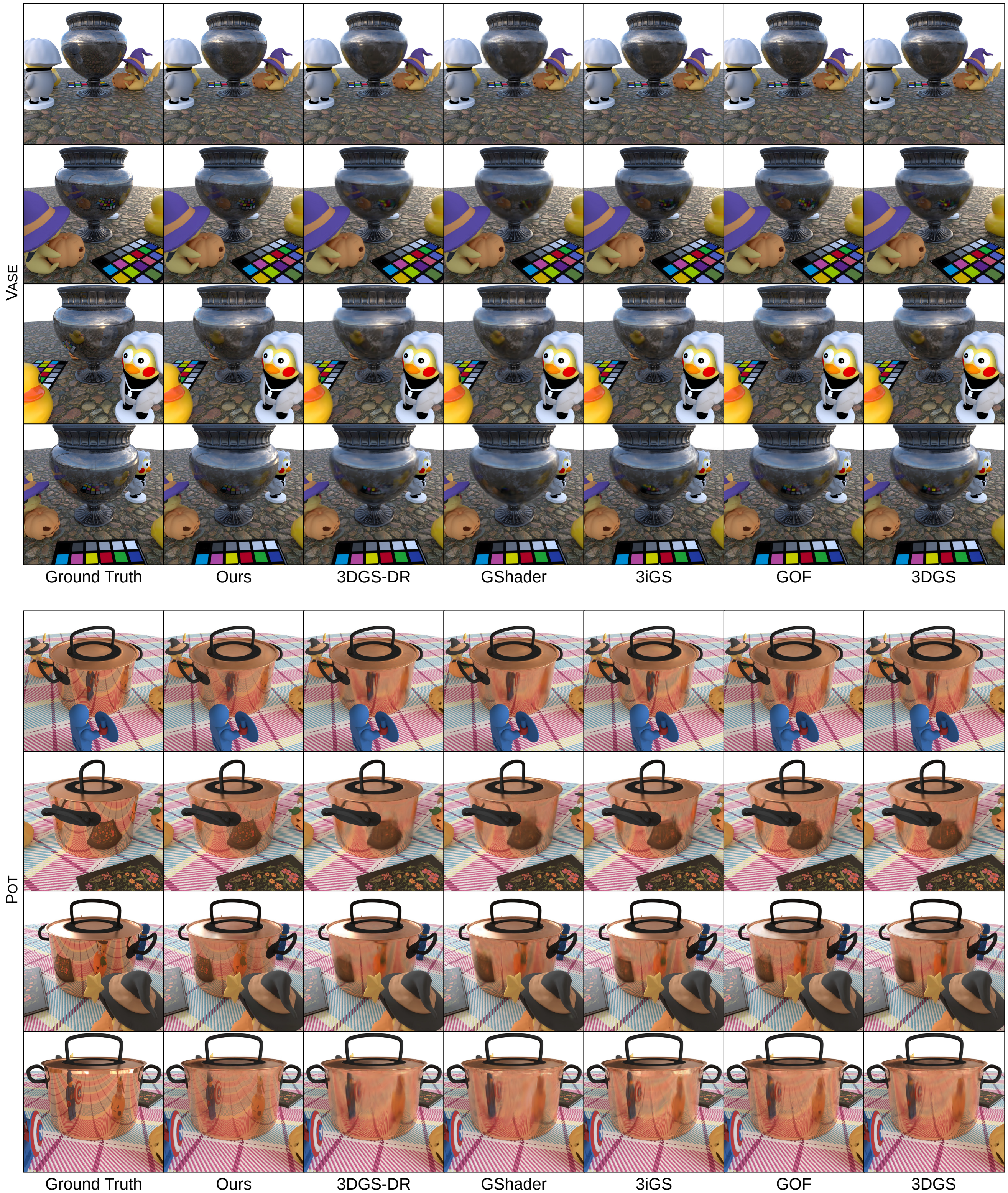


Figure 12. Comparison with state-of-the-art methods on two synthetic scenes: VASE and POT.

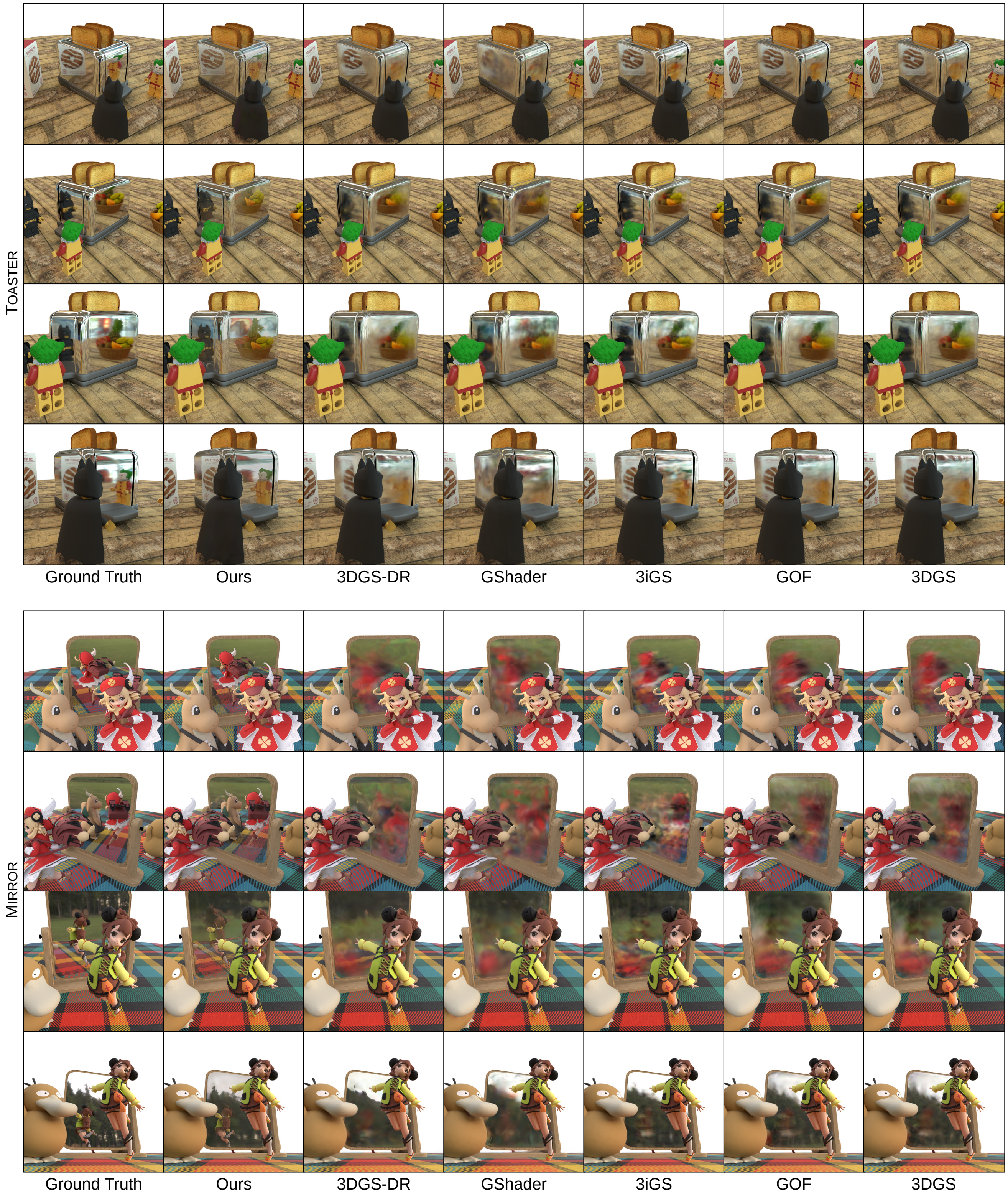


Figure 13. Comparison with state-of-the-art methods on two synthetic scenes: TOASTER and MIRROR.

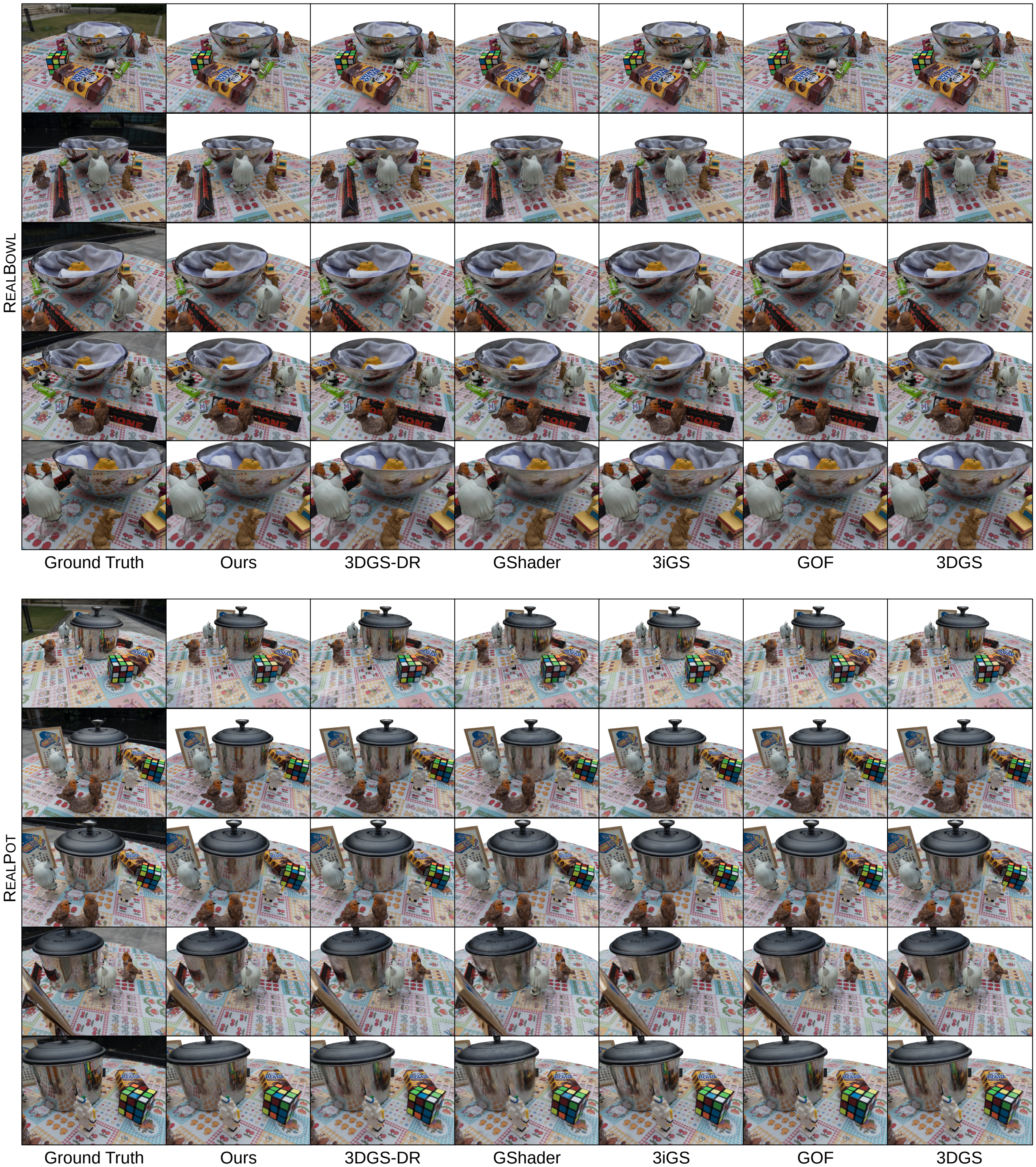


Figure 14. Comparison with state-of-the-art methods on two real-world scenes: REALBOWL and REALPOT.

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 *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 13

[2] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-supervised interest point detection and description. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018. 12

[3] Cycles Developers. Cycles: Open source production rendering, 2024. https://www.cycles-renderer.org/. Accessed: 2024-11-06. 12

[4] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. GaussianShader: 3D Gaussian splatting with shading functions for reflective surfaces. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 14, 15, 16

[5] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42(4):139:1–139:14, 2023. 12, 14, 15, 16

[6] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. LightGlue: Local feature matching at light speed. In *Proc. of IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 12

[7] Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David P. Luebke, David K. McAllister, Morgan McGuire, R. Keith Morley, Austin Robison, and Martin Stich. OptiX: a general purpose ray tracing engine. In *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH)*, 2010. 11

[8] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 12

[9] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *Proc. of European Conference on Computer Vision (ECCV)*, 2016. 12

[10] Zhe Jun Tang and Tat-Jen Cham. 3iGS: Factorised tensorial illumination for 3D Gaussian splatting. In *Proc. of European Conference on Computer Vision (ECCV)*, 2024. 14, 15, 16

[11] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing (TIP)*, 13(4):600–612, 2004. 15, 16

[12] Chongjie Ye, Lingteng Qiu, Xiaodong Gu, Qi Zuo, Yushuang Wu, Zilong Dong, Liefeng Bo, Yuliang Xiu, and Xiaoguang Han. StableNormal: Reducing diffusion variance for stable and sharp normal. In *Proc. of the ACM SIGGRAPH Conference and Exhibition on Computer Graphics and Interactive Techniques in Asia (SIGGRAPH Asia)*, 2024. 12

[13] Keyang Ye, Qiming Hou, and Kun Zhou. 3D Gaussian splatting with deferred reflection. In *Proc. of the ACM SIGGRAPH Conference and Exhibition On Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2024. 14, 15, 16

[14] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 43(6):271:1–271:13, 2024. 11, 12, 14, 15, 16

[15] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 15, 16