Deformable Radial Kernel Splatting

Supplementary Material

Outline

In this supplementary file, we provide additional applications and potential usages of *DRK*, an introduction to the DiverseScenes dataset, implementation details, and further results that could not be included in the main paper due to space constraints. The content is organized as follows:

- Sec. S1: More results of seamless conversion from mesh models to *DRK* representations, bridging millions of 3D assets with high-fidelity reconstructed scenes.
- Sec. S2: Introduction to the DiverseScenes dataset.
- Sec. S3: Additional experimental results on public datasets and an analysis of limitations.
- Sec. S4: Detailed implementation of the *DRK* framework.

S1. Converting Mesh to DRK

We present more examples of converting 3D mesh assets to *DRK* within seconds, without the need for training data preparation or optimization, in Fig. S1. Rendered depth and normal images are also provided. In these examples, the *DRK* is kernel-wise colored and shaded over the base color using the normal and predefined illumination. In the future, by assigning UV attributes to *DRK* and rendering them into images, material properties such as albedo, roughness, and metallicity can be retrieved from the material maps, enabling deferred rendering using the rendered normals. This capability allows *DRK* to handle traditional assets and compose scenes reconstructed from real-world multi-views and man-made artistic 3D assets.

It is worth noting that cache-sorting has minimal impact on reconstruction quality but plays a crucial role in the conversion from mesh to *DRK*. This is because Mesh2*DRK* produces a compact geometric representation, where each *DRK* kernel represents a relatively larger unit of a mesh face compared to those learned from multi-view images, which use smaller units to capture high-frequency appearance details. Cache-sorting ensures that the sorting order is nearly correct, resulting in satisfactory conversion outcomes.

S2. DiverseScenes Dataset

We collected 10 scenes from Sketchfab[1], encompassing a variety of 3D scene types. As illustrated in Fig. S2, the dataset includes scenes with simple geometry and textures (e.g., McCree, House), detailed textures (e.g., Newspaper, Book), fine geometry (e.g., PalmTree, Dress), and large scales (e.g., Minecraft, Street). The training set consists of 200 views, and the test set includes 30 views, sampled from the unit sphere. For Minecraft and Street, there are 230 training views, with the train and test views simulating a walking camera along specified paths.

[1]https://sketchfab.com/

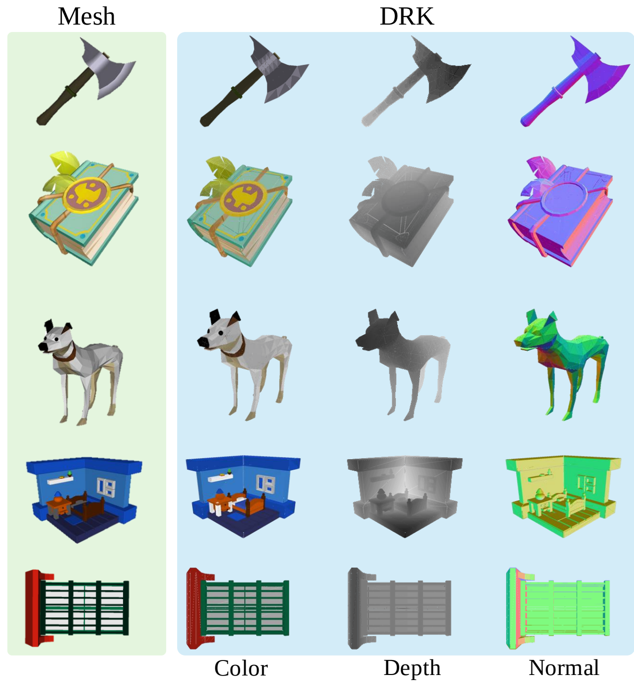

Figure S1. Examples of converting various 3D assets. The normal of *DRK* can be used for shading under illumination. UV texture mapping is applicable for *DRK* in future implementations.

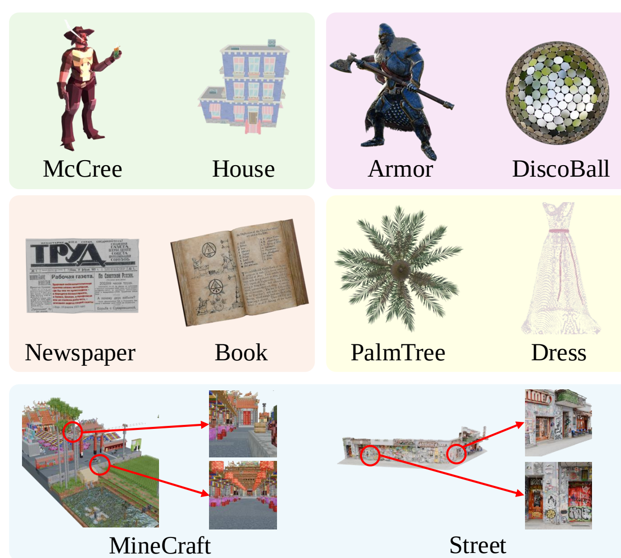

Figure S2. Overview of DiverseScenes. The dataset is composed of five categories: simple, specular, detailed geometry, fine texture, and large scale.

Methods	McCree			House		
	PSNR(↑)	**LPIPS(↓)**	**SSIM(↑)**	**PSNR(↑)**	**LPIPS(↓)**	**SSIM(↑)**
2D-GS	40.95	.0067	.9985	43.25	.0351	.9953
3D-GS	39.15	.0115	.9975	39.87	.0507	.9907
3D-HGS	40.64	.0062	.9965	42.17	.0335	.9897
GES	40.48	.0086	.9982	41.42	.0446	.9927
Ours (S2)	39.76	.0104	.9977	40.92	.0462	.9914
Ours (S1)	40.90	.0067	.9985	43.57	.0269	.9963
Ours	41.70	.0047	.9989	45.22	.0163	.9980

Methods	Book			Newspaper		
	PSNR(↑)	**LPIPS(↓)**	**SSIM(↑)**	**PSNR(↑)**	**LPIPS(↓)**	**SSIM(↑)**
2D-GS	38.20	.1199	.9673	46.97	.0048	.9995
3D-GS	42.25	.1211	.9944	46.89	.0045	.9991
3D-HGS	43.48	.0775	.9842	48.89	.0019	.9981
GES	43.49	.1122	.9960	46.97	.0036	.9993
Ours (S2)	43.28	.1042	.9961	47.75	.0032	.9991
Ours (S1)	45.57	.0904	.9976	49.76	.0023	.9992
Ours	47.28	.0801	.9983	50.88	.0019	.9993

Methods	Dress			PalmTree		
	PSNR(↑)	**LPIPS(↓)**	**SSIM(↑)**	**PSNR(↑)**	**LPIPS(↓)**	**SSIM(↑)**
2D-GS	26.04	.0367	.9807	33.76	.0659	.9593
3D-GS	25.45	.0378	.9782	32.82	.0725	.9533
3D-HGS	27.30	.0217	.9773	33.54	.0595	.9423
GES	27.31	.0254	.9859	33.41	.0682	.9549
Ours (S2)	27.56	.0362	.9770	33.80	.0592	.9648
Ours (S1)	29.19	.0225	.9899	34.14	.0562	.9677
Ours	29.61	.0196	.9917	34.25	.0536	.9688

Methods	Armor			Discoball		
	PSNR(↑)	**LPIPS(↓)**	**SSIM(↑)**	**PSNR(↑)**	**LPIPS(↓)**	**SSIM(↑)**
2D-GS	32.65	.0500	.9742	21.70	.0639	.9357
3D-GS	31.51	.0687	.9684	22.81	.0828	.9361
3D-HGS	32.26	.0518	.9512	22.28	.0505	.9381
GES	31.98	.0603	.9713	22.09	.0661	.9386
Ours (S2)	32.01	.0592	.9736	22.13	.0816	.9404
Ours (S1)	33.27	.0430	.9791	22.26	.0832	.9414
Ours	34.07	.0346	.9814	22.31	.0838	.9415

Methods	Minecraft			Street		
	PSNR(↑)	**LPIPS(↓)**	**SSIM(↑)**	**PSNR(↑)**	**LPIPS(↓)**	**SSIM(↑)**
2D-GS	23.15	.4119	.7262	32.45	.0881	.9514
3D-GS	25.81	.3516	.8116	38.56	.0591	.9917
3D-HGS	26.10	.3070	.7568	40.23	.0274	.9873
GES	25.83	.3554	.8072	37.51	.0594	.9902
Ours (S2)	25.01	.3584	.8019	38.12	.0596	.9950
Ours (S1)	25.98	.3057	.8410	41.58	.0312	.9981
Ours	26.84	.2512	.8755	43.72	.0183	.9989

Methods	Average					
	PSNR(↑)	**LPIPS(↓)**	**SSIM(↑)**	**Num(↓)**	**Size(↓)**	**FPS(↑)**
2D-GS	33.92	.0881	.9514	359K	83.6M	251.3
3D-GS	34.41	.0861	.9621	346K	82.0M	247.1
3D-HGS	35.68	.0637	.9521	373K	89.6M	154.5
GES	35.05	.0804	.9634	330K	78.1M	227.4
Ours (S2)	35.03	.0823	.9637	42K	12.3M	234.9
Ours (S1)	36.62	.0668	.9701	109K	32.1M	119.2
Ours	37.58	.0564	.9752	260K	76.6M	77.5

Table S1. We show the PSNR, LPIPS, and SSIM metrics for novel view synthesis on DiverseScenes.

To offer a more comprehensive overview of the DiverseScenes Dataset, we manually annotated the attributes of each scene, as summarized in Table S2. Note that some attributes overlap; for instance, both Armor and Street include fine textures. To clarify the performance of methods on DiverseScenes, we present the per-scene results in Table S1. These results are also summarized by scene categories in the main paper.

Scene	Geometry		Texture		Scale		Material	
	Coarse	Fine	Coarse	Fine	Small	Large	Diffuse	Specular
McCree	✓	×	✓	×	✓	×	✓	×
House	✓	×	✓	×	✓	×	✓	×
Book	✓	×	×	✓	✓	×	✓	×
Newspaper	✓	×	×	✓	✓	×	✓	×
Dress	×	✓	✓	×	✓	×	✓	×
PalmTree	×	✓	✓	×	✓	×	✓	×
Armor	✓	×	×	✓	✓	×	×	✓
DiscoBall	✓	×	✓	×	✓	×	×	✓
Street	✓	×	×	✓	×	✓	✓	×
MineCraft	✓	×	✓	×	×	✓	✓	✓

Table S2. Summary of scene attribute annotations.

Scene	2D-GS	3D-GS	3D-HGS	GES	*DRK* (S2)	*DRK* (S1)	*DRK*
Chair	34.88	35.83	34.29	34.05	34.38	35.28	35.61
Drums	25.67	26.15	26.29	26.05	25.90	26.12	26.13
Ficus	35.80	34.87	35.45	35.27	35.56	36.27	36.50
Hotdog	36.89	37.72	37.54	37.13	37.19	37.84	38.17
Lego	34.82	35.78	33.92	33.73	33.90	35.38	36.25
Materials	30.14	30.00	29.88	29.74	29.38	30.14	30.48
Mic	34.38	35.36	36.58	35.73	35.17	35.70	36.00
Ship	31.09	30.80	31.10	30.94	30.84	31.28	31.42
Avg PSNR	32.96	33.32	33.13	32.83	32.79	33.50	33.82
Num	107K	131K	83K	73K	32K	75K	158K
Size	25.0M	31.1M	20.0M	17.4M	9.6M	22.0M	46.6M

Table S3. PSNR scores, primitive numbers, and model sizes on NeRF-Synthetic [?] scenes (transposed).

S3. More Experiments & Limitation Analysis

Evaluation on NeRF-Synthetic Scenes. We conducted a quantitative evaluation on NeRF-Synthetic [?] scenes, reporting PSNR scores, primitive numbers, and model sizes in Table S3. Unlike DiverseScenes, the training cameras for NeRF-Synthetic scenes (except Ficus) are sampled from the upper hemisphere, and some test views fall outside the range covered by the training views. This setup partially assesses performance on view extrapolation. Our *DRK* method demonstrates superior performance. While the kernel number for *DRK* is slightly larger than for 3D-GS [?], *DRK* (S1) maintains a compact number and outperforms other methods in PSNR scores. *DRK* (S3) has a very small kernel number and model size, with an average PSNR still comparable to others.

Ablation Study To gain a comprehensive understanding of DRK, we perform ablation studies on its attributes to assess their impact. We remove the effects of sharpening and curvature learning by setting $\tau = 0$ and $\eta = 0$, respectively. We also examine the impact of the hyper-parameter K. Ablation results on DiverseScenes are shown in Tab. S4. We found that K significantly influences performance, with $K = 3$ causing the greatest drop. Additionally, η is more vital than τ, though both enhance representation capability.

Evaluation on Tank&Temple Scenes. To assess our method's performance on more challenging scenes with imperfect camera conditions due to dynamic objects and changing exposures, we evaluated the Tank&Temple [?]

Methods	$\tau = 0$	$\eta = 0$	$K = 3$	$K = 5$	$K = 8$
DRK (S2)	34.97	34.82	33.93	34.47	35.03
DRK (S1)	36.22	36.04	35.50	35.88	36.62
DRK	37.46	37.27	36.60	36.85	37.58

Table S4. Ablation study on the impact of τ, η, and K.

datasets. We used 9 scenes in total, including 8 intermediate scenes and the Truck scene. We report the PSNR scores, foreground-only PSNR scores (M-PSNR), primitive numbers, and model sizes in Table S5. The results indicate that *DRK* faces significant challenges with this dataset, likely due to higher camera error estimated by COLMAP [**?**]. The Tank&Temple datasets are captured in dynamic environments with moving pedestrians and changing exposure, making camera estimation more difficult than in MipNeRF-360 [**?**], where objects are primarily diffuse and free from view-dependent effects, transients, or significant sunlight exposure changes. To further investigate the robustness of *DRK*, we conducted evaluations with noisy camera data.

Methods	2D-GS	3D-GS	3D-HGS	GES	Ours (S2)	Ours (S1)	Ours
PSNR	20.65	21.09	21.59	20.58	20.20	20.31	20.41
M-PSNR	26.50	26.92	27.58	26.56	26.36	26.41	26.37
Num	1168K	275K	267K	259K	173K	212K	383K
Size	271.9M	65.4M	64.2M	61.7M	50.9M	62.4M	112.6M

Table S5. Quantitative evaluation on Tank&Temple scenes.

Robustness against Camera Noise To evaluate the performance of *DRK* under varying levels of camera noise, we simulated camera noise with different standard deviations (Std). The PSNR scores on DiverseScenes with noisy cameras are reported in Table S6. We observed that the PSNR scores of *DRK* drop significantly as the camera noise increases, whereas the performance of 3D-GS degrades more smoothly and slightly.

Noise *Std*	3D-GS	*DRK*	*DRK* (S1)	*DRK* (S2)
0	34.41	37.58	36.62	35.03
$1e-3$	33.44	31.59	31.19	30.88
$2.5e-3$	31.32	29.27	28.89	28.60
$5e-3$	29.37	27.85	27.44	27.14

Table S6. Average PSNR scores on DiverseScenes of 3D-GS and *DRK* under different levels of camera noise.

Fig. S3 shows the rendering results of 3D-GS and *DRK* trained with both accurate and noisy cameras. When trained with accurate cameras, *DRK* achieves higher-quality rendering with sharper and clearer appearances. However, even with very small camera noise, the performance of *DRK* deteriorates significantly, producing blurrier and more chaotic results compared to 3D-GS. In contrast, 3D-GS maintains the ability to model the coarse appearance of the scene under noisy conditions. These results demonstrate that *DRK* is less robust to camera noise, which may explain its performance drop on the Tank&Temple dataset.

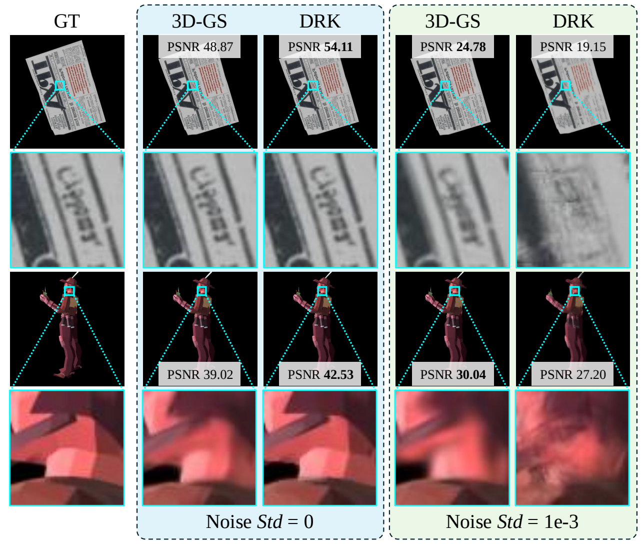

Figure S3. Rendering results of 3D-GS and *DRK* trained on cameras with and without noise.

S4. Method Details

Parametrization. *DRK* attributes are modeled using unconstrained learnable parameters in $(-\infty, \infty)$, with appropriate activation functions to ensure valid ranges. We apply a sigmoid function to constrain opacity o to $(0, 1)$, an exponential function for scale activation s_k, and a normalization function for rotation quaternion q. The sharpness τ is bounded within $(-0.1, 0.99)$ through a composite function combining sigmoid and linear remapping, while the $L1\&L2$ blending weight η is activated using a sigmoid function. For the basis angle θ_k, we employ a three-step activation process: first applying a sigmoid function, then adding a residual term $\frac{1}{K-2}$ to maintain minimum angular separation, applying cumulative summation to enforce monotonic increase, and finally normalizing to make $s_K = 2\pi$. The residual term prevents basis polar angles from exceeding π, thereby avoiding degradation in the representation. We set $K = 8$ to balance flexibility and memory efficiency.

Model Training. Following 3D-GS, we optimize model parameters and dynamically adjust kernel density through an adaptive training process. For *DRK-specific* parameters - sharpness τ, blending weight η, basis angles θ_k, and scales s_k - we set a uniform learning rate of $5e^{-3}$ and decay them gradually to the rate $1e^{-2}\times$ at the end of training (35K steps). We implement three density control configurations through the 2D screen gradient densification threshold and opacity pruning threshold pairs: ($5e^{-4}$, $5e^{-2}$) for density comparable to 3D-GS, ($1e^{-3}$, $5e^{-2}$) for Sparse Level 1 (**S1**), and ($2e^{-3}$, $1e^{-1}$) for Sparse Level 2 (**S2**).

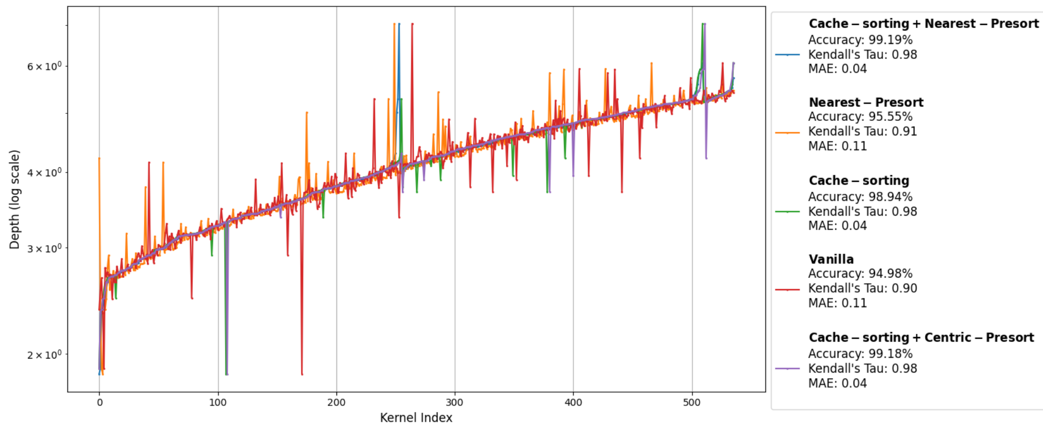

Figure S4. Sorting accuracy comparisons: We found that pre-sorting tile-kernel pairs based on the nearest distance, combined with cache-sorting, achieves the highest accuracy. Cache-sorting is sufficiently effective in correcting most sorting disorders.

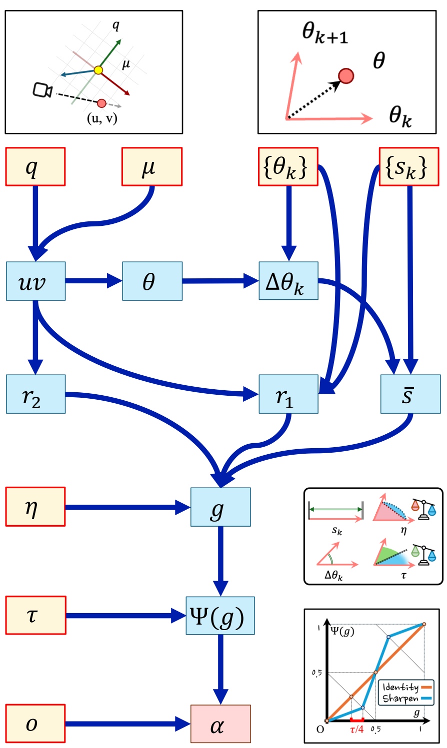

Figure S5. The "Tensor Graph" of *DRK*, showing the dependence between parameters to optimize, intermediate variables, and the final output (α).

Tensor Graph of *DRK* To provide a detailed overview of *DRK*, we present its "Tensor Graph," which illustrates the flow from the learnable leaf parameters through the intermediate variables, ultimately leading to the outputs. The graph is shown in Fig. S5. Blue arrows represent data dependencies, along which gradients are back-propagated in reverse during optimization.

Cache-Sorting To clarify the cache-sorting algorithm, we briefly summarize the process in Algorithm 1. As discussed in StopThePop [?], the backward processing must also be adjusted to proceed from front to back to maintain consistency with forward rendering.

Algorithm 1: Cache-Sorting

Input: A cache chain with limited size, a new *DRK* (with an index and depth)
Output: The index of textitDRK or a status code

```
if the cache is empty then
    if the new DRK is invalid then
        return a finish code;
    Initialize the cache with the new DRK;
    return success;
if the new DRK is invalid then
    Pop the DRK from the head;
    return the index of the popped DRK;
if the cache is full then
    Mark the head DRK to be popped;
Determine where to insert the new DRK by
  scanning the cache, guided by the DRK's depth;
Adjust the pointers in the cache to insert the new
  DRK at the correct position;
if the cache was full then
    Pop the oldest DRK;
return the index of the popped DRK (or success if
  none was popped);
```

We evaluate the effectiveness of cache-sorting using a cache length of 8. *DRK* kernels are randomly sampled from the space, and the depths of *DRK* intersections processed in a front-to-back order are visualized in Fig. S4. Additionally, we assess the performance using metrics such as accuracy, Kendall's Tau, and MAE. Our results show that in the pre-sorting stage (kernel-tile sorting), sorting based on the nearest distance between the *DRK* and the tile achieves the highest sorting accuracy. Sorting based on the most centric approach closely follows in performance. Both methods provide notable improvements compared to cache-sorting alone. Presorting with the nearest distance is also better than the vanilla presorting strategy on *DRK*. For further details and a more in-depth discussion, we refer readers to the StopThePop [**?**] paper, a pioneering work in this field.