GOR-IS: 3D Gaussian Object Removal in the Intrinsic Space

Supplementary Material

In this supplementary material, we provide additional implementation details (Sec. S1), describe the dataset construction and post-processing procedures (Sec. S2), discuss the limitations of our framework (Sec. S3), discuss on non-Lambertian scene modeling (Sec. S4), and present additional ablation studies (Sec. S5). Finally, we present additional visual results (Sec. S6), including evaluations on the extra real-world datasets [1], visualizations of material decomposition, and comparisons on the SPIn-NeRF dataset [7].

S1. More implementation details

This section provides additional implementation details of our framework, including region division in material and lighting decoupling (Sec. S1.1), ray tracing in 3DGS (Sec. S1.2), the screen-space filter (Sec. S1.3), the intrinsic-space inpainting module (Sec. S1.4), the overall training strategy (Sec. S1.5), efficiency analysis (Sec. S1.6), and the implementation of baseline methods (Sec. S1.7).

S1.1. Region division in material and lighting decoupling

In implementation, we divide the scene into two regions based on surface characteristics:

1. **Glossy regions**, corresponding to the non-Lambertian surfaces discussed in this paper, whose appearance varies sharply with viewing direction.
2. **Rough regions**, whose appearance changes smoothly with viewing direction. We approximate these regions as Lambertian surfaces for modeling simplicity.

For glossy regions, we compute the outgoing color using Eq. 1 of the main text to maintain consistent global lighting effects. For rough regions, we approximate the surface as Lambertian and omit glossy reflection modeling, retaining only the diffuse reflection term. This simplification is motivated by the following considerations:

1. Rough regions exhibit negligible glossy reflections and weak global lighting effects, making explicit glossy reflection modeling unnecessary.
2. The BRDF lobes in rough regions are broad, making the glossy reflection difficult to approximate using a single traced ray (even with our screen-space filtering strategy). Dense ray sampling would be required, leading to substantial computational cost.
3. As noted in prior work [14], skipping glossy reflection modeling in rough regions reduces computation. Ray tracing is performed only for glossy regions, while rough regions incur no tracing cost, reducing the total number of rays.

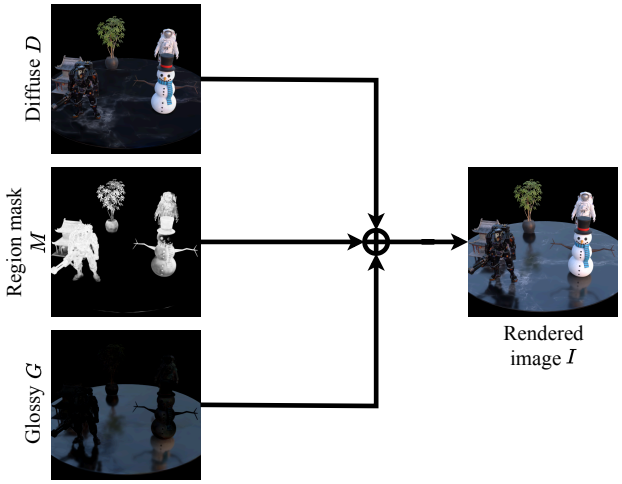

Figure 1. Illustration of region division and blending.

Considering the above factors, explicitly modeling glossy reflection in rough regions incurs substantial computational overhead and provides limited benefit for maintaining global lighting effects consistency. Therefore, we omit explicit glossy reflection modeling for these regions. Finally, although rough surfaces primarily exhibit diffuse behavior, their appearance still shows mild view dependence. To better capture this effect, we model the diffuse reflection term using spherical harmonics (SH), which provide stronger expressive capacity.

For region division, we follow prior work [14] by assigning each Gaussian primitive an indicator property $\boldsymbol{m}$. This property is rasterized onto the screen space to obtain a region mask M that distinguishes the two region types. Then, based on the region mask M, the outgoing color C is redefined as:

$$C = D + (1 - M)G. \tag{1}$$

Here, D denotes the diffuse reflection term and G denotes the glossy reflection term. The region mask M suppresses the glossy component in rough regions while preserving it in glossy regions. Pixels with $M \approx 1$ rely primarily on diffuse reflection, whereas pixels with $M \approx 0$ retain full glossy effects. An intuitive example is shown in Fig. 1, where each pixel in the final rendered image I is obtained by blending diffuse and glossy components guided by the mask M. As in SpecTRe-GS [14], the region mask is implemented as a soft, differentiable mask that allows gradients to flow during training, enabling the model to learn the region division. Besides, We also supervise the region mask M using a precomputed segmentation map M_{gt}

(obtained using SAM2 [9] via click selection, where glossy regions are labeled as 1). The corresponding loss function is defined as:

$$\mathcal{L} = \|M \odot M_{\text{gt}}\|_1, \tag{2}$$

which suppresses M values in glossy regions to 0, thereby promoting effective region division.

S1.2. Ray tracing in 3DGS

We employ gtracer [2] for 3DGS ray tracing. Specifically, each Gaussian primitive is treated as an ellipsoidal volume, and a bounding volume hierarchy (BVH) is constructed over these ellipsoidal primitives to enable efficient ray tracing. Given a spatial point $\boldsymbol{x}$ and a tracing direction $\boldsymbol{\omega}_t$, we cast a ray from $\boldsymbol{x}$ along $\boldsymbol{\omega}_t$, identify all Gaussian primitives intersected by the ray, sort them in order of intersection depth, and accumulate their opacity and radiance via alpha blending to obtain the visibility $V(\boldsymbol{x}, \boldsymbol{\omega}_t)$ and the incident radiance $L_i(\boldsymbol{x}, \boldsymbol{\omega}_t)$. In our implementation, we further extend gtracer to compute the gradients of visibility and incident radiance with respect to both the spatial position $\boldsymbol{x}$ and the tracing direction $\boldsymbol{\omega}_t$, which facilitates the optimization of non-Lambertian (glossy) surface geometry. It should be noted that we only trace incident radiance from Lambertian (rough) regions to avoid multiple non-Lambertian surfaces reflecting each other.

S1.3. Screen-space filter

The screen-space filter achieves realistic glossy effects by constructing a mipmap pyramid to filter the ideal specular reflection S. In practice, we do not directly apply filtering to S. Specifically, we decompose the ideal specular reflection as:

$$\begin{aligned} &S(\boldsymbol{x}, \boldsymbol{\omega_o}) = F(\boldsymbol{x}, \boldsymbol{\omega_r}) L_i(\boldsymbol{x}, \boldsymbol{\omega_r}) \\ &= F(\boldsymbol{x}, \boldsymbol{\omega_r}) \big[L_{\text{ind}}(\boldsymbol{x}, \boldsymbol{\omega_r}) + L_{\text{dir}}(\boldsymbol{x}, \boldsymbol{\omega_r}) V(\boldsymbol{x}, \boldsymbol{\omega_r}) \big], \end{aligned} \tag{3}$$

where L_{ind} and L_{dir} denote the indirect and direct radiance, respectively, and V represents visibility.

In our implementation, we observe that the Fresnel reflectance F varies relatively slowly with the reflection direction and therefore does not require filtering. Consequently, our filtering primarily targets the incident radiance. For the direct radiance L_{dir}, which is modeled by the environment map defined in the spherical domain, we follow the conventional approach from the split-sum approximation and perform filtering in the spherical domain according to the surface roughness R.

For the indirect radiance L_{ind} and the visibility V, we adopt our proposed screen-space mipmap filtering strategy. However, since spherical-space and screen-space filtering are defined in different domains, the same surface roughness cannot be used for both. To address this, we introduce a lightweight neural network that translates the original surface roughness R into the corresponding screen-space roughness R_s.

Furthermore, the screen-space filtering kernels should depend on the distance between the shading point $\boldsymbol{x}$ and the camera. Intuitively, regions farther from the camera occupy fewer pixels and thus require smaller filtering kernels. Therefore, we incorporate depth D as an additional input to the translation network to adaptively adjust the filtering kernel according to the camera–point distance.

In summary, the screen-space filtering procedure is as follows: Given a shading point $\boldsymbol{x}$ with its corresponding indirect radiance L_{ind}, direct radiance L_{dir}, visibility V, Fresnel term F, roughness R, and depth D, we first compute the screen-space roughness R_s using the translation network: $R_s = \text{Net}(R, D)$. We then use mipmap to filter L_{ind} and V according to R_s to obtain the filtered results L'_{ind} and V', respectively. Meanwhile, L_{dir} is filtered in spherical space based on the original roughness R, yielding L'_{dir}. Finally, the final glossy reflection term G is defined as:

$$\begin{aligned} G = L[S, R] &= L[F(\boldsymbol{x}, \boldsymbol{\omega_r}) L_i(\boldsymbol{x}, \boldsymbol{\omega_r}), R] \\ &= F(\boldsymbol{x}, \boldsymbol{\omega_r})(L'_{\text{ind}} + L'_{\text{dir}} V'). \end{aligned} \tag{4}$$

For the mipmap implementation, we construct a 5-level mipmap pyramid. The first level corresponds to the original input, and each subsequent level is generated by applying Gaussian filtering followed by 2× downsampling to the previous level. During filtering, we follow the standard practice of using the screen-space roughness value (ranging from 0 to 1) to perform linear interpolation sampling between mipmap levels, thereby achieving roughness-guided filtering.

For the translation network, we adopt a convolutional neural network architecture. The network takes a 2-channel input (roughness R and depth D) and outputs a 1-channel screen-space roughness map R_s. It consists of 8 convolution layers in total: 2 layers for input and output, each with a kernel size of 1, and 6 latent layers, each with a feature dimension of 8 and a kernel size of 3. ReLU [8] is used as the activation function for all latent layers.

For direct radiance modeling, we employ a simple differentiable environment map. The direct radiance is modeled using a cube-map format with a resolution of ($6 \times 256 \times 256$).

Finally, the computational overhead introduced by the screen-space filter is negligible. The module requires no extra ray-tracing operations— all filtering is performed entirely in screen space and relies only on a lightweight convolutional neural network, resulting in minimal additional cost.

S1.4. Intrinsic-space inpainting

Inpainting mask. We adopt the inpainting mask generation method proposed in 3DGIC [3]. Specifically, we combine multi-view depth maps and object masks to identify completely occluded regions (i.e., regions occluded by the target object) that are never visible from any angle. The detailed algorithm is described in the 3DGIC paper.

Reference view selection. We select reference views for 2D inpainting and lift the inpainted results to 3D to complete 3D object removal. Our reference view selection follows 3DGIC [3]: we choose three views with the largest 2D inpainting mask areas to maximize 3D coverage. In practice, using more reference views (e.g., four or five) does not provide additional benefits. This strategy has proven robust across most of our cases.

To mitigate occlusion interference in the 3D inpainting, we select reference views for 2D inpainting that cover the largest 3D spatial extent, thereby minimizing occluded references. However, in extreme cases where other objects occlude the inpainting region in all reference views, the method cannot reliably inpaint the missing content. A possible extension is to incorporate virtual camera views, as explored in Inpaint360GS [16], which is orthogonal to our framework. We leave it for future work.

Scene inpainting initialization. Scene inpainting aims to achieve geometrically complete and visually seamless restoration of occluded regions. Since these areas are entirely invisible before object removal, we initialize new Gaussian primitives to cover them as completely as possible, facilitating subsequent optimization-based inpainting. We follow the initialization strategy provided in 3DGIC [3]. Specifically, we perform depth inpainting on the rendered depth maps of the selected reference views to obtain the corresponding reference depths. Using the corresponding camera parameters and depths, we then back-project the pixels within the inpainting regions into 3D space to generate the initial Gaussian primitive positions. The remaining geometric properties of each Gaussian are initialized via nearest-neighbor interpolation, while the material and color properties are uniformly set to 0.5.

Scene inpainting. During the scene inpainting stage, we optimize the Gaussian primitives under supervision from 2D inpainting results to repair the occluded regions. Our supervision strategy follows 3DGIC [3]. Specifically, we back-project the reference inpainted images into 3D space to construct a reference point cloud that contains both appearance colors and material properties. For each training iteration with a training camera cam_i, we proceed as follows:

1. We first compute the loss defined in Eq. 5 of the main text on the non-inpainting regions of the current training view cam_i to ensure that these regions remain unchanged.
2. We then randomly select one of the reference views, denoted as cam_r, and use it to compute the inpainting loss $\mathcal{L}_{\text{inpaint}}$ defined in Eq. 6 of the main text.
3. If the current training view cam_i is not one of the reference views, we project the reference point cloud into this view and compute $\mathcal{L}_{\text{inpaint}}$ accordingly.

As described in the main text, the appearance loss $\mathcal{L}_{\text{A}}$ is applied only to non-Lambertian (glossy) regions, whereas the material loss $\mathcal{L}_{\text{M}}$ is applied only to Lambertian (rough) regions. To correctly distinguish these regions during inpainting, we also inpaint the region mask M in the 2D inpainting stage.

Specifically, the region mask M serves as an indicator for distinguishing non-Lambertian (glossy) regions from Lambertian (rough) regions. However, when the target object occludes part of the surface, the region-mask values in those occluded areas become unknown. Without inpainting M in these regions, we would be unable to determine which loss to apply (appearance vs. material) during optimization. To resolve this, we jointly inpaint M together with other properties (color images and material maps) during the 2D inpainting stage. The resulting inpainted region mask $\hat{M}$ is then used to differentiate regions when computing the corresponding losses.

Lighting-aware masking mechanism. In our implementation of the lighting-aware masking mechanism, we set the threshold τ to 0.1 to detect pronounced reflections cast by the target object onto surrounding surfaces.

2D inpainting model. The 2D inpainting model is not the primary focus of our framework; therefore, we adopt the widely used LaMa [13] as the backbone inpainting network. To further assess the robustness of our method to different 2D inpainting backbones, we replace LaMa with SD-1.5-Inpainting [10] and evaluate the model on the GOR-IS-Synthetic dataset. As shown in Table 1, replacing LaMa with SD-1.5-Inpainting yields comparable performance, demonstrating the stability of our framework across different 2D inpainting models.

Table 1. Ablation study of 2D inpainting backbone. The best results are highlighted in red

Inpainting backbones	PNSR	SSIM	LPIPS	M-LPIPS	FID	M-FID
LaMa	31.91	0.947	0.039	0.060	23.4	65.0
SD-1.5-Inpainting	32.00	0.947	0.039	0.066	24.1	68.5

S1.5. Overall training strategy

Gaussian densification and pruning strategy. We follow the Gaussian densification and pruning strategy proposed in RaDe-GS [20]. In the first training stage, densification and pruning begin at step 500 and stop at step 12K. In the second stage, the process starts at step 500 and stops at step 2K. For both stages, the interval between consecutive densification and pruning operations is 500 steps, and the opacity of each Gaussian is reset every 3K steps.

Training strategy for the first stage. In the first stage, we follow the initialization procedure in RaDe-GS [20] to train the radiance field for 4K steps, which provides an initial reconstruction of the scene. After 4K steps, we introduce explicit light transport modeling to optimize the scene intrinsic decomposition.

For the color loss $\mathcal{L}_{\text{c}}$, we follow 3DGS [4] and combine the L1 and SSIM losses between rendered and ground-truth images. The depth distortion loss and depth normal consistency loss follow RaDe-GS [20], with a slight modification: the depth distortion loss is enabled after 3K iterations, and the depth normal consistency loss is enabled after 7K iterations. The depth distortion loss is computed in normalized device coordinate (NDC) space to avoid scale inconsistencies and does not require ground-truth supervision.

The normal loss $\mathcal{L}_{\text{n}}$ is defined as:

$$\mathcal{L}_{\text{n}} = \|M_{\text{gt}} \odot (N - N_{\text{gt}})\|_1, \tag{5}$$

where N denotes the rendered normal, N_{gt} is the reference normal predicted by DiffusionRenderer [5], and M_{gt} is the precomputed region mask introduced in Sec. S1.1. This mask restricts the loss to non-Lambertian (glossy) regions, providing a strong prior for reconstructing non-Lambertian surfaces and preventing geometric artifacts.

The bilateral smoothing loss $\mathcal{L}_{\text{s}}$ is defined as:

$$\mathcal{L}_{\text{s}} = (M_{\text{gt}} \odot \|\nabla X\| \exp^{-\|\nabla I_{\text{gt}}\|}), (X \in \{F, R, N, N_d\}), \tag{6}$$

where ∇ denotes the gradient operator and I_{gt} is the ground-truth image. This loss is applied to the Fresnel F, roughness R, rendered normal N, and depth-normal N_d maps, encouraging material and geometric smoothness while suppressing unwanted artifacts. This loss is further masked by M_{gt} to concentrate the regularization on non-Lambertian regions.

The binary cross-entropy loss $\mathcal{L}_{\Omega}$ is defined as:

$$\mathcal{L}_{\Omega} = -\Big(\Omega_{\text{gt}} \log(\Omega) + (1 - \Omega_{\text{gt}}) \log(1 - \Omega)\Big), \tag{7}$$

where Ω denotes the rendered object mask (derived from Gaussian label properties), and Ω_{gt} is the ground-truth object mask. This loss supervises the label properties of Gaussian primitives using predefined object masks, enabling accurate identification of Gaussian primitives associated with the target object.

The loss weights [λ_{dn}, λ_{n}, λ_{s}, λ_{Ω}] are set to [0.05, 0.5, 0.05, 1.0]. The depth distortion loss weight λ_{d} is set to 1000 for small, bounded object-level scenes, and to 10 for large-scale, unbounded indoor or outdoor scenes. Following SpecTRe-GS [14], the normal loss weight decays exponentially from 4K to 10K iterations, reaching a minimum value of 0.001. Finally, we include the region mask loss introduced in Sec. S1.1 to supervise region division, with its weight set to 1. All loss weights are validated across a wide range of settings to ensure robust generalization.

We further evaluate the stability of the non-Lambertian reconstruction losses $\mathcal{L}_{\text{n}}$ and $\mathcal{L}_{\text{s}}$. Our results show that the method remains stable when loss weights are scaled within the range [$\times 0.5$, $\times 5$]. At low weight scales ($\leq \times 0.2$), insufficient non-Lambertian supervision leads to unstable artifacts. Conversely, excessively large weights over-constrain the geometry, resulting in over-smoothed textures.

Training strategy for the second stage. The detailed training procedure for the second stage is provided in Sec. S1.4. Here, we describe the loss functions used during this stage. The appearance loss $\mathcal{L}_{\text{A}}$ is defined using LPIPS to encourage perceptual realism and mitigate the blurring effects caused by inconsistent multi-view supervision:

$$\mathcal{L}_{\text{A}} = \text{LPIPS}(\hat{I} \odot \hat{M}, I \odot \hat{M}), \tag{8}$$

where I is the rendered RGB image, $\hat{I}$ is the inpainted RGB image, and $\hat{M}$ denotes the inpainted region mask, restricting $\mathcal{L}_{\text{A}}$ to the Lambertian (rough) area. The weight of the appearance loss is set to $\lambda_{\text{A}} = 0.2$.

The material loss $\mathcal{L}_{\text{M}}$ adopts an L1 formulation with a weight of $\lambda_{\text{M}} = 1$:

$$\begin{aligned}\mathcal{L}_{\text{M}} =& \|(\hat{D} - D) \odot (1 - \hat{M})\|_1 + \|(\hat{F} - F) \odot (1 - \hat{M})\|_1 \\ &+ \|(\hat{R} - R) \odot (1 - \hat{M})\|_1 + \|(\hat{N} - N) \odot (1 - \hat{M})\|_1,\end{aligned} \tag{9}$$

where D, F, R, and N denote the rendered diffuse, Fresnel, roughness, and normal maps, and $\hat{D}$, $\hat{F}$, $\hat{R}$, and $\hat{N}$ are their inpainted predictions. Additionally, we apply the inpainted region mask $\hat{M}$ during loss computation to restrict $\mathcal{L}_{\text{M}}$ to non-Lambertian (glossy) regions, enabling more faithful inpainting of non-Lambertian surfaces. The inpainting loss $\mathcal{L}_{\text{inpaint}}$ is applied only to pixels inside the inpainting mask, ensuring that only the occluded regions are modified.

In the second stage, regions outside the inpainting area must remain unchanged. Therefore, we continue to apply the loss terms used in the first stage (excluding the smoothing loss and the binary cross-entropy loss) as supervision for these regions. During loss computation, we further leverage the object masks and lighting-aware masks to exclude (i)

pixels occupied by the target object (object masks) and (ii) pixels influenced by reflections cast by the target object (lighting-aware masks). This prevents these regions from contaminating the supervision.

S1.6. Framework efficiency

The computational bottleneck of our framework primarily lies in the 3DGS ray tracing for indirect radiance estimation, whose complexity scales with both the number of Gaussians and the rendering resolution. We analyze time overhead using a scene from the GOR-IS-Synthetic dataset (scene with target object: snowman). On a single RTX 3090 GPU, with a resolution of 512×512 and approximately 60K Gaussians, two-stage training takes about 1.5 hours, and inference rendering runs at around 15 FPS. We also provide a variant (Ours-distill) that distills ray tracing into the SH representation to accelerate inference, offering a trade-off between quality and efficiency. Table 2 compares training and inference times with baselines on the same scene. The full framework (Ours-full) achieves the highest PSNR but is slower in training and inference. The distilled variant shows a slight drop in PSNR yet maintains SOTA performance while significantly improving inference speed.

For the distillation process, we first apply the full GOR-IS framework to remove the target object, obtaining the resulting scene G. We then initialize a new Gaussian scene G' as the distillation target. At each training iteration, we randomly sample viewpoints and render the scene G to generate a distillation image I_d. This image serves as the ground truth to supervise the training of G'. The distillation training settings follow those of the original RaDe-GS [20], but only the RGB image loss is retained, while geometry-related losses are removed. Notably, distillation is applied only to the trained model to enable fast inference, while the full GOR-IS framework remains indispensable.

S1.7. Implementation of baseline methods

We conduct experiments using the official open-source implementations of all baseline methods. Below, we describe the reference-view setups used in our experiments.

Reference-view setups on the GOR-IS-Synthetic and GOR-IS-Real datasets. Our method and 3DGIC [3] require multiple reference views; for each scene, we select three reference views for training. Infusion [6], AuraFusion360 [18], and GScream [17] rely on a single reference view, for which we choose the highest-quality view among the 3 selected views. GS Grouping [19] and SPIn-NeRF [7] do not depend on reference views and are trained directly to obtain the final results.

Reference-view setups on the SPIn-NeRF dataset. For the SPIn-NeRF dataset [7], all reference-based methods (ours, 3DGIC [3], Infusion [6], AuraFusion360 [18], and GScream [17]) use the reference view provided by GScream [17]. Non-reference-based methods (GS Grouping [19] and SPIn-NeRF [7]) are trained directly to obtain the final results.

S2. Dataset construction and post-processing

In this section, we describe the construction of our proposed dataset and the necessary post-processing procedures.

For synthetic data, we follow the general design principles outlined in previous work [14]. Each scene contains a prominent non-Lambertian surface, such as a polished metal or a smooth marble tabletop, with surface roughness values ranging from 0.01 to 0.25. Around this surface, several objects are placed to generate noticeable global lighting effects. To avoid complex multi-bounce reflections, each scene includes only one non-Lambertian surface. The scenes are rendered in Blender at 800×800 resolution with a black background. Camera parameters are obtained using COLMAP [11, 12]. During training and evaluation, all images are resized to 512×512 resolution.

For real-world data, we capture indoor scenes using a digital camera mounted on a stabilizer to reduce operational errors and mitigate environmental disturbances. The scene setup follows the same principle as in the synthetic data: each scene contains one non-Lambertian surface surrounded by several objects. During data acquisition, we first capture approximately 200 images of the full scene to serve as training views. We then remove a designated object from the scene and capture an additional 100 images, which are used as test views. To ensure consistent illumination, all captures are completed within one hour. Images are recorded in RAW format and processed using standard image-editing software to obtain clean, well-exposed results. For all training images, we employ SAM2 [9] to generate object masks via click selection. SAM2 is a promptable segmentation model that predicts precise segmentation masks for arbitrary objects in both images and videos, supporting interactive segmentation guided by simple prompts such as point clicks or bounding boxes. This capability allows us to obtain object masks with simple user intervention, enabling our framework to be easily extended to unannotated scenes. For the test views, we leverage the model's novel-view synthesis capability to render images containing the target object at the test viewpoints, and subsequently apply SAM2 to these rendered images to obtain the object masks. The captured images have a resolution of 3000×2000, and their camera parameters are estimated using COLMAP [11, 12]. For both training and evaluation, we downsample all images to a resolution of 750×500.

Table 2. Comparison of training and inference time with baseline methods. The best/second-best results are marked as red/gold.

	Ours-distill	Ours-full	3DGIC	AuraFusion360	InFusion	GS-Grouping	GScream	SPIn-NeRF
Training (hour)	2.0	1.5	2.2	1.5	0.2	1.3	0.8	2.0
Inference (FPS)	301	15	164	240	211	100	106	25
PSNR	30.48	31.91	27.30	27.96	26.34	29.64	29.92	24.68

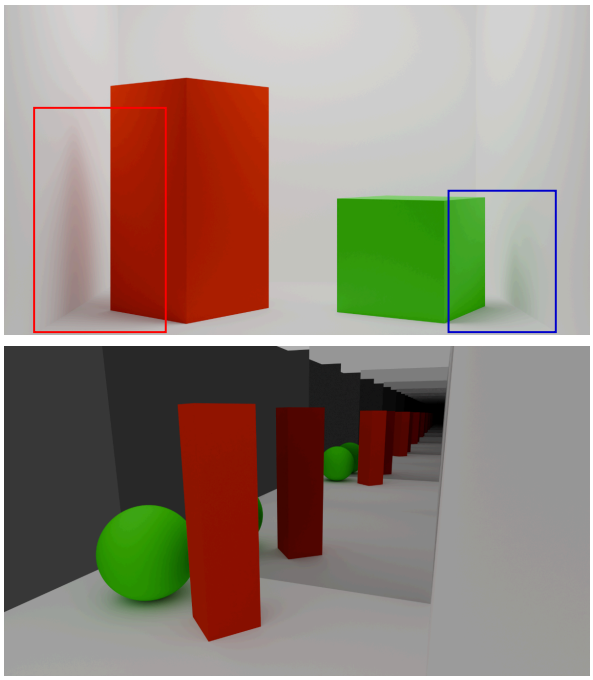

Figure 2. Some scenes that showcase the limitations of our method.

S3. More discussions on limitations

We further provide a more intuitive illustration of the limitations. Fig. 2(a) shows a representative scene composed entirely of Lambertian surfaces. The scene depicts a room with two cuboids illuminated by a top light. As highlighted by the red and blue boxes, radiance emitted from the red and green cuboids is reflected by nearby surfaces. Moreover, the cuboids occlude the light source, casting noticeable shadows on adjacent regions. Our framework struggles to model these diffuse-related global lighting effects, which require more advanced light transport modeling and more robust intrinsic scene decomposition—directions we leave for future work.

Fig. 2(b) further illustrates an extreme case where two mirrors are placed facing each other, causing rays to undergo multiple inter-reflections between the mirror surfaces. Since our method considers only single-bounce rays, it struggles to accurately model multi-bounce light transport. This limitation could be mitigated by incorporating multi-bounce path tracing; however, this would significantly increase computational costs and complicate scene optimization.

Finally, ensuring multi-view consistency in 2D inpainting remains a key challenge in object removal. Inconsistent 2D results may introduce texture inconsistencies across reference views, potentially leading to blur in the inpainted 3D scenes. As our framework relies on the 2D inpainting model LaMa [13], the cross-view inconsistency still persists, particularly in challenging cases. We plan to further investigate improvements in this direction in future work.

S4. More discussions on non-Lambertian scene modeling

Recent works [14, 21] have employed 3DGS to model non-Lambertian (glossy) scenes, with a strong emphasis on reproducing global lighting effects (such as inter-reflections) via intrinsic decomposition and ray tracing, thereby achieving highly realistic novel-view synthesis. Our framework builds upon these advances, but differs in two key aspects:

1. We extend non-Lambertian scene modeling to the 3D object removal task, ensuring consistency of global lighting effects after object removal. To address the unique challenge of inpainting non-Lambertian surfaces, we further introduce a dedicated intrinsic-space inpainting module.
2. We further capture general glossy reflection effects through a screen-space filter, whereas prior work [14, 21] was primarily restricted to modeling ideal specular reflections.

S5. More ablation studies

In this section, we further conduct ablation studies on the external priors we introduced, including the segmentation prior M_{gt} and the normal prior N_{gt}.

The ablation results in Table 3 indicate that both the segmentation prior M_{gt} and the normal prior N_{gt} play important and complementary roles. Removing M_{gt} weakens the separation between rough and specular regions, leading to noticeably degraded visual metrics. The visualization in Fig. 3 further confirms this effect: without M_{gt}, the model struggles to learn correct region segmentation and fails to distinguish glossy from rough areas reliably. Moreover, removing N_{gt} reduces geometric accuracy and shading consistency, leading to performance drops across all metrics. As shown in Fig. 4, N_{gt} provides a strong geometric prior for scene initialization; without it, the model reconstructs

Table 3. Ablation study of the precomputed segmentation map M_{gt} and the normal prior N_{gt}. The best/second-best results are marked as red/gold.

Component	PSNR	SSIM	LPIPS	M-LPIPS	FID	M-FID
w/o M_{gt}	29.38	0.939	0.050	0.090	31.3	76.8
w/o N_{gt}	29.77	0.937	0.053	0.084	33.4	74.1
Full model	31.91	0.947	0.039	0.060	23.4	65.0

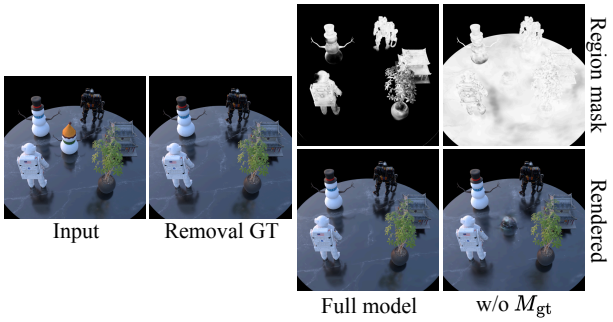

Figure 3. Ablation study of the segmentation prior M_{gt}.

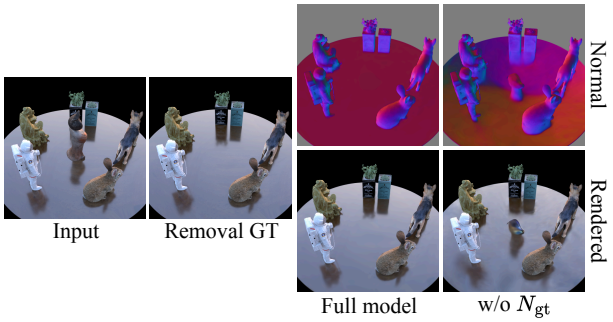

Figure 4. Ablation study of the normal prior N_{gt}.

inaccurate normals, which in turn produce incorrect shading and degraded outputs. The full model achieves the best results, demonstrating that combining both priors enables more accurate intrinsic decomposition and more faithful object removal.

S6. More visualization results

In this section, we present additional visualization results.

We further evaluate our method on the Mip-NeRF 360 [1] and Ref-Real [15] datasets. Specifically, we select the *garden* scene from Mip-NeRF 360 and the *garden spheres* scene from Ref-Real, both of which contain non-Lambertian surfaces (e.g., a glossy desktop and reflective spheres). We choose target objects in each scene and process the data using the same preprocessing pipeline as the GOR-IS-Real dataset. We then apply our method to remove the objects. As shown in Fig. 5, our approach achieves physically consistent object removal.

We present the intermediate results of scene decomposition in Fig. 6. The visualizations include the ground-truth (GT) images, rendered images, decomposed material properties (diffuse reflection, Fresnel, roughness, and normal), as well as the glossy reflection components and the region masks.

Visual comparisons with Infusion [6] and SPIn-NeRF [7] on the GOR-IS-Synthetic and GOR-IS-Real datasets are shown in Fig. 7. And visual comparisons on the SPIn-NeRF dataset [7] are provided in Fig. 8 and Fig. 9.

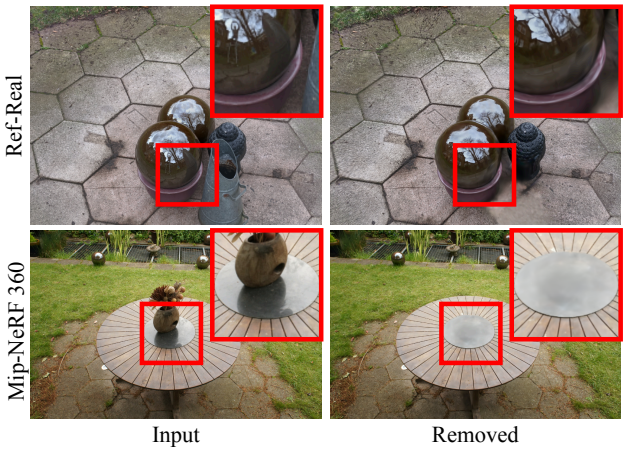

Figure 5. Visual evaluations on the Mip-NeRF 360 and Ref-real datasets.

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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022. 1, 7

[2] Chun Gu and Li Zhang. 3d gaussian ray tracer, 2024. 2

[3] Sheng-Yu Huang, Zi-Ting Chou, and Yu-Chiang Frank Wang. 3d gaussian inpainting with depth-guided cross-view consistency. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26704–26713, 2025. 3, 5

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

[5] Ruofan Liang, Zan Gojcic, Huan Ling, Jacob Munkberg, Jon Hasselgren, Chih-Hao Lin, Jun Gao, Alexander Keller, Nandita Vijaykumar, Sanja Fidler, et al. Diffusion renderer: Neural inverse and forward rendering with video diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26069–26080, 2025. 4

[6] Zhiheng Liu, Hao Ouyang, Qiuyu Wang, Ka Leong Cheng, Jie Xiao, Kai Zhu, Nan Xue, Yu Liu, Yujun Shen, and Yang Cao. Infusion: Inpainting 3d gaussians via learning

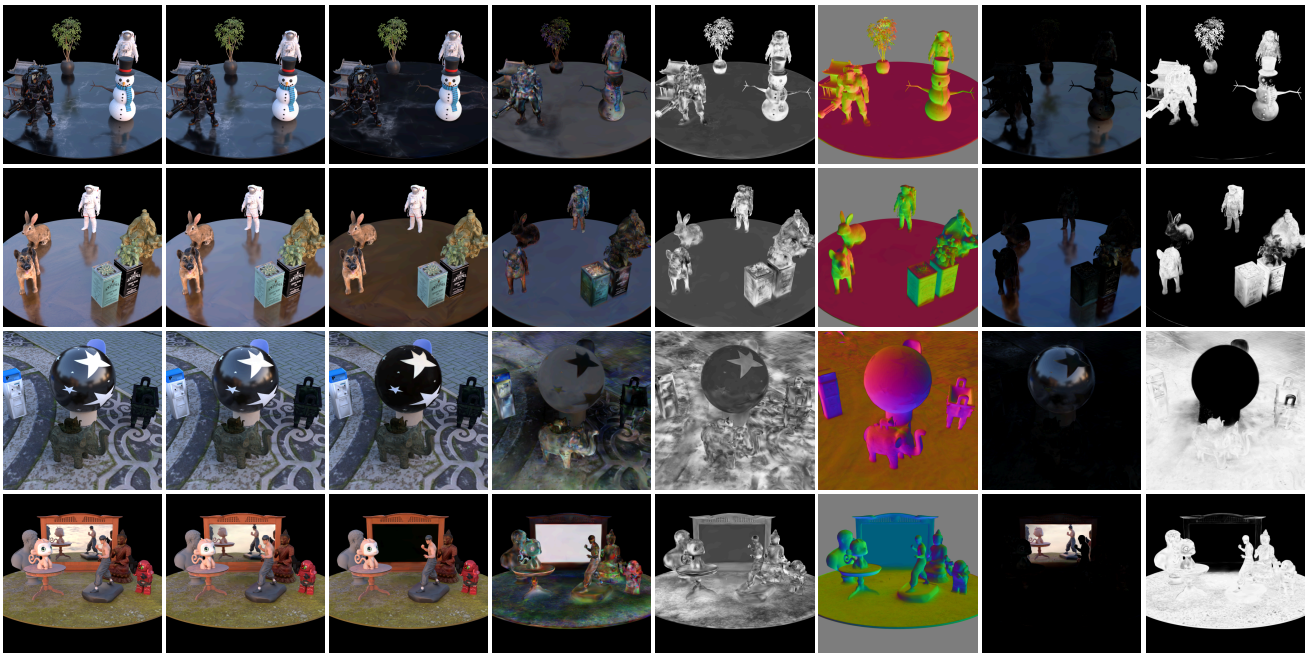

Figure 6. Visualizations of scene decomposition.

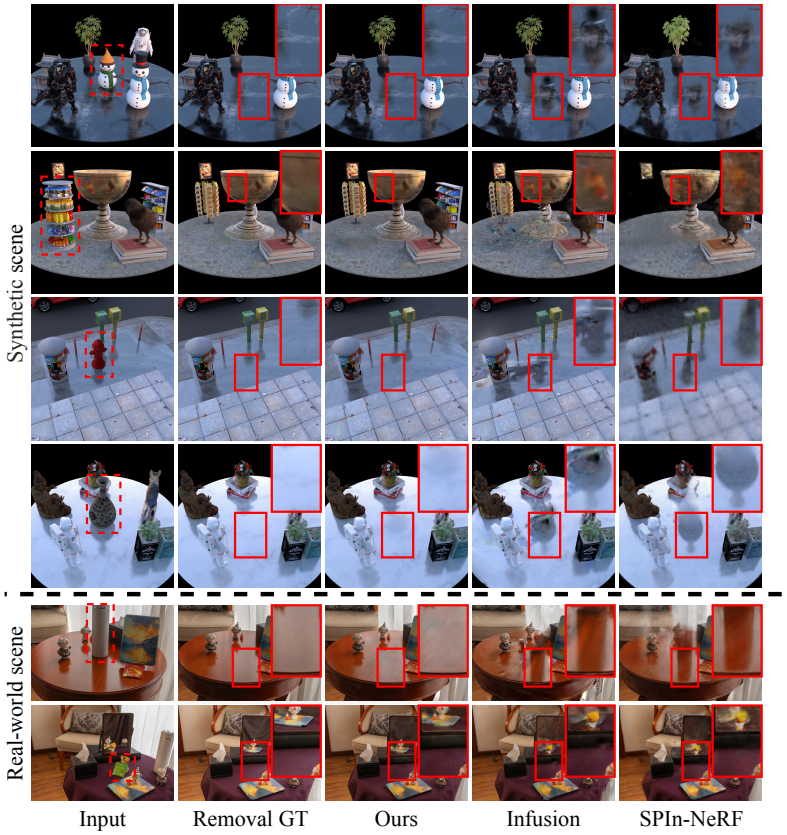

Figure 7. Visual comparisons with Infusion [6] and SPIn-NeRF [7] on the GOR-IS-Synthetic and GOR-IS-Real datasets.

Figure 8. Visual comparisons with baseline methods on the SPIn-NeRF dataset.

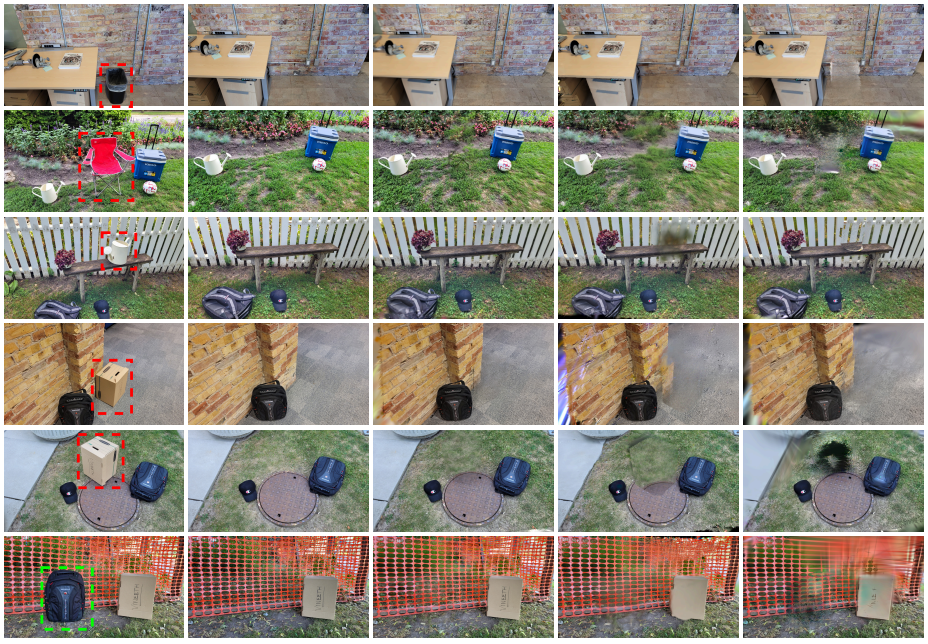

Figure 9. Visual comparisons with baseline methods on the SPIn-NeRF dataset.

depth completion from diffusion prior. *arXiv preprint arXiv:2404.11613*, 2024. 5, 7, 8

[7] Ashkan Mirzaei, Tristan Aumentado-Armstrong, Konstantinos G Derpanis, Jonathan Kelly, Marcus A Brubaker, Igor Gilitschenski, and Alex Levinshtein. Spin-nerf: Multiview segmentation and perceptual inpainting with neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20669–20679, 2023. 1, 5, 7, 8

[8] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning*

(ICML-10), pages 807–814, 2010. 2

[9] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 2, 5

[10] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 3

[11] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[12] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 5

[13] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2149–2159, 2022. 3, 6

[14] Jiajun Tang, Fan Fei, Zhihao Li, Xiao Tang, Shiyong Liu, Youyu Chen, Binxiao Huang, Zhenyu Chen, Xiaofei Wu, and Boxin Shi. Spectre-gs: Modeling highly specular surfaces with reflected nearby objects by tracing rays in 3d gaussian splatting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 16133–16142, 2025. 1, 4, 5, 6

[15] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5481–5490. IEEE, 2022. 7

[16] Shaoxiang Wang, Shihong Zhang, Christen Millerdurai, Rüdiger Westermann, Didier Stricker, and Alain Pagani. Inpaint360gs: Efficient object-aware 3d inpainting via gaussian splatting for 360deg scenes. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 117–127, 2026. 3

[17] Yuxin Wang, Qianyi Wu, Guofeng Zhang, and Dan Xu. Learning 3d geometry and feature consistent gaussian splatting for object removal. In *European Conference on Computer Vision*, pages 1–17. Springer, 2024. 5

[18] Chung-Ho Wu, Yang-Jung Chen, Ying-Huan Chen, Jie-Ying Lee, Bo-Hsu Ke, Chun-Wei Tuan Mu, Yi-Chuan Huang, Chin-Yang Lin, Min-Hung Chen, Yen-Yu Lin, and Yu-Lun Liu. Aurafusion360: Augmented unseen region alignment for reference-based 360deg unbounded scene inpainting. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pages 16366–16376, 2025. 5

[19] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. In *European conference on computer vision*, pages 162–179. Springer, 2024. 5

[20] Baowen Zhang, Chuan Fang, Rakesh Shrestha, Yixun Liang, Xiaoxiao Long, and Ping Tan. Rade-gs: Rasterizing depth in gaussian splatting. *arXiv preprint arXiv:2406.01467*, 2024. 4, 5

[21] Wenyuan Zhang, Jimin Tang, Weiqi Zhang, Yi Fang, Yu-Shen Liu, and Zhizhong Han. MaterialRefGS: Reflective gaussian splatting with multi-view consistent material inference. In *Advances in Neural Information Processing Systems*, 2025. 6