

DIFFUSIONRENDERER: Neural Inverse and Forward Rendering with Video Diffusion Models

Ruofan Liang^{1,2,3,*}, Zan Gojic¹, Huan Ling^{1,2,3}, Jacob Munkberg¹, Jon Hasselgren¹, Zhi-Hao Lin^{1,4}, Jun Gao^{1,2,3}, Alexander Keller¹, Nandita Vijaykumar^{2,3}, Sanja Fidler^{1,2,3}, Zian Wang^{1,2,3,*}
¹NVIDIA ²University of Toronto ³Vector Institute ⁴University of Illinois Urbana-Champaign

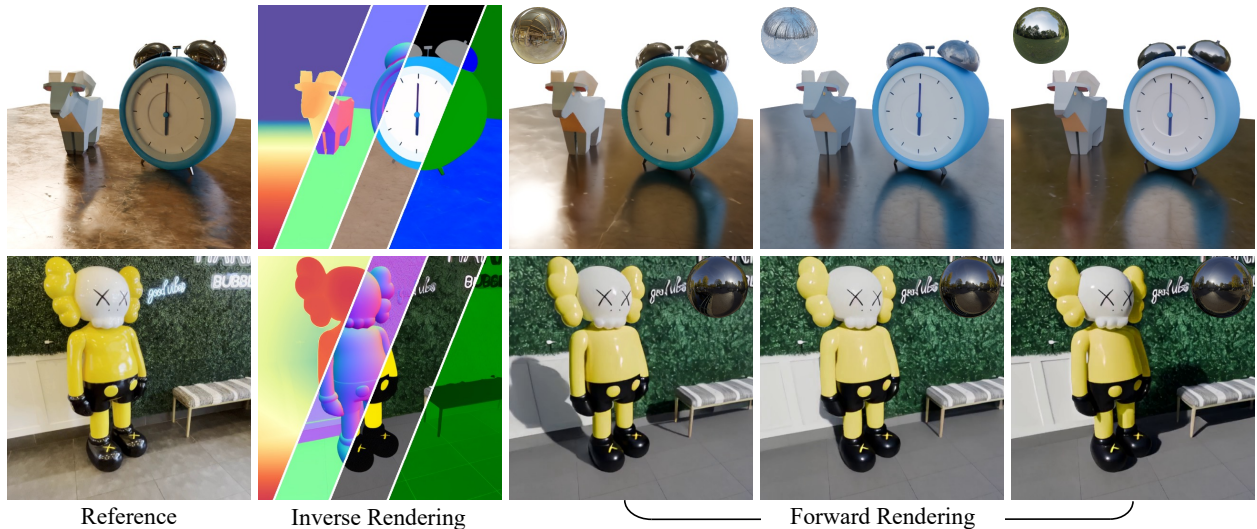


Figure 1. We present DIFFUSIONRENDERER, a general-purpose method for both neural inverse and forward rendering. From input images or videos, it accurately estimates geometry and material buffers, and generates photorealistic images under specified lighting conditions, offering fundamental tools for image editing applications.

Abstract

Understanding and modeling lighting effects are fundamental tasks in computer vision and graphics. Classic physically-based rendering (PBR) accurately simulates the light transport, but relies on precise scene representations—explicit 3D geometry, high-quality material properties, and lighting conditions—that are often impractical to obtain in real-world scenarios. Therefore, we introduce DIFFUSIONRENDERER, a neural approach that addresses the dual problem of inverse and forward rendering within a holistic framework. Leveraging powerful video diffusion model priors, the inverse rendering model accurately estimates G-buffers from real-world videos, providing an interface for image editing tasks, and training data for the rendering model. Conversely, our rendering model generates photorealistic images from G-buffers without explicit light transport simulation. Specifically, we first train a video diffusion model for inverse rendering on synthetic data, which generalizes well to real-world videos and allows us to auto-label diverse real-world videos. We

then co-train our rendering model using both synthetic and auto-labeled real-world data. Experiments demonstrate that DIFFUSIONRENDERER effectively approximates inverse and forwards rendering, consistently outperforming the state-of-the-art. Our model enables practical applications from a single video input—including relighting, material editing, and realistic object insertion.

1. Introduction

Understanding and modeling light transport forms the basis of Physically Based Rendering (PBR) [55]. Modern path tracing algorithms, as regularly used in the gaming and movie industries, simulate light transport to render images that cannot be distinguished from photographs. The quality of such PBR-rendered images heavily depends on the accuracy and realism of the scene’s surface geometry, material properties, and lighting representations. Such a scene description is either designed by artists (synthetic scenes) or reconstructed from data—also known as the inverse rendering problem [2, 3]. Inverse rendering has been extensively

studied, particularly for applications like relighting and object insertion into real-world scenes [37, 63, 71]. However, acquiring high quality surface and material representations is challenging in real-world scenarios, limiting the practicality of PBR methods (Fig. 2).

While physically-based rendering and inverse rendering are usually considered separately, we propose to consider them jointly. Our approach draws inspiration from the success of large-scale generative models [6, 61], which “render” photorealistic images from simple text prompts without any explicit understanding of PBR. These models learn the underlying distribution of real-world images from a vast amounts of data, implicitly capturing the complex lighting effects.

Specifically, we propose DIFFUSIONRENDERER, a general-purpose neural rendering engine that can synthesize light transport simulation—such as shadows and reflections—by leveraging the powerful priors of video diffusion models. Conditioned on input geometry, material buffers, and environment map light source, DIFFUSIONRENDERER acts as a neural approximation of path-traced shading. DIFFUSIONRENDERER is designed to remain faithful to the conditioning signals, while adhering to the distribution of real-world images. As a result, we bypass the need for precise scene representations and description, as our model learns to handle imperfections in the input data.

Training such a model requires some amount of high quality and diverse data, including data with noisy conditions to ensure robustness. Therefore, we first train an *inverse renderer*, a video diffusion model to map input RGB videos to intrinsic properties. Although trained solely on synthetic data, the inverse rendering model generalizes robustly to real-world scenarios. We then use it to generate “pseudo-labels” for diverse real-world videos. Combining both real-world auto-labeled data and synthetic data, we train our *forward renderer* video diffusion model.

DIFFUSIONRENDERER outperforms state-of-the-art methods and effectively approximates the complex functionalities of inverse and forward rendering, allowing us to relight images and videos across diverse scenes and to synthesize consistent shadows and reflections without explicit path tracing and 3D scene representation. Our model can relight any scene from only a single video input, and provides fundamental tools for editing tasks such as material editing and realistic object insertion. To summarize:

- We develop a state-of-the-art inverse rendering method for videos of synthetic and real-world scenes.
- We repurpose a video diffusion model as a neural rendering engine that can synthesize photorealistic images and videos conditioned on noisy G-buffers.
- From a single video input, DIFFUSIONRENDERER enables relighting, material editing, and virtual object insertion in a unified framework, expanding the possibilities for real-world neural rendering applications.

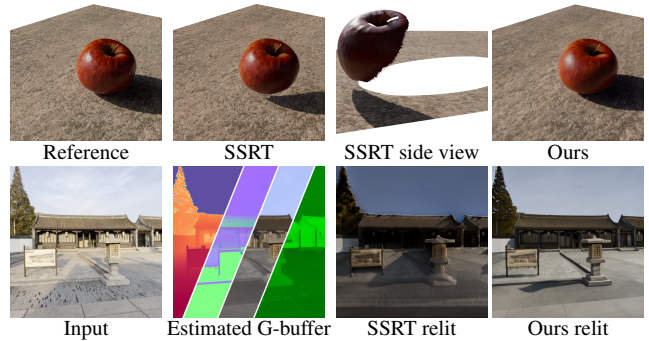


Figure 2. Classic PBR relies on explicit 3D geometry, e.g., meshes. When it is not available, screen space ray tracing (SSRT) struggles to accurately represent shadows and reflections (top). PBR is also sensitive to errors in G-buffers – SSRT with estimated G-buffers from inverse rendering models often fails to deliver quality results (bottom). DIFFUSIONRENDERER bypasses these issues, producing photorealistic results without 3D geometry or perfect G-buffers.

2. Related Work

Neural rendering refers to methods that replace or extend traditional rendering pipelines by neural networks. For example, Deep Shading [52] replaces traditional deferred shading [13] by a CNN to render images with ambient occlusion, global illumination, and depth-of-field from G-buffers. More recently, RGB \leftrightarrow X [78] trains image diffusion models to both estimate a G-buffer from an image and to render an image from a G-buffer. We extend this approach to video diffusion models and provide a novel approach for neural relighting that does not require an irradiance estimate. Other approaches fit rendered data using neural models or introduce neural components into an existing renderer with focus on approximating light transport [24, 27] or radiance caching [21, 49]. A plethora of works on neural and inverse rendering involve volumetric 3D scene representations in the form of NeRF [48] or 3D Gaussian Splats [31]. We refer to [66] for an overview. While providing photo-real view interpolation, these approaches typically bake radiance, and have limited editing capabilities. In contrast, we explicitly target an intermediate scene representation in the form of traditional, easy-to-edit, G-buffers with separate lighting.

Inverse rendering is a fundamental task first formalized in the 1970s [4], aiming to estimate intrinsic scene properties, like geometry, materials, and lighting from input images. Early methods designed hand-crafted priors within an optimization framework [2, 4, 9, 19, 35, 83], typically focusing on low-order effects. These methods lead to errors when the hand-crafted priors do not match reality. Recently, supervised and self-supervised learning has been extensively studied [5, 7, 34, 36–38, 63, 69, 70, 72, 75]. The resulting algorithms are often data-hungry and specific to a certain task or domain. Acquiring sufficient and diverse training data poses a challenge. Recent advances in large text-to-image diffusion models provide new deep learning tools for

inverse rendering [32, 40, 57, 78] resulting in much higher reconstruction quality. Still, the quality is not enough to power physically based rendering pipelines.

Relighting focuses on modifying the lighting conditions of a scene given captured images or videos. Recent methods reconstruct 3D scene representations from multi-view images, performing explicit inverse rendering to recover material properties and enable relighting [8, 11, 20, 25, 39, 41, 50, 62, 64, 71, 74, 79, 80, 82]. These methods often optimize for each scene individually, and their quality may be affected by practical issues such as single-illumination capture, large scene scale, and dynamic content. Learning-based methods that train across multiple scenes have explored latent feature learning [44, 84] and often incorporate neural rendering modules that utilize PBR buffers as inductive priors [18, 33, 53, 56]. To improve relighting quality, recent approaches [26, 33, 58, 77] leverage diffusion models. With very few multi-illumination datasets [51], existing methods often are specialized to a domain, such as portraits, objects, and outdoor scenes, and remain data-hungry.

3. Preliminaries

Physically-based rendering (PBR) is concerned with the simulation of how the incoming radiance contributes to the outgoing radiance

$$L_o(\mathbf{p}, \boldsymbol{\omega}_o) = \int_{\Omega} f_r(\mathbf{p}, \boldsymbol{\omega}_o, \boldsymbol{\omega}_i) L_i(\mathbf{p}, \boldsymbol{\omega}_i) |\mathbf{n} \cdot \boldsymbol{\omega}_i| d\boldsymbol{\omega}_i, \quad (1)$$

at a surface point \mathbf{p} in direction $\boldsymbol{\omega}_o$. The integral over the hemisphere Ω considers the BRDF $f_r(\mathbf{p}, \boldsymbol{\omega}_o, \boldsymbol{\omega}_i)$, incoming radiance $L_i(\mathbf{p}, \boldsymbol{\omega}_i)$, and a cosine factor $|\mathbf{n} \cdot \boldsymbol{\omega}_i|$ for the angle between the normal \mathbf{n} and incoming light. It is evaluated by Monte Carlo methods [55, 67], with meticulously designed BRDF models [10, 12, 68] that approximate real surfaces.

Video diffusion models (VDMs). A diffusion model learns to approximate a data distribution $p_{\text{data}}(\mathbf{I})$ via iterative denoising [16, 22, 65]. Most VDMs operate in a compressed, lower-dimensional latent space [1, 6]. Given an RGB video $\mathbf{I} \in \mathbb{R}^{F \times H \times W \times 3}$, consisting of F frames at resolution $H \times W$, a pre-trained VAE encoder \mathcal{E} first encodes the video into a latent representation $\mathbf{z} = \mathcal{E}(\mathbf{I}) \in \mathbb{R}^{F' \times h \times w \times C}$. The final video $\hat{\mathbf{I}}$ is then reconstructed by decoding \mathbf{z} with a pre-trained VAE decoder \mathcal{D} . Both training and inference stages of the VDM are conducted in this latent space. In this work, we build on Stable Video Diffusion [6], which compresses the video only along the spatial dimensions: $F' = F$, $C = 4$, $h = \frac{H}{8}$, and $w = \frac{W}{8}$.

To train the VDM, noisy versions $\mathbf{z}_\tau = \alpha_\tau \mathbf{z}_0 + \sigma_\tau \epsilon$ are constructed by adding a Gaussian noise ϵ with the noise schedule provided by α_τ and σ_τ following EDM [29]. The diffusion model parameters θ of the denoising function \mathbf{f}_θ are optimized using the denoising score matching objective [29].

Once trained, iteratively applying \mathbf{f}_θ to a sample of Gaussian noise will produce a sample of $p_{\text{data}}(\mathbf{I})$.

Conditioning in VDMs. Two common approaches to inject conditions into VDMs are: (i) concatenating condition channels with image latents \mathbf{z}_τ , which is often used for pixel-wise conditions [6, 30, 32, 78], and (ii) injecting conditions through cross-attention layers [6, 61], which is often used for semantic features such as the CLIP embedding [59]. Note that our method is compatible with any standard VDMs and does not depend on specific architectural details.

4. Method

DIFFUSIONRENDERER is a unified framework comprising two video diffusion models designed for the dual tasks of neural forward and inverse rendering. As illustrated in Fig. 3, the *neural forward renderer* (Sec. 4.1) approximates physically based light transport (Eq. 1), transforming G-buffers [52] and lighting into a photorealistic video. The *neural inverse renderer* (Sec. 4.2) reconstructs geometry and material buffers from input video. The neural forward and inverse renderers are based on pre-trained video diffusion models and fine-tuned for conditional generation [30, 32, 78].

Data is a critical aspect of learning-based methods. We describe our data curation workflow and synthetic-real joint training strategies in Sec. 4.3 and Sec. 4.4. Finally, we discuss image editing applications in Sec. 4.5.

4.1. Neural Forward Rendering

We formulate neural forward rendering as a conditional generation task, producing photorealistic images given geometry, materials, and lighting as conditions. By approximating light transport simulation in a data-driven manner, the model requires neither classic 3D geometry nor explicit path tracing, thus reducing the constraints in real-world applications.

Geometry and material conditions. Similar to the G-buffers in rendering system based on deferred shading [13], we use per-pixel scene attribute maps to represent scene geometry and materials. Specifically, we use surface normals $\mathbf{n} \in \mathbb{R}^{F \times H \times W \times 3}$ in camera space and relative depth $\mathbf{d} \in \mathbb{R}^{F \times H \times W \times 1}$ normalized to $[-1, 1]$ to represent scene geometry. For materials, we use base color $\mathbf{a} \in \mathbb{R}^{F \times H \times W \times 3}$, roughness $\mathbf{r} \in \mathbb{R}^{F \times H \times W \times 1}$, and metallic $\mathbf{m} \in \mathbb{R}^{F \times H \times W \times 1}$ following the Disney BRDF [10].

Lighting conditions. Lighting is represented by an environment maps $\mathbf{E} \in \mathbb{R}^{F \times H_{\text{env}} \times W_{\text{env}} \times 3}$, which are panoramic images that capture the lighting intensity from all directions over the sphere. These environment maps are encoded in high dynamic range (HDR), while the VAEs used in typical latent diffusion models are designed for pixel values between -1 and 1 . To address this discrepancy, similar to the light representation in Neural Gaffer [26], we first apply Reinhard tonemapping to convert HDR environment map into an LDR image \mathbf{E}_{ldr} . To more effectively represent HDR values,

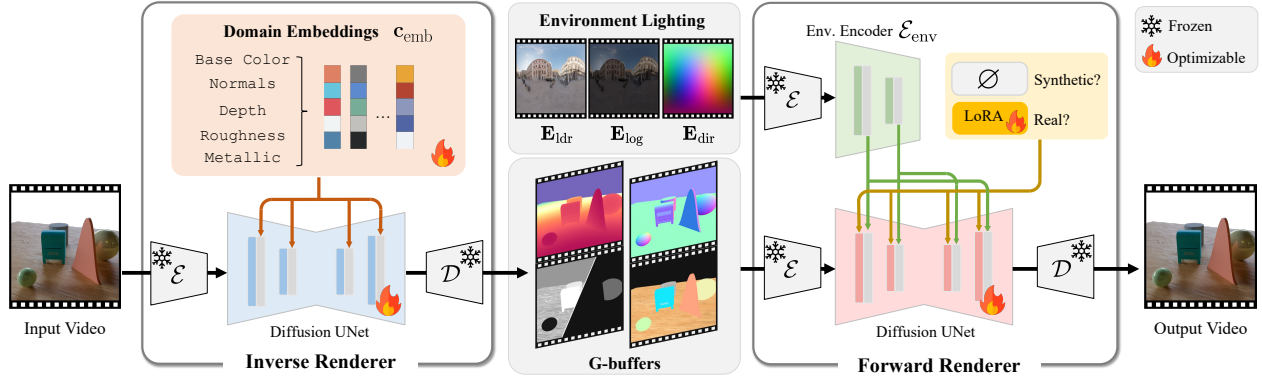


Figure 3. **Method overview.** Given an input video, the neural inverse renderer estimates geometry and material properties per pixel. It generates one scene attribute at a time, with the domain embedding indicating the target attributes to generate (Sec. 4.2). Conversely, the neural forward renderer produces photorealistic images given lighting information, geometry, and material buffers. The lighting condition is injected into the base video diffusion model through cross-attention layers (Sec. 4.1). During joint training with both synthetic and real data, we use an optimizable LoRA for real data sources (Sec. 4.4).

particularly for light sources with high-intensity peaks, we compute $\mathbf{E}_{\log} = \log(\mathbf{E} + 1)/E_{\max}$ where the light intensity values are mapped to logarithm space that is closer to human perception and normalized by max log intensity E_{\max} . Additionally, we also compute a directional encoding image, $\mathbf{E}_{\text{dir}} \in \mathbb{R}^{F \times H_{\text{env}} \times W_{\text{env}} \times 3}$, where each pixel is represented by a unit vector indicating its direction in the camera coordinate system. The resulting lighting encodings used by the model consist of three panoramic images: $\{\mathbf{E}_{\text{ldr}}, \mathbf{E}_{\log}, \mathbf{E}_{\text{dir}}\}$.

Model architecture. Our models are based on Stable Video Diffusion [6], an image-to-video diffusion model with its core architecture including a VAE encoder-decoder pair $\{\mathcal{E}, \mathcal{D}\}$, and a UNet-based denoising function \mathbf{f}_{θ} .

We use the VAE encoder \mathcal{E} to separately encode each G-buffer from $\{\mathbf{n}, \mathbf{d}, \mathbf{a}, \mathbf{r}, \mathbf{m}\}$ into the latent space and concatenate them to produce the pixel-aligned scene attribute latent map $\mathbf{g} = \{\mathcal{E}(\mathbf{n}), \mathcal{E}(\mathbf{d}), \mathcal{E}(\mathbf{a}), \mathcal{E}(\mathbf{r}), \mathcal{E}(\mathbf{m})\} \in \mathbb{R}^{F \times h \times w \times 20}$.

Environment maps are usually in equi-rectangular projection and are not pixel-aligned with the generated images, thus requiring extra consideration. Prior works explored directly concatenating environment maps to the image latents [26] or concatenate split-sum shading buffers [15], which we also experimented with, but found suboptimal (Table 1). Instead, we take the cross-attention layers which originally operate on the text/image CLIP features, and re-purpose them for lighting conditions. To preserve spatial details of the environment maps, we generalize the conditional signals to a list of multi-resolution feature maps.

Specifically, we first pass the environment map information through VAE encoder \mathcal{E} to obtain $\mathbf{h}_{\mathbf{E}} = \{\mathcal{E}(\mathbf{E}_{\text{ldr}}), \mathcal{E}(\mathbf{E}_{\log}), \mathcal{E}(\mathbf{E}_{\text{dir}})\} \in \mathbb{R}^{F \times h_{\text{env}} \times w_{\text{env}} \times 12}$. We additionally use an environment map encoder \mathcal{E}_{env} to further operate on $\mathbf{h}_{\mathbf{E}}$. \mathcal{E}_{env} is the simplified encoder part of diffusion UNet with attention and temporal layers removed. It contains several convolutional layers to downsample and extract

K levels of multi-resolution features as lighting conditions:

$$\mathbf{c}_{\text{env}} := \{\mathbf{h}_{\text{env}}^i\}_{i=1}^K = \mathcal{E}_{\text{env}}(\mathbf{h}_{\mathbf{E}}) \quad (2)$$

As a result, the diffusion UNet \mathbf{f}_{θ} takes the noisy latent \mathbf{z}_{τ} and G-buffer latent \mathbf{g} as pixel-wise input. At each UNet level k , the diffusion UNet *queries* the latent environment map features at the corresponding level $\mathbf{h}_{\text{env}}^k$, and aggregates based on its *keys* and *values*. Through the multi-level self-attention and cross-attention layers, the diffusion model is given the capacity to learn to shade G-buffers with lighting. During inference, the diffusion target can be computed as $\mathbf{f}_{\theta}(\mathbf{z}_{\tau}; \mathbf{g}, \mathbf{c}_{\text{env}}, \tau)$ to produce photorealistic images with iterative denoising.

4.2. Neural Inverse Rendering

We similarly formulate inverse rendering as a conditional generation task. Given an input video \mathbf{I} as a condition, the inverse renderer estimates scene attribute maps $\{\mathbf{n}, \mathbf{d}, \mathbf{a}, \mathbf{r}, \mathbf{m}\}$ which are the G-buffers used by the forward renderer.

Model architecture. The input video \mathbf{I} is encoded into latent space $\mathbf{z} = \mathcal{E}(\mathbf{I})$, and concatenated with the noisy G-buffer latent, which we denote as $\mathbf{g}_{\tau} = \alpha_{\tau}\mathbf{g}_0 + \sigma_{\tau}\epsilon$.

Given an input video, the inverse renderer generates all five attributes $\{\mathbf{n}, \mathbf{d}, \mathbf{a}, \mathbf{r}, \mathbf{m}\}$ using the same model. To preserve the high-quality generation and maximally leverage the diffusion model pre-trained knowledge, each attribute is generated in a dedicated pass, instead of generating all at once. We follow prior works [17, 45, 78] and use a domain embedding to indicate to the model which attribute should be generated. Specifically, we introduce an optimizable domain embedding $\mathbf{c}_{\text{emb}} \in \mathbb{R}^{K_{\text{emb}} \times C_{\text{emb}}}$, where $K_{\text{emb}} = 5$ is the number of buffers and C_{emb} is the dimension of the embedding vector. We re-purpose the cross-attention layers with image CLIP features to take domain embeddings. When estimating an attribute indexed by P , we feed its embedding $\mathbf{c}_{\text{emb}}^P$ as a condition and predict the diffusion target with $\mathbf{f}_{\theta}(\mathbf{g}_{\tau}^P; \mathbf{z}, \mathbf{c}_{\text{emb}}^P, \tau)$.

4.3. Data Strategy

Synthetic data curation. To train our models, we require high-quality video data with paired ground-truth for material, geometry, and lighting information. Specifically, each video data sample should include paired frames of RGB, base color, roughness, metallic, normals, depth, and environment map: $\{\mathbf{I}, \mathbf{a}, \mathbf{r}, \mathbf{m}, \mathbf{n}, \mathbf{d}, \mathbf{E}\}$. These buffers are typically only available in synthetic data, and most existing public datasets contain only a subset of them.

To address the data scarcity, we designed a synthetic data generation workflow to produce a large amount of high-quality data covering diverse and complex lighting effects. We start by curating a collection of 3D assets, PBR materials, and HDRI environment maps. We use 36,500 3D assets from Objaverse LVIS split. For materials and lighting, we collected 4,260 high-quality PBR material maps, and 766 HDR environment maps from publicly available resources.

In each scene, we place a plane with a randomly selected PBR material, and sample up to three 3D objects, and place them on the plane. We perform collision detection to avoid intersecting objects. We also place up to three primitives (cube, sphere, and cylinder) with randomized shape and materials to cover complex lighting effects such as inter-reflections. A randomly selected HDR environment map illuminates the scene. We generate motions including camera orbits, camera oscillation, lighting rotation, object rotation and translation.

We use a custom path tracer based on OptiX [54] to render the videos. In total, we generate 150,000 videos with paired ground-truth G-buffers and environment maps, at 24 frames per video in 512x512 resolution. This dataset can be used to train both rendering and inverse rendering models.

Real world auto-labeling. Synthetic data provides accurate supervision signals, and when combined with powerful image diffusion models, it demonstrates impressive generalization to unseen domains for inverse rendering tasks [17, 30]. However, when it comes to training the forward rendering model, synthetic data alone is insufficient. Since the output of the forward renderer is an RGB video, training only on synthetic renderings biases the model toward synthetic visual styles. Compared to inverse rendering, we observe a much more significant domain gap in complex real-world scenes for forward rendering tasks (Fig. 7).

Acquiring real-world data with paired geometry, material, and lighting ground truth requires complex and impractical capturing setups. Based on the observation that our inverse rendering model generalizes to real-world videos, we apply it to automatically label real-world videos. Specifically, we use the DL3DV10k [43] dataset, which is a large-scale dataset consisting of 10,510 videos featuring diverse real-world environments. We use our inverse rendering model (Sec. 4.2) to generate G-buffer labels and use an off-the-shelf method DiffusionLight [57] to estimate environment

maps. Each video is divided into 15 segments, resulting in around 150,000 real-world video samples with auto-labeled geometry, material, and lighting attributes.

4.4. Training pipeline

Neural inverse renderer. We first co-train the inverse rendering model on the combination of the curated synthetic video dataset and public image intrinsics datasets InteriorVerse [85] and HyperSim [60]. For image datasets, we treat images as single-frame videos. Each data sample consists of a video \mathbf{I} , an attribute index P , and the scene attribute map \mathbf{s}^P . The target latent variable is the latents of the scene attribute $\mathbf{g}_0^P := \mathcal{E}(\mathbf{s}^P)$, and noise is added to \mathbf{g}_0^P to produce \mathbf{g}_τ^P . The model is trained using the objective function [29]:

$$\mathcal{L}(\theta, \mathbf{c}_{\text{emb}}) = \|\mathbf{f}_\theta(\mathbf{g}_\tau^P; \mathbf{z}, \mathbf{c}_{\text{emb}}^P, \tau) - \mathbf{g}_0^P\|_2^2. \quad (3)$$

We fine-tune the diffusion model parameters θ and domain embeddings \mathbf{c}_{emb} , while keeping the latent encoder \mathcal{E} and decoder \mathcal{D} frozen. Once trained, the inverse renderer is used to auto-label real-world videos, generating training data for the forward renderer.

Environment map encoder pre-training. Following the approach of latent diffusion models [61], we pre-train the environment map encoder \mathcal{E}_{env} along with a decoder \mathcal{D}_{env} using an L2 image reconstruction objective on environment maps, similar to an auto-encoder. The decoder architecture is based on the UNet decoder, containing a set of upsampling layers. After training, we discard the decoder \mathcal{D}_{env} and freeze the environment map encoder \mathcal{E}_{env} while training the neural forward rendering model.

Neural forward renderer. We train our rendering model on a combination of a synthetic video dataset and real-world auto-labeled data, using paired G-buffer, lighting, and RGB videos. Although the auto-labeled real-world data is of sufficient quality, it may still contain inaccuracies. To address discrepancies between the synthetic and real-world data sources, we introduce an additional LoRA [23] with a small set of optimizable parameters $\Delta\theta$ during training on real data. We empirically find it improves the rendering quality (Fig. 7).

During training, for an RGB video \mathbf{I} , the target latent variable is defined as $\mathbf{z}_0 := \mathcal{E}(\mathbf{I})$. Noise is added to \mathbf{z}_0 to produce noisy image latent \mathbf{z}_τ . The training objective is:

$$\begin{aligned} \mathcal{L}(\theta, \Delta\theta) = & \|\mathbf{f}_\theta(\mathbf{z}_\tau^{\text{synth}}; \mathbf{g}_{\text{env}}^{\text{synth}}, \mathbf{c}_{\text{env}}^{\text{synth}}, \tau) - \mathbf{z}_0^{\text{synth}}\|_2^2 + \\ & \|\mathbf{f}_{\theta+\Delta\theta}(\mathbf{z}_\tau^{\text{real}}; \mathbf{g}^{\text{real}}, \mathbf{c}_{\text{env}}^{\text{real}}, \tau) - \mathbf{z}_0^{\text{real}}\|_2^2 \end{aligned} \quad (4)$$

4.5. Editing Applications

Our proposed framework provides fundamental solutions for inverse and forward rendering, enabling photorealistic image editing applications through a three-step process: neural inverse rendering, G-buffer and lighting editing, and neural

	<i>SyntheticObjects</i>			<i>SyntheticScenes</i>		
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
SSRT	29.4	0.951	0.037	24.8	0.899	0.113
SplitSum [28]	28.7	0.951	0.038	23.1	0.883	0.116
RGB \leftrightarrow X [78]	25.2	0.896	0.077	18.5	0.645	0.302
DiLightNet [77]	26.6	0.914	0.067	20.7	0.630	0.300
Ours	28.3	0.935	0.048	26.0	0.780	0.201
Ours (image)	27.4	0.916	0.062	25.4	0.760	0.215
Ours (w/o env. encoder)	27.8	0.927	0.057	25.3	0.756	0.237
Ours (+ shading cond.)	28.7	0.930	0.056	25.6	0.761	0.245

Table 1. Quantitative evaluation of neural rendering.

	<i>SyntheticObjects</i>			<i>SyntheticScenes</i>		
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
DiLightNet [77]	23.79	0.872	0.087	18.88	0.576	0.344
Neural Gaffer [26]	<u>26.39</u>	<u>0.903</u>	<u>0.086</u>	<u>20.75</u>	<u>0.633</u>	<u>0.343</u>
Ours	27.50	0.918	0.067	24.63	0.756	0.257

Table 2. Quantitative evaluation of relighting.

rendering. For the example of relighting, given a video \mathbf{I} as input, the inverse rendering model estimates the G-buffers

$$\{\hat{\mathbf{n}}, \hat{\mathbf{d}}, \hat{\mathbf{a}}, \hat{\mathbf{r}}, \hat{\mathbf{m}}\} = \text{InverseRenderer}(\mathbf{I}). \quad (5)$$

With a user-specified target environment map \mathbf{E}_{tgt} , the rendering model produces relit videos

$$\hat{\mathbf{I}}_{\text{tgt}} = \text{ForwardRenderer}(\{\hat{\mathbf{n}}, \hat{\mathbf{d}}, \hat{\mathbf{a}}, \hat{\mathbf{r}}, \hat{\mathbf{m}}, \mathbf{E}_{\text{tgt}}\}). \quad (6)$$

Similarly, editing the G-buffers and rendering the videos can enable material editing and virtual object insertion.

5. Experiments

We extensively evaluate DIFFUSIONRENDERER on a diverse range of synthetic and real-world datasets. Sec. 5.1 details our experimental settings. We compare and ablate across three main tasks: image generation from G-buffers (Sec. 5.2), inverse rendering (Sec. 5.3), and relighting (Sec. 5.4). Finally, we show applications of our method in Sec. 5.5.

5.1. Experiment Settings

We refer to model implementation details in the Supplement.

Task definitions. We evaluate our method on three fundamental tasks: forward rendering, inverse rendering, and relighting. For forward rendering, the methods take the G-buffers and lighting information $\{\mathbf{n}, \mathbf{d}, \mathbf{a}, \mathbf{r}, \mathbf{m}, \mathbf{E}\}$ as input, and output rendered images $\hat{\mathbf{I}}$. We evaluate the consistency between the rendered outputs and the ground-truth images \mathbf{I} .

For inverse rendering, each method uses RGB images \mathbf{I} as input to estimate scene attributes $\{\hat{\mathbf{a}}, \hat{\mathbf{r}}, \hat{\mathbf{m}}, \hat{\mathbf{n}}\}$, and compare against ground truth values. Our focus is primarily on the attributes related to PBR – specifically, base color, roughness, and metallic properties. We recognize dedicated works on normal and depth estimation [17, 30] and do not aim to provide an exhaustive evaluation. For relighting, each method takes RGB images and target lighting conditions $\{\mathbf{I}^{\text{src}}, \mathbf{E}^{\text{tgt}}\}$

as input, output re-lit image sequence $\hat{\mathbf{I}}^{\text{tgt}}$ under the target lighting conditions, and compare with ground truth \mathbf{I}^{tgt} .

Baselines. For forward rendering, we compare with Split Sum [28] and Screen Space Ray Tracing (SSRT). For SSRT, we extract a mesh from the depth buffer and render the mesh with material parameters from the G-buffers and a provided HDR probe in a path tracer. We additionally compare against the neural rendering components of recent state-of-the-art methods RGB \leftrightarrow X [78] and DiLightNet [77]. For inverse rendering, we compare with recent diffusion-based methods Kocsis *et al.* [32], RGB \leftrightarrow X [78] and earlier methods [5, 37, 85]. For relighting, we compare with 2D methods DiLightNet [77], Neural Gaffer [26]. We also compare with 3D reconstruction-based methods [42, 71] in Supplement.

Metrics. We use PSNR, SSIM, and LPIPS [81] for forward rendering and relighting. For inverse rendering, we evaluate albedo with PSNR and LPIPS following [32, 78]. Since albedo estimation involves scale ambiguity [19], we additionally solve and apply a three-channel scaling factor using least-squares error minimization before computing metrics, referred to as si-PSNR and si-LPIPS. We use root mean square error (RMSE) for metallic and roughness evaluation, and mean angular error for normals.

Datasets. We curate two high-quality synthetic datasets for quantitative evaluation, named *SyntheticScenes* and *SyntheticObjects*. The datasets consist of 3D assets from PolyHaven [76] and Objaverse [14] that are not included in the training data of our method or the baseline methods. *SyntheticScenes* contains 40 scenes, each featuring multiple objects arranged on a plane textured with high-quality PBR materials. Each scene is rendered into 24-frame videos under four lighting conditions, with motions such as camera orbiting and oscillation. As some baseline methods perform best with object-centric setups, we also create *SyntheticObjects*, a dataset of 30 individual objects. For each object, we render 24-frame videos under four different lighting conditions, with lighting rotated across frames.

For inverse rendering, we also evaluate on the indoor scene benchmark InteriorVerse [85]. We include qualitative comparisons on the DL3DV10k [43] dataset.

5.2. Evaluation of Forward Rendering

We compare our method with baseline methods in Table 1 and Fig. 4. For the neural methods RGB \leftrightarrow X [78] and DiLightNet [77], we use their rendering models with ground-truth G-buffers to generate the images.

Both classic PBR and neural methods perform well on the *SyntheticObjects* dataset in single-object settings but show significant quality drops on the *SyntheticScenes* dataset due to complex inter-reflections and occlusions. For example, our method exhibits a minor PSNR decrease of 2.3 dB from *SyntheticObjects* to *SyntheticScenes*, while other baselines show more substantial drops.

	Albedo				Metallic RMSE ↓	Roughness RMSE ↓	Normals Angular Error ↓
	PSNR ↑	LPIPS ↓	si-PSNR ↑	si-LPIPS ↓			
RGB↔X [78]	14.3	0.323	19.6	0.286	0.441	0.321	23.80°
Ours	<u>25.0</u>	0.205	26.7	0.204	<u>0.039</u>	0.078	<u>5.97°</u>
Ours (det.)	26.0	0.219	27.7	0.217	0.028	0.060	5.85°
Ours (image)	23.4	<u>0.213</u>	26.0	<u>0.209</u>	0.066	0.098	6.67°
Ours (image, det.)	24.8	0.231	<u>27.2</u>	0.228	0.043	<u>0.069</u>	6.17°

Table 3. Quantitative evaluation of inverse rendering on *SyntheticScenes* video dataset. image: per-frame inference as image model. det.: 1-step deterministic inference.

	PSNR ↑	SSIM ↑	LPIPS ↓
IIW [5]	9.7	0.62	0.47
Li <i>et al.</i> [37]	12.3	0.68	0.52
Zhu <i>et al.</i> [85]	15.9	0.78	0.34
Kocsis <i>et al.</i> [32]	17.4	<u>0.80</u>	0.22
RGB↔X [78]	16.4	0.78	<u>0.19</u>
Ours (image)	<u>21.9</u>	0.87	0.17
Ours (image, det.)	22.4	0.87	<u>0.19</u>

Table 4. Quantitative benchmark of albedo estimation on InteriorVerse dataset [85].

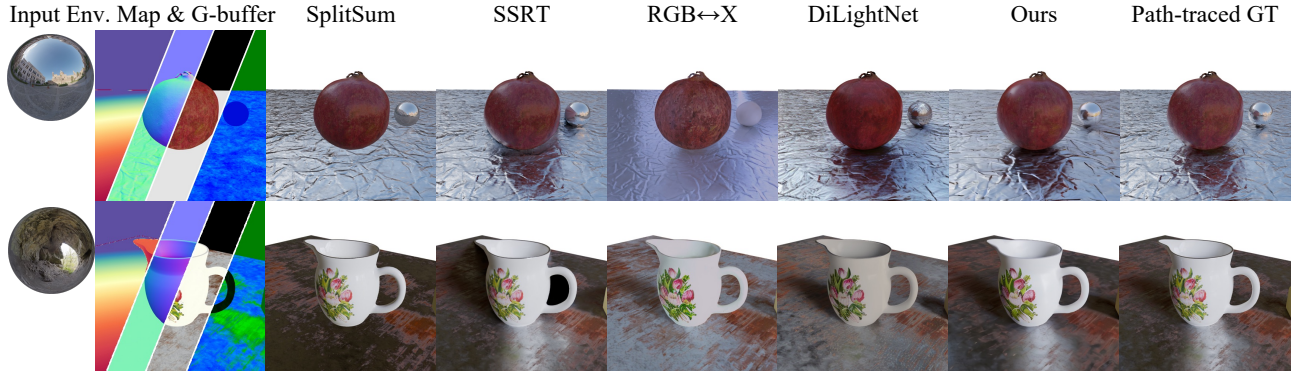


Figure 4. **Qualitative comparison of forward rendering.** Our method generates high-quality inter-reflections (*top*) and shadows (*bottom*), producing more accurate results than the neural baselines.

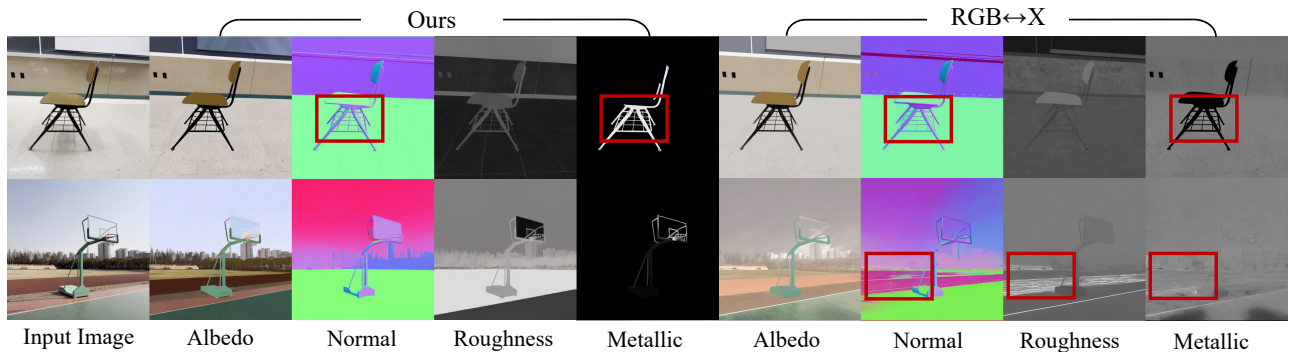


Figure 5. **Qualitative comparison of inverse rendering.** We compare with RGB↔X [78] on DL3DV10k dataset. Both methods work well on indoor scenes, while our method predicts finer details in thin structures and more accurate metallic and roughness channels (*top*), likely benefiting from our curated training data. As compared to RGB↔X, our method generalizes better to outdoor scenes (*bottom row*).

Our method consistently outperforms all neural methods on both datasets and performs comparably to classic methods. In real-world editing applications, however, these PBR techniques often face significant limitations due to missing 3D geometry and noisy G-buffers (Fig. 2). Furthermore, SplitSum does not model shadows and inter-reflections.

Ablation study. We ablate our model design choices in Table 1. While our method generalizes from images to videos, it can treat images as a special case of single-frame videos. To evaluate the benefits introduced by the video model, we compare with an ablated version of our method that performs per-frame inference (Ours, image). The video model consistently improves rendering quality across both datasets. In the ablated variant Ours (w/o env. encoder), we concatenate the VAE encoded environment maps directly to the image channels [26] rather than using a separate environment map

encoder. We demonstrate that a dedicated environment map encoder improves performance. For Ours (+ shading cond.), we include split-sum shading buffers as a conditioning input, following [15]. However, we observed no significant improvement with this addition and chose to exclude it from our final method for simplicity.

5.3. Evaluation of Inverse Rendering

We quantitatively compare our method with baseline methods on *SyntheticScenes* in Table 3 and InteriorVerse [85] benchmark in Table 4. Our methods consistently outperform baseline methods in both datasets, indicating the effectiveness of our data curation workflow and method designs. We show a qualitative comparison with RGB↔X [78] on DL3DV10k [43] dataset in Fig. 5.

Image vs. video model. Comparing Ours (image) and Ours, the video model consistently enhances the quality of



Figure 6. **Qualitative comparison of relighting.** Our method produces more accurate specular reflections compared to the baselines.

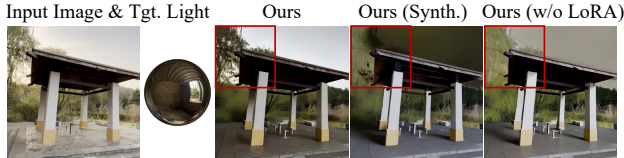


Figure 7. **Qualitative ablation of relighting.** Joint training with real-world data and adding LoRA during training significantly improve relighting quality for real-world scenes.

inverse rendering across all attributes. Notably, for properties associated with specular materials, the video model reduces RMSE by 41% for metallic (from 0.066 to 0.039) and 20% for roughness (from 0.098 to 0.078) compared to the image model. This suggests that the model learns to leverage view changes in video data, effectively capturing view-dependent effects to predict specular properties more accurately.

One-step deterministic fine-tuning. We also ablate the design choice of 1-step deterministic fine-tuning in Table 3 and 4. By default, the inverse renderer performs 20 denoising steps during inference. Building on recent findings in image diffusion models [47], we demonstrate that strongly conditioned *video* diffusion models can also be fine-tuned as 1-step deterministic models. Despite the significantly reduced computational cost, we observe that 1-step models consistently produce more “accurate” predictions and outperform multi-step stochastic models in photometric evaluations such as PSNR scores. However, the 1-step model enforces deterministic output, which can result in blurrier predictions for ambiguous regions with high-frequency details, thus yielding lower perceptual metrics, such as LPIPS. For neural forward rendering and relighting tasks, we use multi-step stochastic models to capture more realistic details, though we note that 1-step models can be a competitive choice for error-sensitive tasks and enhancing runtime efficiency.

5.4. Evaluation of Relighting

In Table 2 and Fig. 6, we compare with recent state-of-the-art relighting methods DiLightNet [77] and Neural Gaffer [26]. Our method outperforms these baselines, particularly in scenes with complex shadows and inter-reflections. Overall, it produces high-quality lighting effects and more accurate color and scale.

Ablation study. We ablate the design choices of synthetic-



Figure 8. **Image editing applications.** Top: Realistic material editing, adjusting the sphere’s roughness and the horse’s metallic. Bottom: Object insertion of a bathtub and table into scene images.

real joint training in Fig. 7. While synthetic data provides accurate supervision signals, it is limited to a specific domain and lacks the diversity and complexity found in real-world data. When training exclusively on synthetic data (Ours Synth.), the model struggles with complex structures, such as trees, which are rarely represented in synthetic datasets. Since the real-world auto-labels are estimated using inverse rendering models and contain imperfections, we find that incorporating a LoRA [23] during training with real data consistently improves visual quality.

5.5. Applications

We show material editing and object insertion applications in Fig. 8. In the top row, we adjust the sphere’s roughness from 0.15 to 0.6 and increase the horse’s metallic property from 0 to 1, achieving photorealistic material edits. In the bottom row, we insert a bathtub and table in the G-buffer space of the input image, and use our forward renderer to produce the edited result. The inserted objects blend naturally into the scene, generating realistic reflections and shadows.

6. Discussion

DIFFUSIONRENDERER provides a scalable, data-driven approach to inverse and forward rendering, achieving high-quality G-buffer estimation and photorealistic image generation without relying on explicit path tracing or precise 3D scene representations. Jointly trained on synthetic and auto-labeled real-world data, DIFFUSIONRENDERER consistently outperforms state-of-the-art methods.

Limitations and future work. Our method is based on Stable Video Diffusion, which operates offline and would benefit from distillation techniques to improve inference speed. For editing tasks, the inverse and forward rendering models preserve most of the original content but may introduce slight variations in color or texture. Future work could explore task-specific fine-tuning [33] and develop neural intrinsic features to enhance content consistency and handle more complex visual effects. Additionally, our real-world auto-labeling currently adopts off-the-shelf lighting estimation model [57] which could benefit from better accuracy and robustness. With rapid advancements in video diffusion models [1] toward higher quality and faster inference speeds, we are optimistic that DIFFUSIONRENDERER will inspire future research in high-quality image synthesis and editing.

Acknowledgments. The authors thank Shiqiu Liu, Yichen Sheng, and Michael Kass for their insightful discussions that contributed to this project. We also appreciate the discussions with Xuanchi Ren, Tianchang Shen and Zheng Zeng during the model development process.

References

- [1] Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025. 3, 8
- [2] Jonathan T. Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. *IEEE transactions on pattern analysis and machine intelligence*, 37(8):1670–1687, 2014. 1, 2
- [3] Harry Barrow and J. M. Tenenbaum. Recovering intrinsic scene characteristics from images, 1978. 1
- [4] Harry Barrow, J. M. Tenenbaum, A. Hanson, and E. Riseman. Recovering intrinsic scene characteristics. *Comput. Vis. Syst.*, 2:3–26, 1978. 2
- [5] Sean Bell, Kavita Bala, and Noah Snavely. Intrinsic images in the wild. *ACM Transactions on Graphics (TOG)*, 33(4): 159, 2014. 2, 6, 7
- [6] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. 2, 3, 4, 1
- [7] Mark Boss, Varun Jampani, Kihwan Kim, Hendrik P.A. Lensch, and Jan Kautz. Two-shot spatially-varying BRDF and shape estimation. In *CVPR*, 2020. 2
- [8] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P.A. Lensch. NeRD: neural reflectance decomposition from image collections. In *ICCV*, 2021. 3
- [9] Adrien Bousseau, Sylvain Paris, and Frédo Durand. User-assisted intrinsic images. In *ACM Transactions on Graphics (TOG)*, page 130. ACM, 2009. 2
- [10] Brent Burley. Physically-based shading at Disney. In *ACM SIGGRAPH*, pages 1–7, 2012. 3
- [11] Wenzheng Chen, Joey Litalien, Jun Gao, Zian Wang, Clement Fuji Tsang, Sameh Khalis, Or Litany, and Sanja Fidler. DIB-R++: Learning to predict lighting and material with a hybrid differentiable renderer. In *NeurIPS*, 2021. 3
- [12] Robert L. Cook and Kenneth E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics (ToG)*, 1(1):7–24, 1982. 3
- [13] Michael Deering, Stephanie Winner, Bic Schediwy, Chris Duffy, and Neil Hunt. The triangle processor and normal vector shader: a VLSI system for high performance graphics. *ACM Trans. on Graphics*, 22(4):21–30, 1988. 2, 3
- [14] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3D objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. 6, 1
- [15] Kangle Deng, Timothy Omerick, Alexander Weiss, Deva Ramanan, Jun-Yan Zhu, Tinghui Zhou, and Maneesh Agrawala. FlashTex: fast relightable mesh texturing with LightControlNet. In *European Conference on Computer Vision (ECCV)*, 2024. 4, 7
- [16] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, 2021. 3
- [17] Xiao Fu, Wei Yin, Mu Hu, Kaixuan Wang, Yuexin Ma, Ping Tan, Shaojie Shen, Dahua Lin, and Xiaoxiao Long. GeoWizard: unleashing the diffusion priors for 3D geometry estimation from a single image. In *ECCV*, 2024. 4, 5, 6
- [18] David Griffiths, Tobias Ritschel, and Julien Philip. Outcast: Outdoor single-image relighting with cast shadows. In *Computer Graphics Forum*, pages 179–193. Wiley Online Library, 2022. 3
- [19] Roger Grosse, Micah K. Johnson, Edward H. Adelson, and William T. Freeman. Ground truth dataset and baseline evaluations for intrinsic image algorithms. In *ICCV*, pages 2335–2342. IEEE, 2009. 2, 6
- [20] Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. Shape, light, and material decomposition from images using Monte Carlo rendering and denoising. *arXiv:2206.03380*, 2022. 3
- [21] Pedro Hermosilla, Sebastian Maisch, Tobias Ritschel, and Timo Ropinski. Deep-learning the latent space of light transport. In *Computer Graphics Forum*, pages 207–217. Wiley Online Library, 2019. 2
- [22] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 3
- [23] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. 5, 8
- [24] Giulio Jiang and Bernhard Kainz. Deep radiance caching: Convolutional autoencoders deeper in ray tracing. *Computers & Graphics*, 94:22–31, 2021. 2
- [25] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5322–5332, 2024. 3
- [26] Haian Jin, Yuan Li, Fujun Luan, Yuanbo Xiangli, Sai Bi, Kai Zhang, Zexiang Xu, Jin Sun, and Noah Snavely. Neural gaffer: Relighting any object via diffusion. In *Advances in Neural Information Processing Systems*, 2024. 3, 4, 6, 7, 8, 2
- [27] Simon Kallweit, Thomas Müller, Brian McWilliams, Markus Gross, and Jan Novák. Deep scattering: Rendering atmospheric clouds with radiance-predicting neural networks. *ACM Transactions on Graphics (TOG)*, 36(6):1–11, 2017. 2

- [28] Brian Karis. Real shading in Unreal Engine 4. ACM SIGGRAPH Course on Physically Based Shading Theory and Practice, 4(3):1, 2013. 6
- [29] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In Proc. NeurIPS, 2022. 3, 5
- [30] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024. 3, 5, 6
- [31] 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. 2
- [32] Peter Kocsis, Vincent Sitzmann, and Matthias Nießner. Intrinsic image diffusion for single-view material estimation. In arxiv, 2023. 3, 6, 7
- [33] Peter Kocsis, Julien Philip, Kalyan Sunkavalli, Matthias Nießner, and Yannick Hold-Geoffroy. LightIt: illumination modeling and control for diffusion models. In CVPR, 2024. 3, 8
- [34] Balazs Kovacs, Sean Bell, Noah Snavely, and Kavita Bala. Shading annotations in the wild. In CVPR, pages 6998–7007, 2017. 2
- [35] Edwin H Land and John J McCann. Lightness and retinex theory. Josa, 61(1):1–11, 1971. 2
- [36] Zhengqi Li and Noah Snavely. Cgintrinsics: Better intrinsic image decomposition through physically-based rendering. In ECCV, pages 371–387, 2018. 2
- [37] Zhengqin Li, Mohammad Shafiei, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image. In CVPR, pages 2475–2484, 2020. 2, 6, 7
- [38] Zhengqin Li, Ting-Wei Yu, Shen Sang, Sarah Wang, Sai Bi, Zexiang Xu, Hong-Xing Yu, Kalyan Sunkavalli, Miloš Hašan, Ravi Ramamoorthi, et al. OpenRooms: an end-to-end open framework for photorealistic indoor scene datasets. arXiv preprint arXiv:2007.12868, 2020. 2
- [39] Ruofan Liang, Huiting Chen, Chunlin Li, Fan Chen, Selvakumar Panneer, and Nandita Vijaykumar. Envidr: Implicit differentiable renderer with neural environment lighting. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 79–89, 2023. 3
- [40] Ruofan Liang, Zan Gojcic, Merlin Nimier-David, David Acuna, Nandita Vijaykumar, Sanja Fidler, and Zian Wang. Photorealistic object insertion with diffusion-guided inverse rendering. In ECCV, 2024. 3, 2
- [41] Zhihao Liang, Qi Zhang, Ying Feng, Ying Shan, and Kui Jia. Gs-ir: 3d gaussian splatting for inverse rendering. arXiv preprint arXiv:2311.16473, 2023. 3
- [42] Zhi-Hao Lin, Bohan Liu, Yi-Ting Chen, David Forsyth, Jia-Bin Huang, Anand Bhattad, and Shenlong Wang. UrbanIR: large-scale urban scene inverse rendering from a single video. arXiv preprint arXiv:2306.09349, 2023. 6, 3, 5
- [43] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. DL3DV-10K: a large-scale scene dataset for deep learning-based 3D vision. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 22160–22169, 2024. 5, 6, 7
- [44] Andrew Liu, Shiry Ginosar, Tinghui Zhou, Alexei A. Efros, and Noah Snavely. Learning to factorize and relight a city. In ECCV, 2020. 3
- [45] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. Wonder3D: single image to 3D using cross-domain diffusion. arXiv preprint arXiv:2310.15008, 2023. 4
- [46] Rafal K. Mantiuk, Param Hanji, Maliha Ashraf, Yuta Asano, and Alexandre Chapiro. Colorvideovdp: A visual difference predictor for image, video and display distortions. ACM Trans. Graph., 43(4), 2024. 3
- [47] Gonzalo Martin Garcia, Karim Abou Zeid, Christian Schmidt, Daan de Geus, Alexander Hermans, and Bastian Leibe. Fine-tuning image-conditional diffusion models is easier than you think. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2025. 8
- [48] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: representing scenes as neural radiance fields for view synthesis. arXiv preprint arXiv:2003.08934, 2020. 2
- [49] Thomas Müller, Fabrice Rousselle, Jan Novák, and Alexander Keller. Real-time neural radiance caching for path tracing. arXiv preprint arXiv:2106.12372, 2021. 2
- [50] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting triangular 3D models, materials, and lighting from images. arXiv:2111.12503, 2021. 3
- [51] Lukas Murmann, Michael Gharbi, Miika Aittala, and Fredo Durand. A multi-illumination dataset of indoor object appearance. In 2019 IEEE International Conference on Computer Vision (ICCV), 2019. 3
- [52] Oliver Nalbach, Elena Arabadzhiyska, Dushyant Mehta, H.-P. Seidel, and Tobias Ritschel. Deep shading: convolutional neural networks for screen space shading. In Computer graphics forum, pages 65–78. Wiley Online Library, 2017. 2, 3
- [53] Rohit Pandey, Sergio Orts-Escolano, Chloe LeGendre, Christian Haene, Sofien Bouaziz, Christoph Rhemann, Paul Debevec, and Sean Fanello. Total relighting: Learning to relight portraits for background replacement. In ACM Transactions on Graphics (Proceedings SIGGRAPH), 2021. 3
- [54] Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. OptiX: a general purpose ray tracing engine. ACM Trans. Graph., 29(4), 2010. 5, 1
- [55] Matt Pharr, Wenzel Jacob, and Greg Humphreys. Physically Based Rendering - From Theory to Implementation. Morgan Kaufmann, fourth edition, 2023. 1, 3
- [56] Julien Philip, Michaël Gharbi, Tinghui Zhou, Alexei A Efros, and George Drettakis. Multi-view relighting using a geometry-aware network. ACM Trans. Graph., 38(4):78–1, 2019. 3

- [57] Pakkapon Phongthawee, Worameth Chinchuthakun, Nontaphat Sinsunthithet, Amit Raj, Varun Jampani, Pramook Khungurn, and Supasorn Suwajanakorn. DiffusionLight: light probes for free by painting a chrome ball. In *ArXiv*, 2023. 3, 5, 8, 2
- [58] Yohan Poirier-Ginter, Alban Gauthier, Julien Philip, Jean-François Lalonde, and George Drettakis. A Diffusion Approach to Radiance Field Relighting using Multi-Illumination Synthesis. *Computer Graphics Forum*, 2024. 3
- [59] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 3
- [60] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *International Conference on Computer Vision (ICCV) 2021*, 2021. 5
- [61] 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 (CVPR)*, 2022. 2, 3, 5
- [62] Viktor Rudnev, Mohamed Elgharib, William Smith, Lingjie Liu, Vladislav Golyanik, and Christian Theobalt. NeRF for outdoor scene relighting. In *ECCV*, 2022. 3
- [63] Soumyadip Sengupta, Jinwei Gu, Kihwan Kim, Guilin Liu, David W. Jacobs, and Jan Kautz. Neural inverse rendering of an indoor scene from a single image. In *ICCV*, 2019. 2
- [64] Yahao Shi, Yanmin Wu, Chenming Wu, Xing Liu, Chen Zhao, Haocheng Feng, Jingtuo Liu, Liangjun Zhang, Jian Zhang, Bin Zhou, et al. Gir: 3d gaussian inverse rendering for relightable scene factorization. *arXiv preprint arXiv:2312.05133*, 2023. 3
- [65] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2015. 3
- [66] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. State of the art on neural rendering. In *Computer Graphics Forum*, pages 701–727. Wiley Online Library, 2020. 2
- [67] Eric Veach. *Robust Monte Carlo methods for light transport simulation*. Stanford University, 1998. 3
- [68] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, page 195–206, 2007. 3
- [69] Zian Wang, Jonah Philion, Sanja Fidler, and Jan Kautz. Learning indoor inverse rendering with 3D spatially-varying lighting. In *ICCV*, 2021. 2
- [70] Zian Wang, Wenzheng Chen, David Acuna, Jan Kautz, and Sanja Fidler. Neural light field estimation for street scenes with differentiable virtual object insertion. In *ECCV*, 2022. 2
- [71] Zian Wang, Tianchang Shen, Jun Gao, Shengyu Huang, Jacob Munkberg, Jon Hasselgren, Zan Gojic, Wenzheng Chen, and Sanja Fidler. Neural fields meet explicit geometric representations for inverse rendering of urban scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 3, 6, 5
- [72] Felix Wimbauer, Shangzhe Wu, and Christian Rupprecht. De-rendering 3D objects in the wild. In *CVPR*, 2022. 2
- [73] Tong Wu, Guandao Yang, Zhibing Li, Kai Zhang, Ziwei Liu, Leonidas Guibas, Dahua Lin, and Gordon Wetzstein. Gpt-4v(ision) is a human-aligned evaluator for text-to-3d generation. In *CVPR*, 2024. 3
- [74] Chen Xi, Peng Sida, Yang Dongchen, Liu Yuan, Pan Bowen, Lv Chengfei, and Zhou. Xiaowei. IntrinsicAnything: learning diffusion priors for inverse rendering under unknown illumination. *arxiv: 2404.11593*, 2024. 3
- [75] Ye Yu and William A. P. Smith. InverseRenderNet: learning single image inverse rendering. In *CVPR*, 2019. 2
- [76] Greg Zaal and et al. *Poly Haven - The Public 3D Asset Library*, 2024. 6
- [77] Chong Zeng, Yue Dong, Pieter Peers, Youkang Kong, Hongzhi Wu, and Xin Tong. DiLightNet: fine-grained lighting control for diffusion-based image generation. In *ACM SIGGRAPH 2024 Conference Papers*, 2024. 3, 6, 8, 2
- [78] Zheng Zeng, Valentin Deschaintre, Iliyan Georgiev, Yannick Hold-Geoffroy, Yiwei Hu, Fujun Luan, Ling-Qi Yan, and Miloš Hašan. RGB \leftrightarrow X: image decomposition and synthesis using material-and lighting-aware diffusion models. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 2, 3, 4, 6, 7
- [79] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. PhysSG: Inverse rendering with spherical Gaussians for physics-based material editing and relighting. In *CVPR*, 2021. 3
- [80] Kai Zhang, Fujun Luan, Zhengqi Li, and Noah Snavely. IRON: inverse rendering by optimizing neural SDFs and materials from photometric images. In *CVPR*, 2022. 3
- [81] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 6
- [82] Xiuming Zhang, Pratul P. Srinivasan, Boyang Deng, Paul Debevec, William T. Freeman, and Jonathan T. Barron. NeRFactor: neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (TOG)*, 40(6):1–18, 2021. 3
- [83] Qi Zhao, Ping Tan, Qiang Dai, Li Shen, Enhua Wu, and Stephen Lin. A closed-form solution to retinex with nonlocal texture constraints. *TPAMI*, 34(7):1437–1444, 2012. 2
- [84] Hao Zhou, Sunil Hadap, Kalyan Sunkavalli, and David W. Jacobs. Deep single-image portrait relighting. In *ICCV*, 2019. 3

- [85] Jingsen Zhu, Fujun Luan, Yuchi Huo, Zihao Lin, Zhihua Zhong, Dianbing Xi, Rui Wang, Hujun Bao, Jiayang Zheng, and Rui Tang. Learning-based inverse rendering of complex indoor scenes with differentiable Monte Carlo raytracing. In SIGGRAPH Asia 2022 Conference Papers. ACM, 2022. [5](#), [6](#), [7](#)

DIFFUSIONRENDERER: Neural Inverse and Forward Rendering with Video Diffusion Models

Supplementary Material

In the supplementary material, we provide additional implementation details (Sec. A) and further results and analysis (Sec. B). Please refer to the [ACCOMPANYING VIDEO](#) for more qualitative results and comparisons.

A. Experimental Settings

Implementation details. We fine-tune our models based on Stable Video Diffusion¹ [6].

For the *inverse renderer*, we modify the diffusion UNet by expanding four additional channels in the first convolutional layer to include image conditions. We optimize both the diffusion UNet parameters and the domain embedding parameters using a learning rate of 3×10^{-5} . The training is conducted with a batch size of 256, with a mix of multiple scene attributes. When generating the single-channel depth, metallic, and roughness maps, we average the outputs across the three channels to obtain the final result for each map.

In the *forward renderer*, we expand the first convolutional layer of the diffusion UNet by 20 additional channels to concatenate the additional pixel-aligned G-buffer conditions. Since the depth, metallic, and roughness maps are single-channel properties, we replicate each to create three-channel inputs before passing them into the VAE encoder \mathcal{E} . The weights of the cross-attention layers are repurposed for lighting conditions, and are reset prior to training. We use a learning rate of 1×10^{-4} for optimization.

Both models are trained using the AdamW optimizer for 20,000 iterations, with mixed-precision (fp16) training at a resolution of 512×512 pixels. The training takes around 2 days on 32 A100 GPUs. We have empirically observed that the video model performs best when trained on video lengths that it will encounter during inference. To ensure robust generalization across different frame lengths, we randomly select training video lengths of 1, 4, 8, 16, and 24 frames. This strategy allows the model to adapt effectively to varying video lengths during inference without compromising output quality. As a result, the models can also effectively process a single image by treating it as a video with one frame. During the training of both models, a 0.1 dropout is applied independently to each condition channel to reduce reliance on individual conditions and potentially enhance robustness. During inference, we empirically observe that a small classifier-free guidance (CFG) such as 1.2 enhances the visual quality of the forward rendering model. CFG

does not provide noticeable benefit for the inverse rendering model and we do not use it for the inverse rendering model.

Data preparation. For synthetic data curation, we begin with the Objaverse [14] LVIS split, containing 46,207 3D models. The 3D assets are filtered based on the following criteria: (i) assets include valid PBR attributes such as roughness and metallic, (ii) assets can be rendered without geometry/texture artifacts. This process yields a final set of 36,500 3D assets. We collect 766 HDR panoramas from three sources: PolyHaven², DoschDesign³, and HDRMaps⁴. For PBR textures, we collect 6,300 CC0 textures from multiple sources: 3D Textures⁵, ambientCG⁶, cgbookcase⁷, PolyHaven⁸, sharettextures⁹, and TextureCan¹⁰. We remove textures that include only diffuse channels or lack diffuse textures, and manually exclude non-tileable textures, resulting in 4,260 high-quality PBR textures.

In each scene, we place a plane with a randomly selected PBR material, and sample up to three 3D objects, and place them on the plane after randomly rotating, translating, and scaling. We perform collision detection to avoid intersecting objects. We also place up to three primitives (cube, sphere, and cylinder) with randomized materials to cover complex lighting effects such as inter-reflections. The materials of primitives can be from the aforementioned texture maps or a monolithic material with varying albedo, roughness, and metallic. A randomly selected HDR environment map illuminates the scene. We also add random horizontal rotation, flipping, and intensity scaling to the environment map. The rendered videos contain 5 types of motions, 1) 360-degree camera orbits; 2) small-scale regional camera oscillation; 3) 360-degree rotating light with a fixed camera; 4) rotating objects with a fixed camera; and 5) translating objects around the plane.

We render videos of all scenes with corresponding intrinsic images in a custom path tracer based on OptiX [54], with 256 spp, OptiX denoising and AgX tonemapper¹¹. In total, there are 150,000 videos with paired ground-truth G-buffers and environment maps, at 24 frames per video in 512x512

²polyhaven.com/hdris (License: CC0)

³doschdesign.com(License: [link](#))

⁴hdrmaps.com (License: Royalty-Free)

⁵<https://3dtextures.me/tag/cc0/>

⁶<https://ambientcg.com>

⁷<https://www.cgbookcase.com/textures>

⁸<https://polyhaven.com/>

⁹<https://www.sharettextures.com>

¹⁰<https://www.texturecan.com>

¹¹<https://github.com/sobotka/AgX>

¹<https://huggingface.co/stabilityai/stable-video-diffusion-img2vid>

CVVDP \uparrow	<i>SyntheticObjects</i>	<i>SyntheticScenes</i>
DiLightNet [77]	5.44	2.99
Neural Gaffer [26]	6.49	3.47
Ours	6.77	6.40

Table S1. Quantitative evaluation of relighting in terms of ColorVideoVDP. ColorVideoVDP reports video quality in the JOD (Just-Objectionable-Difference) units. The highest quality (no difference) is reported as 10 and lower values are reported for distorted content. We compute a JOD value per clip for three novel lighting conditions in each series and report the average over all clips.

resolution.

Baseline configurations. DiLightNet [77] requires a text prompt per example, so we used `meta/llama-3.2-11b-vision-instruct`¹² to generate a short prompt for each example in *SyntheticObjects* and *SyntheticScenes* based on the first image in each clip and the instruction "What is in this image? Describe the materials. Be concise and produce an answer with a few sentences, no more than 50 words."

Environment map encoder pre-training. As detailed in the main paper, the environment lighting condition in our forward rendering model is encoded through cross-attention between the UNet’s spatial latent features and the environment map representation. To provide effective lighting encodings, similar to VAE and CLIP embeddings in diffusion models, we propose pre-training an environment map auto-encoder specifically designed to capture HDR light intensity and orientation.

With both LDR space and log space environment maps (\mathbf{E}_{ldr} and \mathbf{E}_{log}) as the model input and auto-encoder’s reconstruction target, our encoder can retain detailed ambient lighting information while emphasizing high-intensity HDR light spots. To ensure precise control over light orientation in scene rendering, we introduce a directional encoding map, \mathbf{E}_{dir} , where each pixel represents a unit vector corresponding to a light direction in the camera coordinate system. By modifying \mathbf{E}_{dir} , the light orientation in the scene can be adjusted accordingly.

The pre-training process aims to produce an environment map encoder \mathcal{E}_{env} , capable of encoding complex directional HDR lighting. For this, we pair \mathcal{E}_{env} with two auxiliary modules: an environment map decoder \mathcal{D}_{env} and a direction query encoder \mathcal{E}_{dir} . This forms an auto-encoder training pipeline, as illustrated in Fig. S1. The encoder \mathcal{E}_{env} processes concatenated VAE-encoded inputs $\mathbf{h}_{\mathbf{E}} = (\mathcal{E}(\mathbf{E}_{\text{ldr}}), \mathcal{E}(\mathbf{E}_{\text{log}}), \mathcal{E}(\mathbf{E}_{\text{dir}}))$, generating $K = 4$ levels of multi-resolution features $(\mathbf{h}_{\text{env}}^i)_{i=1}^K$. Similarly, \mathcal{E}_{dir} takes a VAE-encoded directional map $\mathbf{h}_{\mathbf{D}} = \mathcal{E}(\mathbf{E}'_{\text{dir}})$, producing features $(\mathbf{h}_{\text{dir}}^i)_{i=1}^K$ of the same shape. The decoder \mathcal{D}_{env} reconstructs the inputs \mathbf{E}'_{ldr} and \mathbf{E}'_{log} using the features $(\mathbf{h}_{\text{env}}^i)_{i=1}^K$ and $(\mathbf{h}_{\text{dir}}^i)_{i=1}^K$ through cross-attention layers. To

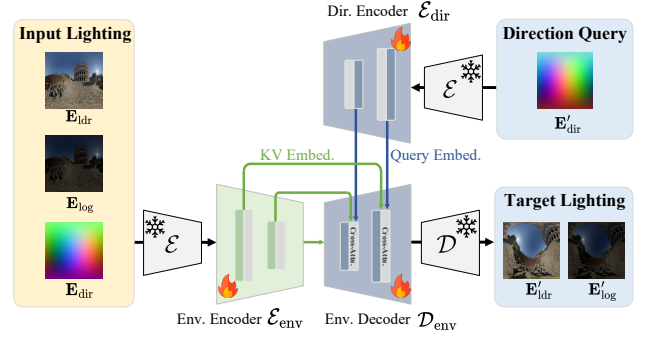


Figure S1. The overview of our environment map auto-encoder training pipeline.

enhance directional encoding, the training objective involves re-projecting the environment map with random rotations applied to the lighting sphere. This rotation information can be precisely represented by \mathbf{E}'_{dir} . To reconstruct the re-projected environment map, we use the features $(\mathbf{h}_{\text{dir}}^i)_{i=1}^K$ encoded from \mathbf{E}'_{dir} as embedding to query the directional HDR lighting encoded in $(\mathbf{h}_{\text{env}}^i)_{i=1}^K$ (serving as key-value embedding) through the cross-attention layers in environment map decoder \mathcal{D}_{env} . The training objective therefore is:

$$\mathcal{L}_{\text{env}} = \|\mathbf{h}_{\mathbf{E}'} - \mathcal{D}_{\text{env}}(\mathcal{E}_{\text{env}}(\mathbf{h}_{\mathbf{E}}), \mathcal{E}_{\text{dir}}(\mathbf{h}_{\mathbf{D}}))\|^2 \quad (7)$$

where $\mathbf{h}_{\mathbf{E}'} = (\mathcal{E}(\mathbf{E}'_{\text{ldr}}), \mathcal{E}(\mathbf{E}'_{\text{log}})) \in \mathbb{R}^{\times h_{\text{env}} \times w_{\text{env}} \times 8}$.

Object Insertion. We provide additional details of object insertion application shown in main paper Fig. 8. The objective is to seamlessly insert an object (either 2D or 3D) into a given background image \mathbf{I}_{bg} , ensuring consistent appearance with the background (e.g., aligned lighting effects). Our method achieve this task with a combination of the inverse and forward rendering processes, as illustrated in Fig. S2.

First, our inverse rendering model estimates the G-buffer of the background image \mathbf{I}_{bg} . The G-buffer of the object to be inserted is obtained either through our inverse renderer or directly from a rendering engine. Based on the known foreground object mask \mathbf{M} , these G-buffers are then blended to create a composite G-buffer. Additionally, we estimate the lighting using an off-the-shelf model [57].

Using the composite G-buffer and estimated lighting, our forward rendering model generates two images: $\mathbf{I}_{\text{ins}}^*$ representing the scene with the inserted object, and \mathbf{I}_{bg}^* , the re-rendering of the original background. To minimize unintended changes to the original background image, we follow [37, 40, 70] and compute a shading ratio $\rho = \mathbf{I}_{\text{ins}}^* / \mathbf{I}_{\text{bg}}^*$ that accounts for the relative shading effects introduced by the inserted object.

The final edited image \mathbf{I}_{ins} is computed by multiplying the shading ratio with the original background image \mathbf{I}_{bg} and compositing the masked foreground object $\mathbf{M} \cdot \mathbf{I}_{\text{ins}}^*$ onto the

¹²<https://www.llama.com/>

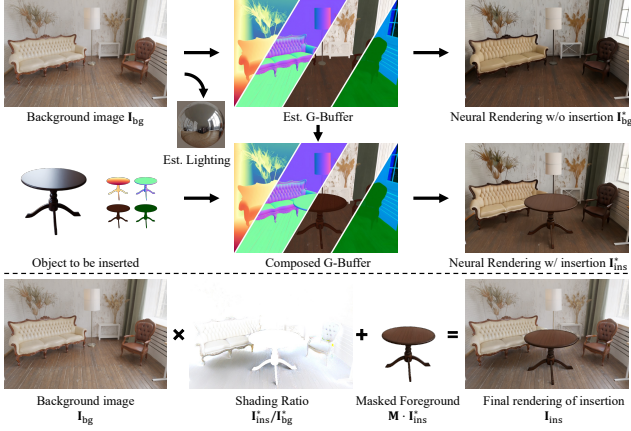


Figure S2. Overview of the object insertion workflow.

shaded background:

$$\mathbf{I}_{\text{ins}} = (1 - M) \cdot \mathbf{I}_{\text{bg}} \cdot \frac{\mathbf{I}_{\text{ins}}^*}{\mathbf{I}_{\text{bg}}^*} + M \cdot \mathbf{I}_{\text{ins}}^* \quad (8)$$

This process is visualized in Fig. S2 (bottom).

B. Additional Results

Runtime cost. Since our models are built on top of Stable Video Diffusion, the inference runtime cost of our models is roughly on the same level as Stable Video Diffusion. For a 24-frame video with a resolution of 512x512, the peak GPU memory cost for both models at inference time is around 21 GB. the inverse rendering model takes 9.7 seconds to perform 20 denoising steps including VAE encoding and decoding, clocked on one A100 GPU. The forward rendering model takes 20.3 seconds to run 20 denoising steps including VAE encoding and decoding. The increased runtime of the forward renderer is due to additional condition signals, which require extra time for encoding.

Without a separate environment map encoder, Ours (w/o Env. Encoder) completes 20 denoising steps in 19.9 seconds. The runtime overhead introduced by the environment map encoder is negligible.

Temporal consistency. In Table S1 we report ColorVideoVDP [46] (CVVDP) scores for the relighting comparison (c.f., Table 2 and Fig. 6 in the main paper). CVVDP predicts the perceptual difference between pairs of videos and accounts for spatial and temporal aspects of vision. We note that our method has the highest CVVDP score for both test sets, which is consistent with visual inspections. Please refer to the supplemental video to assess temporal consistency. In contrast to Neural Gaffer and DiLightNet, which leverage image diffusion models, our approach builds upon video diffusion models, which provide considerably improved temporal consistency. For reproducibility, CVVDP was configured according to:

ColorVideoVDP v0.4.2, 75.4 [pix/deg], Lpeak=200, Lblack=0.2, Lrefl=0.3979 [cd/m²] (standard_4k).

User study. We conducted a user study to evaluate the image perceptual quality of our method. In this study, participants were shown a reference path-traced rendering alongside a pair of renderings: one from our method and one from a baseline (randomly shuffled). They were asked to select which rendering perceptually more closely resembles the reference, considering aspects like lighting, shadows, and reflections. This user study was conducted for both neural rendering and relighting tasks. The evaluation data were sampled from SyntheticScenes and SyntheticObjects (the same datasets used for Table 1 and Table 2) (70 scenes). For each comparison, we collected 9 user selections to determine the preferred rendering by majority voting. The preference percentages for our method compared to baseline approaches are reported across all examples. Inspired by GPTEval3D [73], we repeat this experiment using GPT-4V as perceptual evaluators. Reported in Table S2, the user study results align with our findings in the main paper, and indicate a reasonable level of agreement between human and GPT-4V assessments.

		Neural Rendering				Relighting	
		SSRT	SplitSum	RGB↔X	DiLightNet	DiLightNet	N.Gaffer
Scenes	Human	72%	75%	85%	85%	90%	65%
	GPT4V	40%	50%	80%	85%	60%	68%
Objs	Human	37%	43%	76%	83%	57%	57%
	GPT4V	57%	45%	87%	54%	55%	52%

Table S2. **User study.** We report the percentage of images where users preferred Ours over baselines. A preference > 50% indicates Ours outperforming baselines. Evaluation follows main paper Table 1, 2 on *SyntheticScenes* and *SyntheticObjects*.

Comparison with FEGR [71] and UrbanIR [42]. We additionally compare to 3D inverse rendering and relighting approaches FEGR [71] and UrbanIR [42] in Fig. S5. These methods optimize neural 3D representation, then use volume rendering and PBR to produce the final relighting result. As the input data is limited to a single illumination condition, they often cannot cleanly remove shadows from the albedo, resulting in shadow artifacts in re-lit results. Additionally, existing scene reconstruction methods struggle to handle highly detailed structures such as trees, and dynamic scenes, which limits their fidelity for PBR path tracing. In contrast, our method consistently generates more photorealistic results without relying on explicit 3D geometry constraints. We refer to the accompanying video for animated results.



Figure S3. Visualization of the synthetic datasets for quantitative evaluation.



Figure S4. Additional qualitative comparison of relighting. Our method produces more accurate specular reflections compared to the baselines.

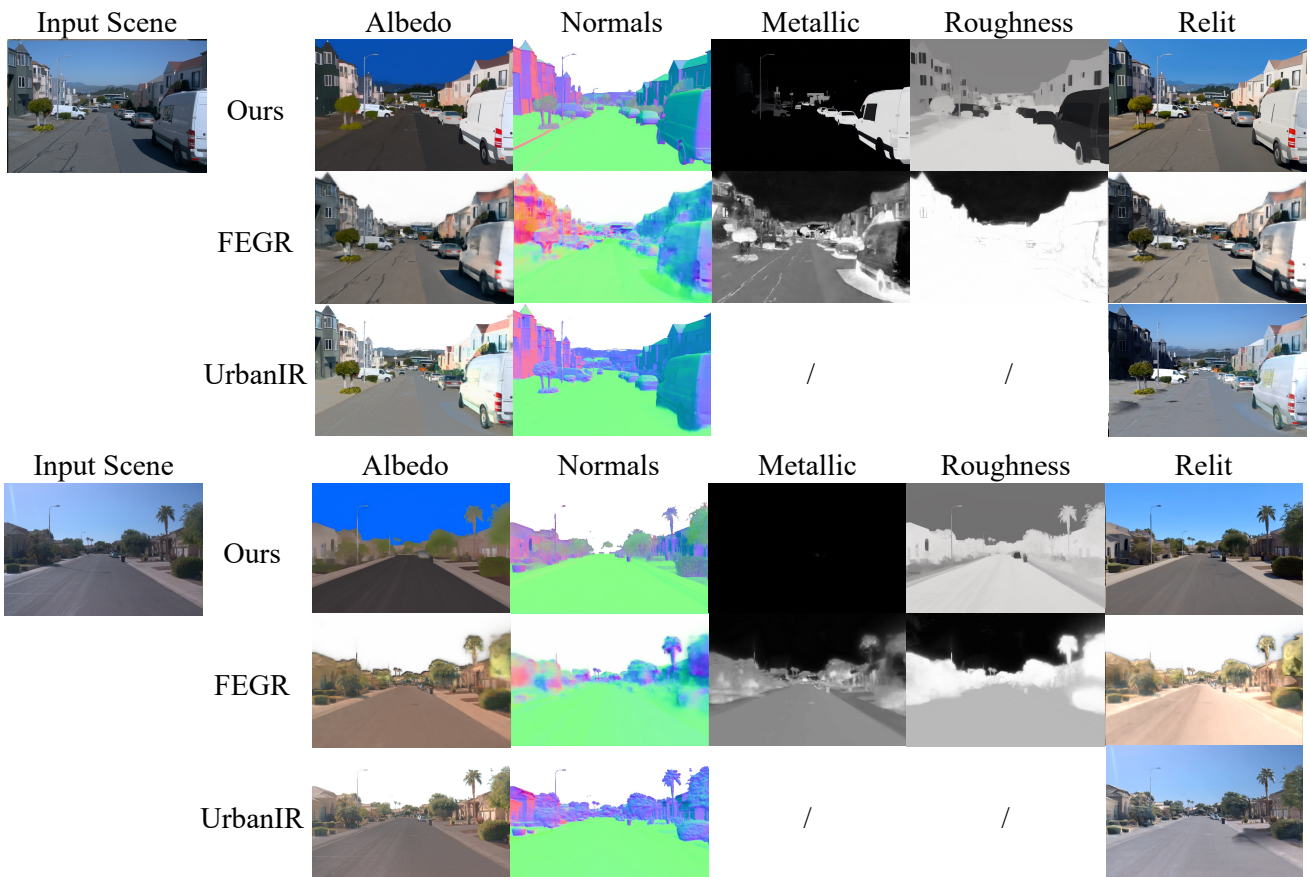


Figure S5. Qualitative comparison of inverse rendering and relighting on Waymo dataset with FEGR [71] and UrbanIR [42].