


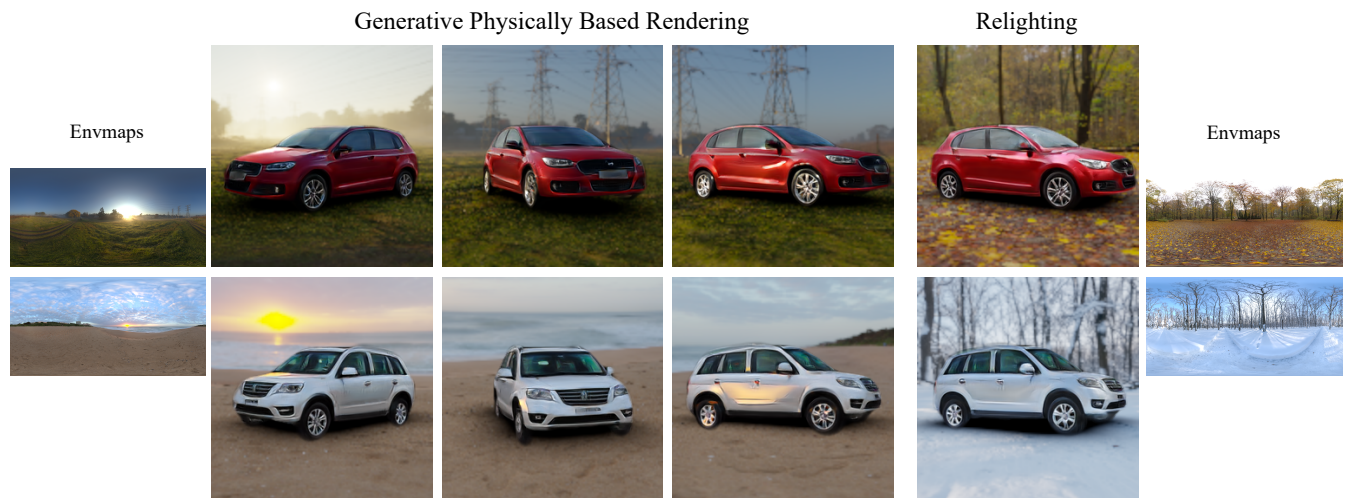
# Physically-Based Lighting for 3D Generative Models of Cars

N. Violante<sup>1</sup>  A. Gauthier<sup>1</sup>  S. Diolatzis<sup>2</sup>  T. Leimkühler<sup>3</sup>  G. Drettakis<sup>1</sup> 

<sup>1</sup>Inria & Université Côte d'Azur, France

<sup>2</sup>Intel, France

<sup>3</sup>MPI Informatik, Germany



**Figure 1:** Our method enables interactive physically-based rendering (PBR) with 3D GANs under arbitrary illumination conditions in the form of environment maps and unconstrained camera navigation, including 360° rotations. We achieve this with three main contributions: a method to estimate poses of a dataset of car images, a generative pipeline for PBR, and an improved generative network architecture and training solution. We only require a dataset of images of the desired object class (cars in our case) and a dataset of environment maps for training. Our method learns a disentangled representation of shading, enabling relighting with high-frequency reflections on shiny car bodies.

## Abstract

Recent work has demonstrated that Generative Adversarial Networks (GANs) can be trained to generate 3D content from 2D image collections, by synthesizing features for neural radiance field rendering. However, most such solutions generate radiance, with lighting entangled with materials. This results in unrealistic appearance, since lighting cannot be changed and view-dependent effects such as reflections do not move correctly with the viewpoint. In addition, many methods have difficulty for full, 360° rotations, since they are often designed for mainly front-facing scenes such as faces. We introduce a new 3D GAN framework that addresses these shortcomings, allowing multi-view coherent 360° viewing and at the same time relighting for objects with shiny reflections, which we exemplify using a car dataset. The success of our solution stems from three main contributions. First, we estimate initial camera poses for a dataset of car images, and then learn to refine the distribution of camera parameters while training the GAN. Second, we propose an efficient Image-Based Lighting model, that we use in a 3D GAN to generate disentangled reflectance, as opposed to the radiance synthesized in most previous work. The material is used for physically-based rendering with a dataset of environment maps. Third, we improve the 3D GAN architecture compared to previous work and design a careful training strategy that allows effective disentanglement. Our model is the first that generate a variety of 3D cars that are multi-view consistent and that can be relit interactively with any environment map.

## 1. Introduction

Recent 3D Generative Adversarial Networks (GANs) [CLC\*22, GLWT21, XLSL22] have achieved im-

pressive results in synthesizing 3D objects using neural volumetric representations. An appealing property is that such models can be trained using only unstructured 2D image collections [HMR19, CMK\*21, SLNG20]. Most existing solutions however are restricted to simplistic shading, such as the NeRF-style [MST\*20] emission-absorption model that simply reproduces observed radiance. This can lead to implausible appearance (e.g., no or incoherent view-dependent effects) and provides little or no disentanglement of materials, lighting and geometry. Our method focuses on cars, since they exhibit intricate and high-frequency view-dependent appearance and are most often viewed from arbitrary directions, contrary to mainly front-facing datasets such as faces often used in previous work. In this case, achieving multi-view consistency is harder because of view-dependent effects. Such consistency also requires estimation of poses for the training image dataset, which is challenging in our use case. In this paper, we address these challenges by presenting a novel methodology to train a 3D GAN that allows physically-based shading and (re-)lighting, while improving multi-view coherence compared to previous work for 360° interactive rendering.

With 3D generative models maturing, there is an increasing interest in transitioning towards physically-based models. For many use cases, it is not enough that a rendering of a generated sample merely looks plausible, but instead, appearance should be driven by physical principles of light transport. However, existing 3D GANs use the NeRF model, that computes a simple view-dependent radiance field. Lighting is thus entangled with materials and view-dependent effects are often “faked” using semi-transparent volumetric density etc. These limitations have motivated recent work, that attempts to introduce physically-based generators [DWW23, PXL\*21, RYCT23], but they only consider low-frequency illumination and/or shading. Such methods are insufficiently expressive and/or general for the complex lighting and shading of shiny objects such as cars.

In our solution, we introduce a physically-based generator that produces a *reflectance* field [BXS\*20] with a microfacet BRDF illuminated by an environment map, disentangling lighting from materials. Such a model has the potential to lead to powerful tools for 3D content creation and controllable data generation under arbitrary illumination for downstream tasks. The first major challenge is training such a representation, due to material–illumination ambiguities: a car that has no reflections could be a shiny under low frequency lighting or diffuse under high frequency lighting. One key observation is that, in addition to an established dataset of car images [YLCL15] which we augment with foreground masks, a second dataset of high-dynamic-range (HDR) environment maps [HGAL19] can be used to help disentangle light transport when generating reflectance and performing lighting on the fly. This auxiliary dataset is independent of the main dataset and hence does not impose additional restrictions. Our design results in high-quality and view-coherent samples that can be relit and composited with arbitrary environment maps.

A second major challenge in learning 3D GAN models from 2D data is the obvious mismatch between dimensions. The key to successful training is that the discriminator only gets access to *projections* of the generated 3D scene [HMR19]. For optimal mod-

eling of 3D shapes, the virtual cameras used for the projection should match the distribution of real cameras used to create the training dataset [BPD18]. Failing to match camera distributions results in distorted shapes (e.g., flat noses in faces), which may not be important when the shape is viewed from a limited range of angles (e.g., mostly from the front). However, for full 360° viewing, shape distortions become visually obvious and negatively effect training, making a faithful match of camera distributions indispensable. Camera distributions have been modeled using simple analytic priors [CMK\*21, SLNG20], learned jointly with the generator [NG21a], or estimated from the training data [CLC\*22]; these are however insufficient to achieve our goal of multi-view consistency with realistic lighting.

To obtain the precise camera distributions required for high-quality 360° viewing, we propose a novel hybrid strategy that takes into account both extrinsic and intrinsic camera parameters: We first propose a pipeline to estimate rough camera poses from the training images, based on computer vision tools, providing a first approximation to the distribution. These estimates are inherently ambiguous, e.g., a shorter object–camera distance can be compensated by a larger field of view, etc. Therefore, in a second step, we refine the camera pose estimates using a dedicated neural module that is jointly trained with the rest of the generator. We show that our strategy boosts both sample quality and disentanglement.

Evaluating the quality of 3D generative models is very challenging; there is no reliable methodology to obtain a quantitative evaluation of how well a solution respects multi-view consistency and relighting [BF22] that is the main goal of our work. We thus introduce a new methodology for evaluation using ground-truth synthetic data and GAN inversion.

In summary, our main contributions are:

- A volumetric field renderer that produces a *reflectance* representation, suitable for an Image-Based Lighting model that, combined with a style-based generator enables, for the first time, synthesis of realistic relightable assets with consistent and high-quality specular view-dependent effects.
- A method to directly estimate the distribution of full camera models for a dataset of images of single-class objects (cars in our case), including jointly learning the distribution of camera intrinsics and extrinsics during training.
- An improved generator architecture and training solution that is central in allowing the network to disentangle lighting from reflectance and to allow relighting, together with a novel evaluation methodology for multi-view consistency and relighting.

Taken together, these elements constitute the first method that can generate full 360° images of cars with free camera movement, allowing multi-view consistent renderings with view-dependent effects such as high-frequency reflections, all at interactive rates (Fig. 1). Code and pretrained models available at <https://repo-sam.inria.fr/fungraph/lighting-3d-generative-cars>

## 2. Related Work

In this section, we give an overview of techniques that use relightable neural fields (Sec. 2.1), before discussing 3D-aware gen-

erative image models (Sec. 2.2). We then review how these methods handle camera distributions (Sec. 2.3), and discuss the state of the art in relightable generative models (Sec. 2.4).

### 2.1. Neural Fields for Relighting

Neural fields [XTS\*22] are coordinate-based neural networks used to represent visual-computing primitives, frequently involving Multi-Layer Perceptrons (MLPs). The appearance of 3D scenes can be represented using a Neural Radiance Field (NeRF) [MST\*20], which maps a position and a view direction to volumetric density and color. Images are obtained using volume rendering with an emission-absorption model. While powerful for novel-view synthesis, this approach makes it difficult to perform editing such as relighting or material editing. As a remedy, reparameterizations can help to better handle view-dependent effects [VHM\*21], or the radiance transfer function can be learned [LTL\*22]. However, the most popular approaches for enabling editing are based on a decomposition of the scene into reflectance and illumination in conjunction with a more sophisticated rendering model [BBJ\*21, BJB\*21, BEK\*22, ZLW\*21, WSLG23, MHS\*22], which can include visibility [ZSD\*21, SDZ\*21] and indirect lighting [JLX\*23]. The environment-map-based models inspired our generator design, which utilizes a reflectance field [BGP\*21, BXS\*20].

### 2.2. 3D Generative Models

Given a large collection of images, Generative Adversarial Networks (GANs) [GPAM\*14] learn to capture the underlying data distribution and allow the generation of new images by jointly training a generator and a discriminator. The initial solutions naturally involved 2D image pipelines, where style-based architectures have evolved to become dominant [KLA19, KLA\*20, KAH\*20, KAL\*21]. Their well-behaved latent spaces furthermore allow high-quality image editing [AQW20, HHL20, CBPS20, GPM\*22, PWS\*21, KPACO21], including dedicated approaches to change poses [TEB\*20, LD21]. However, 3D-coherent editing with 2D generators is challenging.

As a remedy, 3D generative models internally employ a 3D representation, which is turned into images using differentiable rendering. This gives considerable control over the generated image content without requiring explicit supervision with 3D data [HMR19]. Most commonly, NeRF-like representations are used for their appealing properties during training [SLNG20, CMK\*21, DYXT22], but other representations have also been considered [GSW\*22]. Rendering an image and, consequently, training the model is very costly when using a volumetric representation and does not scale well to higher image resolutions. Therefore, many attempts have been made towards scalable generator architectures and training procedures [STWW22, GLWT21, CLC\*22, ZXNT21]. Notably, EG3D [CLC\*22] uses a triplane representation that decouples feature generation from ray marching, paired with a lightweight MLP for efficient volume rendering. Images are rendered at low resolution and then upsampled by a super-resolution network. Our approach is inspired by EG3D, as it allows scalable and high-quality sample generation, but we significantly extend the

architecture with a physically-based lighting and shading model as well as careful modeling of the camera distribution.

Injecting inductive biases into a 3D generative model aids the disentanglement of semantic attributes and enables more fine-grained control over the generated images [ZZZ\*18, TBRP\*22]. An instance of this approach is compositionality in the form of the explicit separation of foreground and background in object-centric scenes [NG21b, XLSL22]. We develop this powerful idea further by separately generating our object of interest (the car) and using an HDR 360° environment map for high-quality physically-based lighting.

An orthogonal line of recent research focuses on 3D generation using diffusion models [PJB22, LGT\*23, WLW\*23]. While showing great promise, these approaches are currently in an early stage and, different from our GAN-based approach, do not allow interactive sampling of the latent space, limiting their flexibility.

### 2.3. Camera Distributions in Generative Models

3D generative adversarial models build on a lossy stochastic projection [BPD18] to learn a distribution of 3D scenes from 2D image observations. During training, each generated 3D scene is differentially rendered from a camera to produce an image that is passed to the discriminator. For this to work, the distribution of camera parameters needs to be closely matched to the camera distribution that gave rise to the training dataset.

This camera distribution can be modeled in an ad-hoc fashion, e.g., by assuming that the camera extrinsics follow a simple distribution, while the intrinsics are heuristically determined and fixed [SLNG20, CMK\*21, ZZZ\*18]. A more faithful distribution of camera extrinsics can be obtained by analyzing the poses of the depicted objects in the training data using computer vision techniques [CLC\*22]. However, not considering the camera intrinsics frequently leads to distortions of proportions. This is a particularly important issue when considering 360° views, which tend to reveal global disproportions quickly. As a remedy, the distribution of all camera parameters can be jointly learned during training [NG21a]. We show that this approach leads to sub-par results compared to our hybrid strategy of estimating camera extrinsics from the training data, and learning the intrinsics as well as refined extrinsics jointly with the generator. PanoHead [AXS\*23] employs a similar strategy for 360° view synthesis, but only considers camera extrinsics. 3DGP [SSX\*23] considers field of view and look-at point for diverse in-the-wild datasets, but is restricted to frontal views. Our design allows us to resolve ambiguities that give rise to geometric distortions in the generated 3D scenes, including the Vertigo effect (Fig. 3), which has been shown to play a crucial role for precise camera control in generative models [LD21].

### 2.4. Relightable Generative Models

Obtaining control over lighting in generative models is an emerging research field. Since the latent space of a trained generative model encodes all image properties including lighting, an informed manipulation of latent codes can be used to coarsely relight the generated images [HHL20, AZMW21, BF22]. This is in contrast to



our approach that injects physical light transport into the model to obtain precise control over lighting.

In the 3D setting, ShadeGAN [PXL\*21] employs a simple Lambertian shading model and directional lighting. We show that this design does not reproduce well the intricate high-frequency view-dependent effects of our setting. Volux-GAN [TFM\*22] extends the lighting to arbitrary environment maps, but requires pseudo ground truth for supervision, which in turn needs to be obtained from costly light stage data [POEL\*21]. Similar to our approach, MesoGAN [DNR\*23] uses generative reflectance fields, but only considers 3D texture shells and requires synthetic training data.

HeadNeRF [HPX\*22] proposes a parametric head model with coarse control over the illumination. Similar to Volux-GAN, they require a dataset with controlled lighting conditions. The recent FaceLit method [RYCT23] enables relighting of human faces in a Spherical Harmonics (SH) representation. However, it requires knowledge of the lighting conditions (low-frequency SH coefficients) for each image in the training set, which we cannot estimate for our dataset. NeRFFaceLighting [JCFG23] distills a pre-trained EG3D into shading and albedo components using also a SH lighting representation. Most similar to our method is LumiGAN [DWW23], which uses environment maps for shading but, in contrast to our work, focuses on visibility and diffuse illumination.

### 3. Overview

In this work, we develop a relightable generative model of cars, which employs advanced physically-based shading under arbitrary distant illumination in the form of environment maps. Importantly, our model is trained from only in-the-wild image collections of cars and a readily available dataset of environment maps, significantly lowering the data requirements for successful training.

First, we deal with the challenge of estimating the distribution of camera parameters used for projecting the generated 3D samples to 2D images (Sec. 4). This is important to obtain correct 3D car shapes that can be viewed from arbitrary directions. Our solution consists of a hybrid strategy that first estimates the distribution from the training data corpus, which is subsequently refined during training.

Second, we devise a novel 3D generator architecture that enables physically-based lighting and shading (Sec. 5), by building a reflectance field generator in conjunction with a fixed-function shading module. Concretely, during training, we sample: 1) an estimated camera from the training data and a latent  $\mathbf{z}_{\text{cam}}$  used to adjust the camera parameters with a neural module, 2) a latent  $\mathbf{z}_{\text{obj}}$  to generate a 3D representation of a car and 3) an environment map  $\mathcal{E}$  from an independent dataset. The complete model  $\mathcal{F}$  can thus be expressed as

$$I = \mathcal{F}(\mathbf{z}_{\text{obj}}, \mathbf{z}_{\text{cam}}, \mathcal{E}) \quad (1)$$

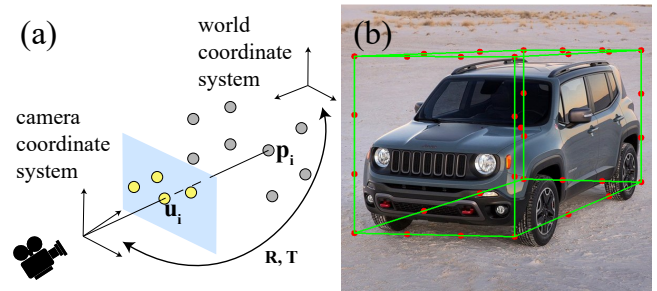
where  $I$  is the resulting image with physically-based lighting. Our trained generator can be used to synthesize 3D car samples that can be lit by any environment map. At inference time we do not apply the adjustment to the camera.

## 4. Estimating the Distribution of Camera Parameters

Reasonable pose information is critical in high-quality 3D generation, providing better quality when the images in the dataset are enhanced with camera pose information [CLC\*22]. This information is used in two ways: 1) a distribution of poses is used to choose a pose for the generator and 2) a pose is provided together with each dataset image to the discriminator as an additional cue.

In previous work, pose estimation techniques have been developed for datasets of objects with well-defined landmarks (eyes, mouth, nose for faces, cats etc.). These datasets also typically have a limited range of camera motion, since the objects are most often viewed from the front [LD21]. Also, previous methods ignore intrinsics typically setting them to some fixed value; we show that for the case of dataset with full rotations, this is insufficient.

We present a two-stage method for estimating the distribution of camera parameters. We first find the poses of car images combining a pre-trained pose estimator and traditional computer vision tools (Sec. 4.1), and then we *learn* intrinsic and extrinsic camera residuals during the GAN training phase (Sec. 4.2).



**Figure 2:** (a) Illustration of the PnP procedure used to recover poses for cars. (b) Example of the projection of the bounding box using the estimated pose.

### 4.1. Estimating Camera Extrinsics

The first step of our approach is to estimate the extrinsic parameters of each image in the dataset that is used by the GAN. We first investigated the quality of the best existing estimators. Most such solutions only provide a subset of the camera parameters we require. Specifically, the recent EgoNet [LYLC21] method is pre-trained on autonomous driving data and estimates camera rotations only. The autonomous driving data [GLSU13] is quite different from the car images we handle, and this domain gap reduces the accuracy of the pose estimator networks.

To estimate extrinsic parameters, we start with the output of EgoNet and estimate the camera-to-world transformation matrix for each image in the dataset using computer vision techniques. Even though EgoNet only estimates rotations, it outputs intermediate 2D key-points corresponding to the projection of the 3D bounding box of the car. We leverage these points to build a pose estimator based on Perspective-n-Point (PnP) [HZ03], an optimization procedure that solves for the full world-to-camera matrix given camera intrinsics and a set of 3D points and their 2D projections.

We create a fixed-size canonical bounding box centered at the





**Figure 3: Vertigo effect.** Left: The camera has a narrow FOV and is located far away from the car. Right: The camera has a wide FOV and is located close to the car. Notice how the car has the same projected size in both cases. We model the corresponding ambiguities in the distribution of cameras to yield distortion-free 3D shapes that look plausible from any viewpoint.

world origin as a proxy for the 3D bounding box of the car. Ideally, the projection of the 3D points  $\mathbf{p}_i$  of the canonical box should match the detected 2D key-points. The 3D points  $\mathbf{p}_i$  in world coordinates and their corresponding 2D points in image plane coordinates  $\mathbf{u}_i$  are related by a change of basis (world-to-camera) and a projection given by the camera intrinsics matrix  $\mathbf{K}$  with focal length  $f$  and offset  $(c_x, c_y)$  (Fig. 2a). To apply PnP, we fix the intrinsics matrix by setting the principal point at the center of the image plane ( $c_x = c_y = 0.5$ ), and the focal length  $f$  corresponding to a field of view of  $50^\circ$  degrees as a first approximation. For this intrinsics matrix, the PnP solver finds the world-to-camera matrix that minimizes the reprojection error, and we then invert it to obtain the camera-to-world matrix, with rotation  $\mathbf{R}$  and translation  $\mathbf{T}$ . An example of the projected 3D points of the bounding box is shown in Fig. 2b.

We apply this process to each image in the dataset of cars, providing approximate pose information for training. The discriminator is conditioned on the camera pose for both real and generated images. To synthesize an image during training, we randomly sample a camera for the training dataset and apply a learned adjustment described next.

#### 4.2. Learning the Distribution of Camera Residuals

There is a well-understood ambiguity between the field of view and camera distance (see Fig. 3). Moreover, these two values are often correlated, which can encourage the generator to synthesize distorted shapes of cars when using a fixed field of view.

For this reason, we *learn* a residual for the focal length, the principal point, and the camera distance during GAN training. To do so, we add a small camera adjustment network  $\mathcal{A}(\mathbf{z}_{cam}, \mathbf{R}, \mathbf{T})$  to our pipeline that takes the estimated extrinsics  $\mathbf{R}, \mathbf{T}$  and a random latent vector  $\mathbf{z}_{cam}$  as input and outputs residuals

$$\Delta f, \Delta c_x, \Delta c_y, \Delta d = \mathcal{A}(\mathbf{z}_{cam}, \mathbf{R}, \mathbf{T}) \quad (2)$$

For the intrinsics, these residuals are simply added to their respective base parameters. The camera position  $\mathbf{T}$  is updated by applying a  $\Delta d$  shift along the  $\frac{\mathbf{T}}{\|\mathbf{T}\|}$  direction. The camera adjustment network handles the ambiguities in the intrinsics and prevents the ambiguities from showing up as distortions in the generated shapes.

We implement the camera adjustment network as a one-layer styled-convolution [KLA\*20] with a tanh activation that takes  $\mathbf{z}_{cam}$

as input and is modulated by the camera extrinsics via a mapping network. During training, we randomly sample a camera from the distribution of estimated poses and apply this network to obtain the adjusted parameters (intrinsics and extrinsics), which we denote  $\mathbf{c}$  for simplicity. The images are then rendered using the adjusted camera  $\mathbf{c}$ .

Using this approach improves overall quality, reducing the effect of distortions and allowing the network to generate more consistent car shapes. Without this correction during training, cars with the same poses would be generated centered at a slightly different position, scale, and suffer from the vertigo effect. (see Sec. 7.5).

### 5. Physically-Based Lighting for 3D GANs

Our goal is a generative model capable of: 1) realistic rendering and 2) interactive performance. The main requirement for realistic rendering is to properly model reflections and in particular, the Fresnel effect that is pronounced in the shiny car paint materials and to shade with the overall environment lighting. Also, efficient rendering is essential for user interaction and most importantly to render the millions of images required to train the model. We first present our shading model (Sec. 5.1), that requires the generation of a base color (Sec. 5.2) and efficient computation of normal maps (Sec 5.3). With these elements we can then perform efficient image-based lighting (Sec. 5.4). An overview of our architecture is given in Fig. ??.

#### 5.1. Shading Model

Consider the rendering equation for a scene without emissive surfaces:

$$L_o(\mathbf{x}, \omega_o) = \int_{\Omega} L_i(\mathbf{x}, \omega_i) f(\mathbf{x}, \omega_o, \omega_i) \cos \theta_{\omega_i} d\omega_i \quad (3)$$

where  $L_o(\mathbf{x}, \omega_o)$  is the outgoing radiance at point  $\mathbf{x}$  in direction  $\omega_o$ ,  $\Omega$  is the hemisphere of directions around  $\mathbf{x}$  and  $L_i$  is the incoming radiance at  $\mathbf{x}$  taken in direction  $\omega_i$ . The material is described by its Bidirectional Reflectance Distribution Function (BRDF)  $f$ , and the incoming light is modulated by the cosine of the angle  $\theta_{\omega_i}$  between the incoming light direction and the normal.

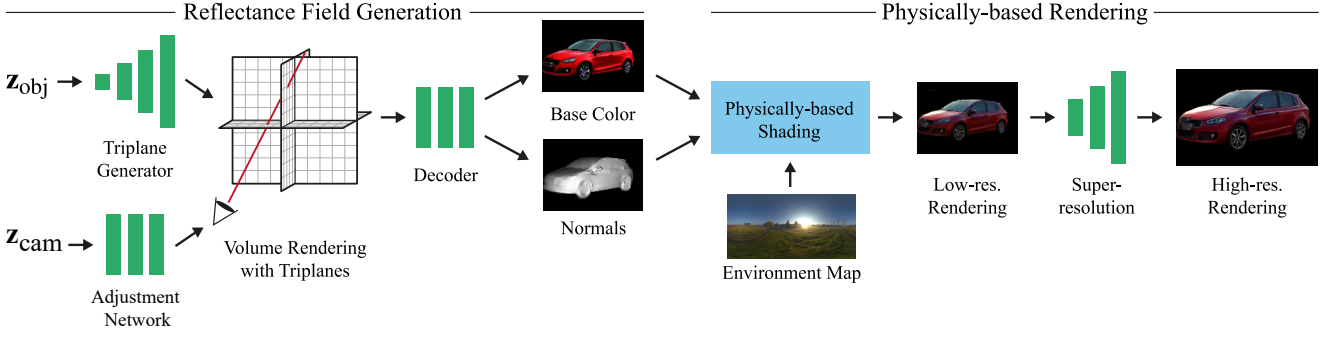
We use a simple Lambertian model for the diffuse, and a microfacet-based BRDF for the specular component [CT82, WMLT07]:

$$f(\mathbf{x}, \omega_i, \omega_o) = \underbrace{\frac{\rho_d}{\pi}}_{\text{diffuse}} + \underbrace{\frac{F(\omega_i, \mathbf{n})G(\omega_i, \omega_o, \mathbf{n})D(\mathbf{n}, \alpha)}{4 \cos \theta_{\omega_i} \cos \theta_{\omega_o}}}_{\text{specular}} \quad (4)$$

where  $\rho_d$  is the diffuse albedo,  $\mathbf{n}$  the surface's normal, and  $\alpha$  the roughness (set to 0.2 throughout). We parametrize the diffuse and specular terms of Eq. 4 with a simplified version of Disney's model [Bur12]

$$\begin{cases} \rho_d = (1 - m)b \\ F_0 = (1 - m)F_0^{\text{diel}} + mb \end{cases} \quad (5)$$

where  $F_0$  is the specular reflectance at normal incidence,  $b$  is the base color,  $m$  is the metallic parameter which we set to  $m = 0.7$  (ranging from 0 for dielectrics to 1 for metals), and  $F_0^{\text{diel}} = 0.04$



**Figure 4:** Our method is divided into two main steps. First, we sample a random vector  $\mathbf{z}_{obj} \in \mathbb{R}^{512}$  and generate a volumetric representation of triplane features. During training, we also sample a camera from the estimated poses of the training data, and apply our camera adjustment  $\mathcal{A}$  with latent  $\mathbf{z}_{cam} \in \mathbb{R}^{512}$ . At inference, the camera is set by the user and the adjustment is not applied. To render from a specific camera viewpoint, we cast a ray for each pixel and sample positions along the ray to compute their corresponding triplane features, which are decoded into base color and density using a lightweight decoder MLP (Sec. 5.2). The base color and density are volume-rendered using Eq. 6. We also compute the surface normals by backpropagating the density through the decoder w.r.t to the position of the sample (Sec. 5.3). Second, we sample an environment map  $\mathcal{E}$  from our dataset and shade our car representation using the efficient physically-based approximation (Sec. 5.4) of a microfacet BRDFs (Sec. 5.1). We render the images at  $128 \times 128$  for efficiency (and memory constraints during training) and then apply a super-resolution network to reach  $256 \times 256$ . Green denotes trainable neural networks and blue fixed functions.

is a typical approximation for the specular reflectance at normal incidence for dielectrics [Kar13].

To use this shading model during training, we need to generate the base color  $b$  and efficiently compute normals so we can evaluate the values of  $\cos \theta_{\omega_i}$  and  $\cos \theta_{\omega_o}$  in the equations above.

## 5.2. Base Color Generator

We generate the base color  $b$  required by our shading model (Eq. 5) using a triplane feature generator [CLC\*22] subsequently decoded by a lightweight MLP (Sec. 6.3) into density and base color and then volume-rendered.

To synthesize the base color during training, we use the adjusted camera parameters  $\mathbf{c}$  (intrinsic and extrinsic) for rendering. To render a pixel, we cast a ray  $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$  from the camera’s origin  $\mathbf{o}$  with direction  $\mathbf{d}$ . We then sample  $N$  points  $\mathbf{x}_i$  along the ray and compute their triplane features  $F_i = F_{i_{xy}} + F_{i_{xz}} + F_{i_{yz}}$ , which are decoded by a lightweight MLP into density  $\sigma_i$  and base color  $b_i$ . Using volumetric rendering [MST\*20], we obtain the final base color

$$b(\mathbf{r}) = \sum_{i=1}^N b_i w_i \quad (6)$$

for the pixel, where  $w_i = (1 - e^{-\sigma_i \delta_i}) e^{-\sum_{j=1}^{i-1} \sigma_j \delta_j}$ , and  $\delta_i = t_{i+1} - t_i$  is the spacing between samples.

## 5.3. Efficient Normal Computation

The normal at any point  $\mathbf{x}$  can be computed by taking the derivative of the density with respect to the position using automatic differentiation [BBJ\*21, BJB\*21, PXL\*21]

$$\mathbf{n}(\mathbf{x}) = -\frac{\nabla_{\mathbf{x}} \sigma(\mathbf{x})}{\|\nabla_{\mathbf{x}} \sigma(\mathbf{x})\|} \quad (7)$$

However, doing this for all the samples on the viewing ray

used during rendering (often hundreds per pixel [MST\*20]) is prohibitively expensive in terms of memory footprint. An alternative and more efficient approach to estimate the normal for the surfaces intersected by a ray  $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ , is to compute the expected termination point  $\mathbf{x}_e$ , i.e., the point corresponding to the object’s surface, directly from the weights  $w_i$  used for volume rendering (Eq. 6)

$$\mathbf{x}_e = \mathbf{r} \left( \sum_{i=1}^N w_i t_i \right) \quad (8)$$

and then take the derivative of the density with respect to  $\mathbf{x}_e$  [BJB\*21]. However, we observed that using only  $\mathbf{x}_e$  for the estimation produced noisy results in practice. To address this, we robustly estimate the normal vector by sampling a few points  $N_e$  along the ray that fall into a small neighborhood  $[-\delta, 0]$  in front of  $\mathbf{x}_e$ , with  $\delta = 0.05$  and  $N_e = 10$  samples (we found this choice to be robust; halving/doubling  $\delta$  did not affect quality). We then combine the normal estimation of each point using the volume rendering equation, similar to ShadeGAN [PXL\*21]:

$$\mathbf{n}(\mathbf{x}_e) = \frac{1}{Z} \sum_{j=1}^{N_e} w_j \frac{-\nabla_{\mathbf{x}} \sigma(\mathbf{x}_j)}{\|\nabla_{\mathbf{x}} \sigma(\mathbf{x}_j)\|} \quad (9)$$

where  $Z$  is a normalization constant to ensure unit norm. The main difference between our approach and ShadeGAN is that we only use points near the expected termination instead of all the samples from volumetric rendering. After obtaining the normal map, we apply an edge-preserving bilateral filter guided by the base color image to reduce blocky artifacts.

## 5.4. Efficient Image-Based Lighting

Given the base color and normal maps, we shade this representation using image-based lighting leveraging the split sum approximation [Kar13]. This method consists of splitting Eq. 3 into two terms, incident light and BRDF, and storing them in precomputed tables

for an efficient fetch at runtime. This approximation results in two precomputations: one for the diffuse term and one for the specular term. The diffuse contribution provides an exact formulation of the convolution between the incoming light and the Lambertian term, which can be precomputed and stored in a single cubemap. The incident light of the specular term is encoded in a pyramid of mipmapped environment maps, whose depth is parameterized by the roughness  $\alpha$ . The BRDF approximation in the specular term consists of a lookup table parameterized by the roughness  $\alpha$  and the cosine of the normal and viewing direction  $\cos\theta_{\omega}$ .

We follow the implementation of the Filament Engine [GA23] for our precomputation and shading. After shading, we composite with a black background, based on a mask recovered from the density. Finally, the image is tone-mapped [RSSF02] and converted into sRGB space [Sto96]. For the final visualization, we composite our rendering with the environment map used for shading, which we map to a hemisphere to avoid the impression that the car is floating in the air. We also add a ground shadow based on the signed distance function of a rounded rectangle using the world space coordinates to improve perceived realism.

## 6. Data, Generator, and Training Loss

The pose and mask estimation along with our efficient shading model are two key elements that allow our method to generate re-lightable cars viewable in  $360^\circ$ . Generating both the environment map and the foreground object with two generators is much more challenging. Different from faces, the appearance of cars is dominated by reflections, and as a result, the environment map needs to be high resolution and much more detailed to create a realistic final image. In our experiments, we found that this task is too hard for the environment map generator, which gives the discriminator an unfair advantage leading to unstable training. To overcome this issue we choose to estimate foreground masks to remove any background pixels and use an auxiliary dataset of environment maps for shading.

In this section, we first explain how we preprocess the car dataset to create the masks (Sec. 6.1). We then give details about the dataset of environment maps which we use to learn variable lighting (Sec. 6.2). Finally, we explain how we modified the generator to allow for more robustness (Sec. 6.3 and Sec. 6.4) and describe the loss and regularization we use for training (Sec.6.5).

### 6.1. Car Dataset Preprocessing

We train our model on the CompCars dataset [YLCLT15]. For each training image, we estimate the pose as described in Sec. 4.1.

Since we want our generated images to contain only the foreground car we also need to estimate a mask for each image. We achieve this using a text-based detection network [LZR\*23] to obtain precise bounding boxes, followed by the state-of-the-art Segmentation Anything Model (SAM) [KMR\*23].

Finally, we pad the images to be square, and resize them to the target resolution of  $256 \times 256$ . We also horizontally mirror each image and its pose to augment the dataset [CLC\*22].

### 6.2. Environment Maps

As we mentioned, we cannot estimate or learn the distribution of environment maps during training, but we still need to explicitly provide the illumination conditions to our shading model to create the final image. The environment maps have a significant impact on the generated cars through the specular and diffuse shading so they need to be detailed and exhibit a wide range of variation. We have found that using an *independent* environment dataset provided high-quality results while maintaining stable training. Specifically, we use environment maps from the Laval Outdoor dataset [HGAL19] and freely available online resources (PolyHaven, iHDR) to shade the cars during training. Finally, we adjust the brightness of the environment maps to better match the lighting conditions of the real images.

### 6.3. Triplane Reparametrization for the Generator

We observed that the original triplane architecture from EG3D [CLC\*22] struggled in our more challenging case of  $360^\circ$  car images. As with EG3D, we start from a random latent vector  $\mathbf{z}_{\text{obj}} \in \mathbb{R}^{512}$  and use a StyleGAN2 generator  $G$  to synthesize a grid of triplane features that define the 3D layout of the object. We observed that the typical approach of synthesizing  $N \times N \times 96$  feature maps and directly taking the three  $N \times N \times 32$  feature planes ( $xy$ ,  $xz$ , and  $yz$ ) from it failed to effectively disentangle the triplanes in our  $360^\circ$  setup.

To address this, we propose a simple yet effective reparameterization of the triplanes inspired by neural texture warps. We use  $G$  to generate  $N \times N \times 32$  feature grid and then explicitly map three regions of the grid to each plane by simply transforming world coordinates  $xyz$  to reparametrized grid coordinates. Concretely, we map the bottom half of the grid to the side plane ( $yz$ ), the top left to the horizontal plane ( $xz$ ), and the top right to the vertical plane ( $xy$ ) (see Fig. 14). Since now each plane occupies only a portion of the triplane image, we increase its resolution from 256 to 512 to compensate for that. This reparameterization results in more interpretable features and smoother surfaces (notably on the sides) because the decoder does not have to disambiguate between the superimposed parts of the car.

### 6.4. Superresolution Network

Similar to previous work [CLC\*22, GLWT21, AXS\*23], we use a super-resolution network due to performance constraints in memory and speed. We base our architecture on StyleNeRF's upsampling blocks [GLWT21], which include pixel-shuffle for faster convolutions [SCH\*16] and low-pass filtering before increasing the resolution of the layer to better address aliasing.

### 6.5. Regularization and Training Loss

To successfully train our generator to produce *base color* (rather than radiance) we apply several regularizations. The problem we solve is more complex than that of the standard 3D GAN generators, since we need to find the base color that, given the environment map and our shading model, will result in rendered cars that are realistic.



**Density Regularization.** Following EG3D, we penalize local variations on the density with an  $\ell_1$  loss

$$\mathcal{L}_{\text{den}} = \frac{1}{N} \sum_{i=1}^N \|\sigma(\mathbf{x}_i) - \sigma(\mathbf{x}_i + \delta\mathbf{x}_i)\|_1 \quad (10)$$

where  $\delta\mathbf{x}_i$  is a small perturbation sampled from a Gaussian distribution.

**Base Color Regularization.** We also penalize local variations of base color with an  $\ell_1$  loss between nearby points to encourage smooth variations

$$\mathcal{L}_{\text{base}} = \frac{1}{N} \sum_{i=1}^N \|b(\mathbf{x}_i) - b(\mathbf{x}_i + \delta\mathbf{x}_i)\|_1 \quad (11)$$

**Binary Density Regularization.** Since we use surface-based shading, we encourage our model to produce well-defined surfaces without floaters. For this, we include a loss term to encourage binary density

$$\mathcal{L}_{\text{binary}} = \|\bar{\sigma} - \max(\bar{\sigma}, \sigma_{\text{min}})\|_1 \quad (12)$$

where  $\bar{\sigma} = \frac{1}{N} \sum_{i=1}^N \sigma(\mathbf{x}_i)$  is the average density,  $\sigma_{\text{min}} = 0.125$ . This loss activates when  $\bar{\sigma}$  falls below  $\sigma_{\text{min}}$  to discourage degenerate cases where the final image is being reconstructed mostly by the decoder and superresolution module and not from actual geometry generated by the triplanes.

We train our model using a non-saturating adversarial loss [GPAM\*14] with  $R_1$  regularization [MGN18] with  $\gamma = 5$ . Our overall training objective also includes the regularization terms:

$$\mathcal{L}_{\text{GAN}} + \frac{\gamma}{2} \mathcal{L}_{R_1} + \lambda_{\text{den}} \mathcal{L}_{\text{den}} + \lambda_{\text{base}} \mathcal{L}_{\text{base}} + \lambda_{\text{binary}} \mathcal{L}_{\text{binary}} \quad (13)$$

We set  $\lambda_{\text{den}} = 0.25$ ,  $\lambda_{\text{binary}} = 0.25$  and  $\lambda_{\text{base}} = 2.5$ . All network parameters, including the camera adjustment network from Sec. 4.2, are trained jointly end-to-end. Unlike other methods that use a progressive training schedule that first generates radiance and then transitions to reflectance for shading [BBJ\*21, BJB\*21, DWW23], we can directly start training with the reflectance representation thanks to our illumination prior. We initialize the training by directly generating the base color and shading with our environment maps; we found this worked better, indicating that our method is capable of directly generating intrinsic appearance.

For training, we first render the low-resolution images at  $64 \times 64$  for 5400K images and then resume rendering them at  $128 \times 128$  until 9400K images. The total number of iterations corresponds to 40 hours of training on 8 A100 GPUs using a batch size of 32. We did not observe noticeable improvements with longer training. We also use ADA augmentations [KAH\*20] but limit its probability to 0.3 to avoid unstable behavior. For volume rendering, we use 64 samples for both the coarse and fine sampling [MST\*20].

## 7. Results and Evaluation

We first show results generated using different latent codes for the GAN generator, with different views lit by several environment maps (Fig. ??), the base color, and a grey shaded car to illustrate the quality of the normals and the mask (lefthand side of the figure). We also show extracted meshes (Fig. 6). Please also see the supplemental video where the accurate motion of highlights is clearly visible.

Moreover, the decoupling of the generation process into base color synthesis and physically-based lighting enables our model to perform material editing (Fig. 7).

We next provide an evaluation of our solution that allows multi-view consistency and with consistent (re)lighting. Specifically, we first discuss a baseline for pose estimation, demonstrating that providing good poses is necessary to obtain any reasonable results. Evaluating the quality of our multi-view and lighting consistent method is hard, especially using quantitative measures. We first present a qualitative comparison, and then a quantitative evaluation based on GAN inversion of synthetic data, that provides ground truth to compute error metrics. We also present several ablations that demonstrate the relative importance of each component of our method.

### 7.1. Camera Estimation

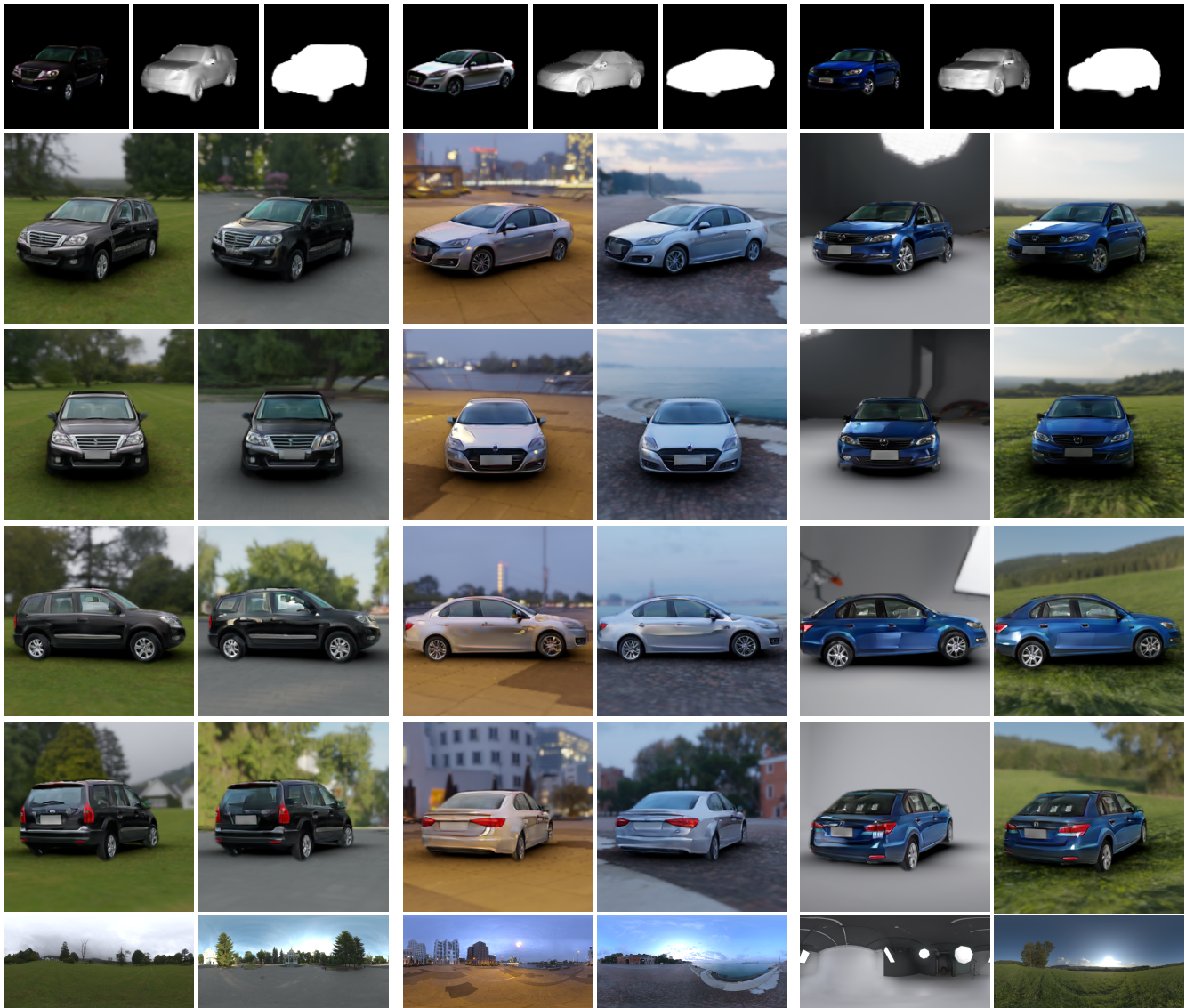
Our pose estimation and refinement is essential to allow  $360^\circ$  rendering of cars. To demonstrate this, we developed a best-effort baseline, building on CAMPARI [NG21a], a method that jointly estimates the camera distribution during GAN training. To do so, we use their camera generator network that consists of an MLP that maps a uniform prior camera distribution to predicted cameras without conditioning the discriminator on the pose. We use this instead of our precomputed poses in our setup and train end-to-end.

This approach did not give satisfactory results; the training failed to generate consistent geometry and appearance of cars and is prone to instabilities. We show examples of the best hand-picked results in Fig. 8. We hypothesize that this is because in our method, we estimate the pose for each image and then condition the discriminator on the poses during training, whereas CAMPARI estimates the distribution of cameras (not individual cameras). Estimating individual cameras enables discriminator conditioning, which guides the generator toward generating cars in a canonical position.

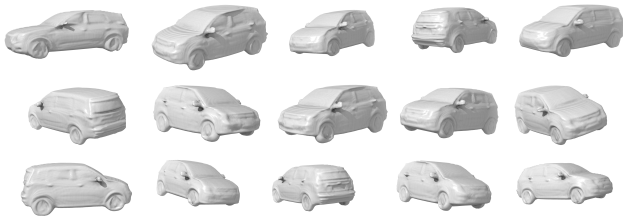
### 7.2. Qualitative Comparisons

We qualitatively compare our approach with three state-of-the-art 3D-aware generators: StyleNeRF [GLWT21], PanoHead [AXS\*23] and EG3D [CLC\*22]. It is important to note that in this comparison, we provide our poses from Sec. 4 to PanoHead and EG3D; we thus denote these PanoHead+ and EG3D+ from now on to indicate that the results already include one of our contributions. The qualitative comparison is shown in Fig. 9 and in the supplemental video where we show  $360^\circ$  videos for all methods. We also tried to train GIRAFFE HD [XLSL22] but the method failed to produce usable results on our masked dataset. For Table 1, we report the FID of the original publication. Despite doing  $360^\circ$  rotations, GIRAFFE HD has noticeable distortions when moving the camera (e.g. different plate positions and back light shapes – please refer to their supplemental).

StyleNeRF can perform  $360^\circ$  rotations for some latents but produces discontinuities in shape and appearance when moving the camera. While EG3D+ appears to hallucinate highlights that can sometimes change with viewpoint, these changes are inconsistent with realistic motion of reflections since the method has no notion of lighting. Such changes with viewpoint are often modeled



**Figure 5:** We showcase different car instances (left to right) from different views (top to bottom) with different environment lighting. On the top we show intrinsic images: the alpha mask, normals and base color for each car. Below we include the environment maps used for shading.



**Figure 6:** Extracted shapes from our model. Note that the bilateral filtering is not applied here.

by changes in density, sometimes resulting in floaters (see first column of Fig. 9, where part of the car is masked by a floater). The inconsistent rendering of reflections is clearly visible in the accompanying video. PanoHead+ avoids the floaters, but produces even less physically accurate renderings of the reflective car-body appearance, essentially “baking” highlights into the 3D representation. This is clearly visible in the video. In contrast, our method creates only the car density without any undesired floaters (Fig. 16) and our physically-based shading module correctly synthesizes view-dependent reflections. We argue that a proper model of light transport, as in our method, is important for better perceived realism of the 3D cars with respect to purely radiance-based approaches [AXS\*23].



**Figure 7:** Our model enables material editing. Given a latent, we modify the hue of the base color in HSV color space and, from top to bottom, change the metallic parameter  $m \in \{0.2, 0.8\}$ . We also vary the roughness  $\alpha \in \{0.2, 0.4, 0.6, 0.8\}$  for a different car.

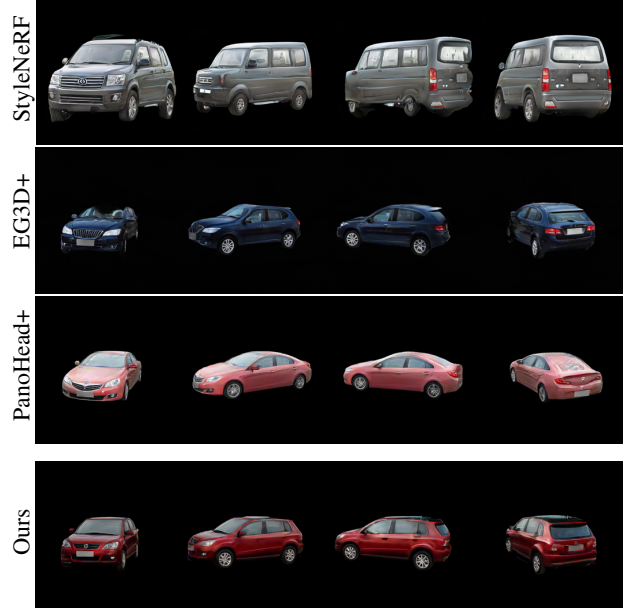


**Figure 8:** Examples of our best-effort baseline trained without our pose estimation, building on CAMPARI [NG21a] instead.

### 7.3. Quantitative Evaluation

Evaluating the multi-view consistency of generative models is a challenging task. Usual metrics such as FID [HRU\*17] and KID [BSAG18] only focus on the similarity between the training and generated distributions in image space, without taking into consideration the 3D capabilities and physical correctness of the model. In addition, these metrics only measure distribution similarity and not strict realism. Since we shade with an independent set of environment maps because we do not have access to the illumination conditions of the car dataset, our method produces out-of-distribution images in terms of lighting, which hurts the scores despite the generated images being realistic [BF22] (see Table 1).

We introduce an evaluation protocol based on GAN inversion to quantify how successful our method is in generating multi-view and lighting consistent 3D models using synthetic data which provides ground truth. We base our evaluation on PTI [RMBCO22], a state-of-the-art approach for single-image inversion. Different from PTI, whose aim is to faithfully invert an image for editing in latent space, our goal is to assess the capability of the model to generate consistent content that matches a known reference. For this, we use synthetic 3D models of realistic cars from which we can produce images with known cameras, both for “training” inversion and for separate unseen test views for evaluation. Concretely, we project a set of  $M$  multi-view images  $\{I_i\}_{i=1}^M$  with cameras  $\{c_i\}_{i=1}^M$



**Figure 9:** Our method is multi-view consistent and produces realistic specular highlights when doing 360° camera rotations around the car. Note how EG3D+ has floaters (leftmost image where the top of the car is occluded), while PanoHead+ has “baked” highlights.

Table 1: To evaluate the distribution of generated images, we report the FID and KID ( $\times 10^3$ ) between the entire dataset and 50K generated images. The † value is taken from the original publication (trained with backgrounds), where the FID is computed between 20K real and generated images.

	FID ↓	KID ↓	360°	Relighting
GIRAFFE HD†	8.36	-	✗	✗
StyleNeRF	<b>4.67</b>	<b>1.66</b>	✗	✗
EG3D+	8.91	2.80	✓	✗
PanoHead+	7.07	2.01	✓	✗
ShadeGAN	48.94	35.65	✗	✓
Ours	17.45	4.92	✓	✓

into latent space. In particular, we take a front, back and two side views. Then, we compare  $M_{\text{test}}$  generated images from unseen camera viewpoints  $\{c_i\}_{i=1}^{M_{\text{test}}}$  with their corresponding reference  $\{I_i\}_{i=1}^{M_{\text{test}}}$  obtained from the 3D model.

For inversion, we first optimize for a vector  $w^* \in \mathcal{W}$  in latent space using Adam [KB14] with a learning rate of 0.1 and betas (0.9, 0.999) for 500 iterations:

$$w^* = \operatorname{argmin}_{w \in \mathcal{W}} \frac{1}{M} \sum_{i=1}^M \mathcal{L}_{LPIPS}(I_i, \mathcal{F}(w, c_i)) + \mathcal{L}_{\ell_2}(I_i, \mathcal{F}(w, c_i)) \quad (14)$$

Second, we fine-tune the learnable parameters of our model  $\mathcal{F}$  while keeping  $w^*$  constant, using Adam with a learning rate of  $3 \times 10^{-4}$  and the same betas, for another 500 iterations. We also remove two bands of 75 pixels on the top and the bottom of the



images to focus more on the car.

$$\mathcal{F}^* = \underset{\mathcal{F}}{\operatorname{argmin}} \frac{1}{M} \sum_{i=1}^M \mathcal{L}_{LPIPS}(I_i, \mathcal{F}(w^*, c_i)) + \mathcal{L}_{\ell_2}(I_i, \mathcal{F}(w^*, c_i)) \quad (15)$$

Once we have the pair  $(w^*, \mathcal{F}^*)$ , can we generate a new set of images corresponding to unseen cameras  $\{c_i\}_{i=1}^{M_{\text{test}}}$  and compute error metrics with the reference views  $\{I_i\}_{i=1}^{M_{\text{test}}}$  for those cameras. We provide FLIP [ANAM\*20], a perceptually motivated image error metric, as well as PSNR, SSIM, and LPIPS. Averaged results for four cars with four test views each are summarized in Table 2 and illustrated in Fig. 10.

As in the qualitative comparison, StyleNerf produces poor quality results. EG3D+ does not represent lighting well compared to the ground truth. PanoHead+ is better but is still worse at faithfully reproducing highlights than our solution. Both EG3D+ and PanoHead+ create a representation based on the four training views and interpolate between them. We hypothesize that the more successful results of PanoHead+ are due to the improvements compared to EG3D+ for 360° viewing, which in turn allows higher quality interpolation. Also, PanoHead+ inversion better captures fine details of the front of the car. Recall however, that in contrast to our solution (see Fig. 11), neither EG3D+ nor PanoHead+ allow relighting after inversion.

Table 2: We compare the multi-view consistency of our method and other state-of-the-art 3D GANs by inverting a set of 360° images into latent space and then generating previously unseen test views.

	FLIP ↓	PSNR ↑	SSIM ↑	LPIPS ↓
StyleNerf	0.230	17.65	0.538	0.340
EG3D+	0.177	20.11	0.696	0.166
PanoHead+	0.143	21.97	<b>0.789</b>	<b>0.101</b>
Ours	<b>0.125</b>	<b>23.10</b>	0.780	0.132

#### 7.4. Relighting

For most previous 3D GANs that have attempted relighting we cannot provide a comparison, since some methods require controlled lighting conditions [HPX\*22, TFM\*22] or knowledge of lighting for each training images [RYCT23], while for LumiGAN [DWW23] no code is available at the time of submission.

We thus compare the relighting capabilities of our model with ShadeGAN [PXL\*21]. Since this method cannot handle 360° rotations, we invert only a single image using the same method as that described above, but following ShadeGAN’s inversion procedure, we also optimize for the light direction. For a fair comparison, given that ShadeGAN uses a Lambertian model, we use an environment map with a soft area light that approximates the appearance ShadeGAN can handle. For the test views we horizontally shift the environment by  $\pm 45^\circ$  in the azimuth direction.

Results averaged over four test cars with two test images for each, are summarized in Table 3 and illustrated in Fig. 12. We see that our method achieves better quantitative results compared to the ground truth, and that ShadeGAN has more difficulty with both the intensity and placement of highlights. In Fig. 12 we also show the

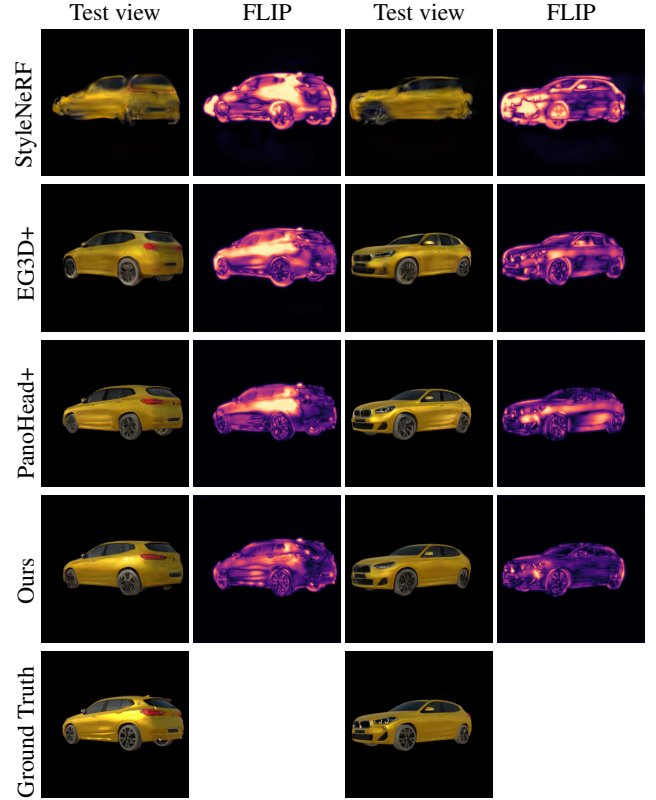


Figure 10: We compare the multi-view consistency of our method by inverting a set of multi-view images and then changing the camera pose. Every second column provides the pixel-wise FLIP error map, showing that our method is the only one capable of moving reflections in a physically accurate way.

normals; as we can see the normals produced by ShadeGAN fail to correctly approximate the shape of the car.

Following ShadeGAN’s default CelebA setup, we trained at a resolution of 128, with cameras sampled from a Gaussian distribution. We did not obtain satisfactory results when sampling cameras on a 360° or even a 180° range. We also observed unstable behavior when increasing the resolution to 256. To compute comparable metrics for our method, we downsample our renders to 128.

Table 3: We compare the relighting capabilities of our method and ShadeGAN by doing single-view inversion and then changing the illumination conditions.

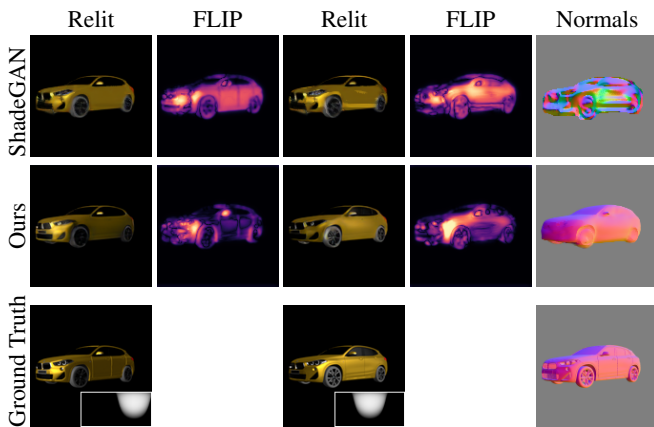
	FLIP ↓	PSNR ↑	SSIM ↑	LPIPS ↓
ShadeGAN	0.148	22.69	0.786	<b>0.094</b>
Ours	<b>0.106</b>	<b>24.46</b>	<b>0.809</b>	0.096

#### 7.5. Ablations

**Camera Adjustment Network.** We show the importance of learning the distribution of camera intrinsics. With our joint optimization of the camera adjustment network during GAN training, at in-



**Figure 11:** Since we decouple base color generator and physically-based shading, we can perform GAN inversion with a given illumination and then relit with a different environment map. Note how the highlights move when using a different environment map.

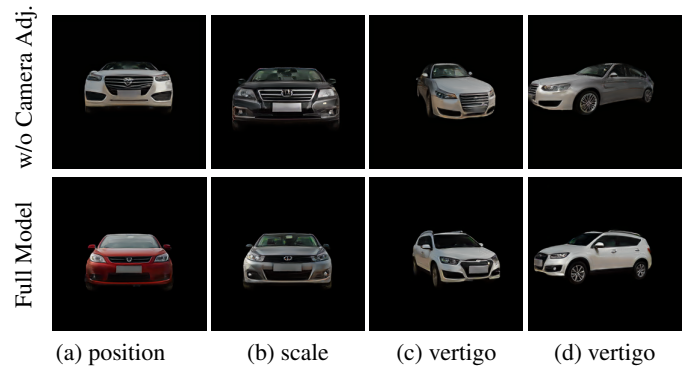


**Figure 12:** We compare the relighting capabilities of our method with those of ShadeGAN by inverting a car in latent space and then changing the illumination conditions, shown on the bottom right of the ground truth image. Every second column provides the pixel-wise FLIP error map. We also show the normals used for shading.

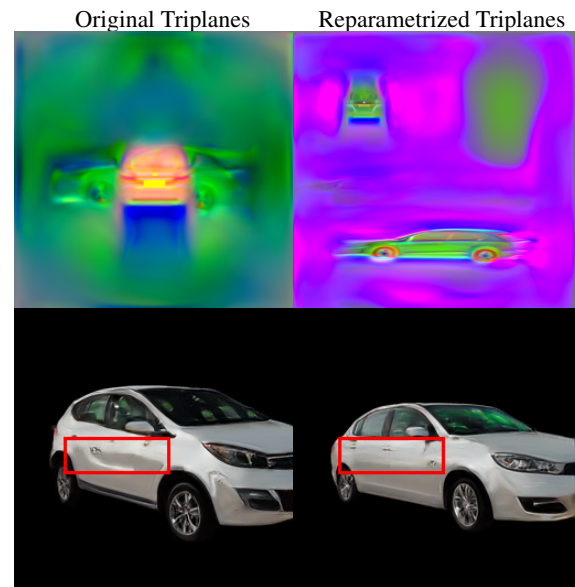
ference time the cars are effectively generated at a canonical position (Fig. 13a), and on a similar scale, avoiding excessively large shapes (Fig. 13b). Lastly, distortions produced by the vertigo effect are prevented (Fig. 13c-d)

**Triplane Reparametrization.** We propose a reparametrization of the triplanes to achieve a better disentanglement between them in our  $360^\circ$  scenario, which avoids the superposition of individual planes (Fig. 14). This has practical impacts on the quality of our renders; it produces smoother surfaces without bumps (which appear when the front of the car is superimposed with the sides), which are particularly important on the side of the cars.

**Bilateral Filtering of the Normals.** Applying an edge-preserving filter trades-off reduced artifacts for smoother (yet accurate) normal maps. We show an example in Fig. 15, where we disable the filtering at inference time.



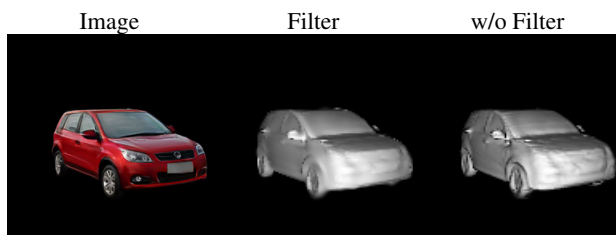
**Figure 13:** Our camera adjustment network encourages the generator to synthesize cars of similar scale in a canonical position and prevents distortions (e.g vertigo effect) in the shape distribution.



**Figure 14:** Our reparametrization of the triplanes effectively disentangles the features of each plane ( $xy$ ,  $xz$ , and  $yz$ ). In contrast, the original triplane features are a superposition of each plane. Reparametrized triplanes produce less bumpy surfaces, which is particularly noticeable on the sides of the car.

**Binary Bensity Regularization.** The binary density term provides a strong constraint on the generated geometry, especially at the beginning of training. Without it, our model cannot be trained properly. Most notably, our method creates only the car density without any undesired floaters (Fig. 16).

**No Physically-based Shading.** To showcase the importance of the physically-based shading model in the perceived realism of the cars, we compare it with the Phong model used in FaceLit [RYCT23] with our prefiltered environment maps. Our shading create more realistic metallic appearance with shiny reflections (Fig. 17), particularly noticeable on the sides of the car.



**Figure 15:** The bilateral filtering of the normals produces smooth normals and removes undesired blocky artifacts.



**Figure 16:** Our binary density regularization helps remove undesired floaters around the car.

## 7.6. Limitations & Future Work

Explicitly modeling the shading process allows us to relight under arbitrary illumination conditions, but it also produces out-of-distribution images in terms of lighting, which results in larger FID and KID scores [BF22].

The shading model itself has some limitations: it is tailored for cars and lacks effects such as multi-layered reflections and sub-surface scattering. Integrating more sophisticated shading models will probably require fine-tuning of the training process. A fully learned appearance model including roughness and metallic is a promising future work.

Our base color estimation is not perfect: there are still some residual shadows and highlights that remain (see Fig. ??, left). It is possible that a more sophisticated lighting model, and some additional priors will allow the estimation to be improved.

Also, similar to previous work [CLC\*22, AXS\*23], our method exhibits artifacts in the form of flickering texture (albeit more noticeable due to the high-frequency reflections) inherited from the StyleGAN2 architecture [KLA\*20] used for the super-resolution network. Some form of temporal regularization would be interesting avenues for future work.

## 8. Conclusions

We introduced a novel relightable 3D generative model capable of synthesizing high-resolution images with interactive rendering for 360° viewing. We specialized in cars to illustrate the difficult case of significant high-frequency reflections on shiny car bodies.



**Figure 17:** Physically-based rendering produces more realistic cars, with accurate metallic appearance.

Our method builds on three components: 1) pose estimation and refinement for training images, 2) the decoupling of the generation process into base color generation and 3) physically-based lighting with environment maps, and a specialized generator network architecture and training approach. Our method allows interactive image synthesis under arbitrary illumination conditions, which we illustrated with multiple examples of different cars and lighting conditioned. We also proposed a quantitative evaluation strategy based on synthetic data and GAN inversion, demonstrating the success of our approach.

Each algorithmic components contributes significantly to its success of our method, that is the first to allow a truly relightable 3D GAN for shiny objects such as car bodies while being fully multi-view consistent.

## 9. Acknowledgments

This research was funded by the ERC Advanced grant FUNGRAPH No 788065 (<https://fungraph.inria.fr>). The authors are grateful to Adobe for generous donations, NVIDIA for a hardware donation, the OPAL infrastructure from Université Côte d’Azur and for the HPC/AI resources from GENCI-IDRIS (Grants 2022-AD011013898 and AD011013898R1). We thank A. Tewari and A. Bousseau for insightful comments on a draft.

## References

- [ANAM\*20] ANDERSSON P., NILSSON J., AKENINE-MÖLLER T., OSKARSSON M., ÅSTRÖM K., FAIRCHILD M. D.: Flip: A difference evaluator for alternating images. *Proc. ACM Comput. Graph. Interact. Tech.* 3, 2 (aug 2020). URL: <https://doi.org/10.1145/3406183>, doi:10.1145/3406183. 11
- [AQW20] ABDAL R., QIN Y., WONKA P.: Image2stylegan++: How to edit the embedded images? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), pp. 8296–8305. 3
- [AXS\*23] AN S., XU H., SHI Y., SONG G., OGRAS U., LUO L.: Panohead: Geometry-aware 3d full-head synthesis in 360°, 2023. *arXiv:2303.13071*. 3, 7, 8, 9, 13
- [AZMW21] ABDAL R., ZHU P., MITRA N. J., WONKA P.: Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows. *ACM Transactions on Graphics (ToG)* 40, 3 (2021), 1–21. 3



- [BBJ\*21] BOSS M., BRAUN R., JAMPANI V., BARRON J. T., LIU C., LENSCH H.: Nerd: Neural reflectance decomposition from image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 12684–12694. 3, 6, 8
- [BEK\*22] BOSS M., ENGELHARDT A., KAR A., LI Y., SUN D., BARRON J. T., LENSCH H., JAMPANI V.: Samurai: Shape and material from unconstrained real-world arbitrary image collections. *arXiv preprint arXiv:2205.15768* (2022). 3
- [BF22] BHATTAD A., FORSYTH D. A.: Stylitgan: Prompting stylegan to produce new illumination conditions. *arXiv preprint arXiv:2205.10351* (2022). 2, 3, 10, 13
- [BGP\*21] BAATZ H., GRANSKOG J., PAPAS M., ROUSSELLE F., NOVÁK J.: Nerf-tex: Neural reflectance field textures. In *Eurographics Symposium on Rendering* (June 2021), The Eurographics Association. 3
- [BJB\*21] BOSS M., JAMPANI V., BRAUN R., LIU C., BARRON J., LENSCH H.: Neural-pil: Neural pre-integrated lighting for reflectance decomposition. *Advances in Neural Information Processing Systems 34* (2021), 10691–10704. 3, 6, 8
- [BPD18] BORA A., PRICE E., DIMAKIS A. G.: AmbientGAN: Generative models from lossy measurements. In *ICLR* (2018). 2, 3
- [BSAG18] BIŃKOWSKI M., SUTHERLAND D. J., ARBEL M., GRETTON A.: Demystifying mmd gans. *arXiv preprint arXiv:1801.01401* (2018). 10
- [Bur12] BURLEY B.: Physically-based shading at disney. 5
- [BXS\*20] BI S., XU Z., SRINIVASAN P., MILDENHALL B., SUNKAVALLI K., HAŠAN M., HOLD-GEOFFROY Y., KRIEGMAN D., RAMAMOORTHI R.: Neural reflectance fields for appearance acquisition. *arXiv preprint arXiv:2008.03824* (2020). 2, 3
- [CBPS20] COLLINS E., BALA R., PRICE B., SUSSTRUNK S.: Editing in style: Uncovering the local semantics of gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 5771–5780. 3
- [CLC\*22] CHAN E. R., LIN C. Z., CHAN M. A., NAGANO K., PAN B., DE MELLO S., GALLO O., GUIBAS L. J., TREMBLAY J., KHAMIS S., ET AL.: Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 16123–16133. 1, 2, 3, 4, 6, 7, 8, 13
- [CMK\*21] CHAN E. R., MONTEIRO M., KELLNHOFFER P., WU J., WETZSTEIN G.: pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2021), pp. 5799–5809. 2, 3
- [CT82] COOK R., TORRANCE K.: A reflectance model for computer graphics. *ACM Trans. Graph.* 1 (01 1982), 7–24. doi:10.1145/965161.806819. 5
- [DNR\*23] DIOLATZIS S., NOVAK J., ROUSSELLE F., GRANSKOG J., AITTALA M., RAMAMOORTHI R., DRETTAKIS G.: Mesogan: Generative neural reflectance shells. In *Computer Graphics Forum* (2023), Wiley Online Library. 4
- [DWW23] DENG B., WANG Y., WETZSTEIN G.: Lumigan: Unconditional generation of relightable 3d human faces. *arXiv preprint arXiv:2304.13153* (2023). 2, 4, 8, 11
- [DYXT22] DENG Y., YANG J., XIANG J., TONG X.: Gram: Generative radiance manifolds for 3d-aware image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 10673–10683. 3
- [GA23] GUY R., AGOPIAN M.: Physically based rendering in filament, 2023. URL: <https://google.github.io/filament/Filament.html>. 7
- [GLSU13] GEIGER A., LENZ P., STILLER C., URTASUN R.: Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research* 32, 11 (2013), 1231–1237. 4
- [GLWT21] GU J., LIU L., WANG P., THEOBALT C.: Stylenerf: A style-based 3d-aware generator for high-resolution image synthesis. *arXiv preprint arXiv:2110.08985* (2021). 1, 3, 7, 8
- [GPAM\*14] GOODFELLOW I., POUGET-ABADIE J., MIRZA M., XU B., WARDE-FARLEY D., OZAIR S., COURVILLE A., BENGIO Y.: Generative adversarial nets. In *Advances in Neural Information Processing Systems* (2014), Ghahramani Z., Welling M., Cortes C., Lawrence N., Weinberger K., (Eds.), vol. 27, Curran Associates, Inc. URL: <https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf>. 3, 8
- [GPM\*22] GAL R., PATASHNIK O., MARON H., BERMANO A. H., CHECHIK G., COHEN-OR D.: Stylegan-nada: Clip-guided domain adaptation of image generators. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–13. 3
- [GSW\*22] GAO J., SHEN T., WANG Z., CHEN W., YIN K., LI D., LITANY O., GOJCIC Z., FIDLER S.: Get3d: A generative model of high quality 3d textured shapes learned from images. In *Advances In Neural Information Processing Systems* (2022). 3
- [HGAL19] HOLD-GEOFFROY Y., ATHAWALE A., LALONDE J.-F.: Deep sky modeling for single image outdoor lighting estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 6927–6935. 2, 7
- [HHL\*20] HÄRKÖNEN E., HERTZMANN A., LEHTINEN J., PARIS S.: Ganspace: Discovering interpretable gan controls. *Advances in Neural Information Processing Systems 33* (2020), 9841–9850. 3
- [HMR19] HENZLER P., MITRA N. J., RITSCHER T.: Escaping plato’s cave: 3d shape from adversarial rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 9984–9993. 2, 3
- [HPX\*22] HONG Y., PENG B., XIAO H., LIU L., ZHANG J.: Headnerf: A real-time nerf-based parametric head model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 20374–20384. 4, 11
- [HRU\*17] HEUSEL M., RAMSAUER H., UNTERTHINER T., NESSLER B., HOCHREITER S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems 30* (2017). 10
- [HZ03] HARTLEY R., ZISSERMAN A.: *Multiple view geometry in computer vision*. Cambridge university press, 2003. 4
- [JCFG23] JIANG K., CHEN S.-Y., FU H., GAO L.: Nerffacelight: Implicit and disentangled face lighting representation leveraging generative prior in neural radiance fields. *ACM Trans. Graph.* 42, 3 (jun 2023). URL: <https://doi.org/10.1145/3597300>, doi: 10.1145/3597300. 4
- [JLX\*23] JIN H., LIU I., XU P., ZHANG X., HAN S., BI S., ZHOU X., XU Z., SU H.: Tensor: Tensorial inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 165–174. 3
- [KAH\*20] KARRAS T., AITTALA M., HELLSTEN J., LAINE S., LEHTINEN J., AILA T.: Training generative adversarial networks with limited data. *Advances in Neural Information Processing Systems 33* (2020), 12104–12114. 3, 8
- [KAL\*21] KARRAS T., AITTALA M., LAINE S., HÄRKÖNEN E., HELLSTEN J., LEHTINEN J., AILA T.: Alias-free generative adversarial networks. In *Proc. NeurIPS* (2021). 3
- [Kar13] KARIS B.: Real shading in unreal engine 4. 6
- [KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 10
- [KLA19] KARRAS T., LAINE S., AILA T.: A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 4401–4410. 3

- [KLA\*20] KARRAS T., LAINE S., AITTALA M., HELLSTEN J., LEHTINEN J., AILA T.: Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), pp. 8110–8119. 3, 5, 13
- [KMR\*23] KIRILLOV A., MINTUN E., RAVI N., MAO H., ROLLAND C., GUSTAFSON L., XIAO T., WHITEHEAD S., BERG A. C., LO W.-Y., DOLLÁR P., GIRSHICK R.: Segment anything. *arXiv:2304.02643* (2023). 7
- [KPACO21] KAFRI O., PATASHNIK O., ALALUF Y., COHEN-OR D.: Stylefusion: A generative model for disentangling spatial segments. *arXiv preprint arXiv:2107.07437* (2021). 3
- [LD21] LEIMKÜHLER T., DRETTAKIS G.: Freestylegan: Free-view editable portrait rendering with the camera manifold. doi:10.1145/3478513.3480538. 3, 4
- [LGT\*23] LIN C.-H., GAO J., TANG L., TAKIKAWA T., ZENG X., HUANG X., KREIS K., FIDLER S., LIU M.-Y., LIN T.-Y.: Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 300–309. 3
- [LTL\*22] LYU L., TEWARI A., LEIMKUEHLER T., HABERMANN M., THEOBALT C.: Neural radiance transfer fields for relightable novel-view synthesis with global illumination. In *ECCV* (2022). 3
- [LYLC21] LI S., YAN Z., LI H., CHENG K.-T.: Exploring intermediate representation for monocular vehicle pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 1873–1883. 4
- [LZR\*23] LIU S., ZENG Z., REN T., LI F., ZHANG H., YANG J., LI C., YANG J., SU H., ZHU J., ET AL.: Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499* (2023). 7
- [MGN18] MESCHEDER L., GEIGER A., NOWOZIN S.: Which training methods for gans do actually converge? In *International conference on machine learning* (2018), PMLR, pp. 3481–3490. 8
- [MHS\*22] MUNKBERG J., HASSELGREN J., SHEN T., GAO J., CHEN W., EVANS A., MÜLLER T., FIDLER S.: Extracting triangular 3d models, materials, and lighting from images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 8280–8290. 3
- [MST\*20] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHI R., NG R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision* (2020), Springer, pp. 405–421. 2, 3, 6, 8
- [NG21a] NIEMEYER M., GEIGER A.: Campari: Camera-aware decomposed generative neural radiance fields. In *2021 International Conference on 3D Vision (3DV)* (2021), IEEE, pp. 951–961. 2, 3, 8, 10
- [NG21b] NIEMEYER M., GEIGER A.: Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 11453–11464. 3
- [PBJM22] POOLE B., JAIN A., BARRON J. T., MILDENHALL B.: Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988* (2022). 3
- [POEL\*21] PANDEY R., ORTS-ESCOLANO S., LEGENDRE C., HAENE C., BOUAZIZ S., RHEMANN C., DEBEVEC P., FANELLO S.: Total relighting: Learning to relight portraits for background replacement. vol. 40. doi:10.1145/3450626.3459872. 4
- [PWS\*21] PATASHNIK O., WU Z., SHECHTMAN E., COHEN-OR D., LISCHINSKI D.: Styleclip: Text-driven manipulation of stylegan imagery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 2085–2094. 3
- [PXL\*21] PAN X., XU X., LOY C. C., THEOBALT C., DAI B.: A shading-guided generative implicit model for shape-accurate 3d-aware image synthesis. *Advances in Neural Information Processing Systems* 34 (2021), 20002–20013. 2, 4, 6, 11
- [RMBCO22] ROICH D., MOKADY R., BERMANO A. H., COHEN-OR D.: Pivotal tuning for latent-based editing of real images. *ACM Transactions on Graphics (TOG)* 42, 1 (2022), 1–13. 10
- [RSSF02] REINHARD E., STARK M., SHIRLEY P., FERWERDA J.: Photographic tone reproduction for digital images. *ACM Trans. Graph.* 21, 3 (jul 2002), 267–276. URL: <https://doi.org/10.1145/566654.566575>, doi:10.1145/566654.566575. 7
- [RYCT23] RANJAN A., YI K. M., CHANG J.-H. R., TUZEL O.: Facelit: Neural 3d relightable faces. In *CVPR* (2023). URL: <https://arxiv.org/abs/2303.15437>. 2, 4, 11, 12
- [SCH\*16] SHI W., CABALLERO J., HUSZÁR F., TOTZ J., AITKEN A. P., BISHOP R., RUECKERT D., WANG Z.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 1874–1883. 7
- [SDZ\*21] SRINIVASAN P. P., DENG B., ZHANG X., TANCIK M., MILDENHALL B., BARRON J. T.: Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 7495–7504. 3
- [SLNG20] SCHWARZ K., LIAO Y., NIEMEYER M., GEIGER A.: Graf: Generative radiance fields for 3d-aware image synthesis. *Advances in Neural Information Processing Systems* 33 (2020), 20154–20166. 2, 3
- [SSX\*23] SKOROKHOV I., SIAROHIN A., XU Y., REN J., LEE H.-Y., WONKA P., TULYAKOV S.: 3d generation on imagenet. In *International Conference on Learning Representations* (2023). URL: <https://openreview.net/forum?id=U2WjB9xxZ9q>. 3
- [Sto96] STOKES M.: A standard default color space for the internet-srgb. <http://www.color.org/contrib/sRGB.html> (1996). 7
- [STWW22] SKOROKHOV I., TULYAKOV S., WANG Y., WONKA P.: Epigraf: Rethinking training of 3d gans. *arXiv preprint arXiv:2206.10535* (2022). 3
- [TBRP\*22] TEWARI A., B R M., PAN X., FRIED O., AGRAWALA M., THEOBALT C.: Disentangled3d: Learning a 3d generative model with disentangled geometry and appearance from monocular images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (june 2022), IEEE. 3
- [TEB\*20] TEWARI A., ELGHARIB M., BHARAJ G., BERNARD F., SEIDEL H.-P., PÉREZ P., ZÖLLHOFFER M., THEOBALT C.: Stylerig: Rigging stylegan for 3d control over portrait images, cvpr 2020. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (june 2020), IEEE. 3
- [TFM\*22] TAN F., FANELLO S., MEKA A., ORTS-ESCOLANO S., TANG D., PANDEY R., TAYLOR J., TAN P., ZHANG Y.: Volux-gan: A generative model for 3d face synthesis with hdri relighting. *arXiv preprint arXiv:2201.04873* (2022). 4, 11
- [VHM\*21] VERBIN D., HEDMAN P., MILDENHALL B., ZICKLER T., BARRON J. T., SRINIVASAN P. P.: Ref-nerf: Structured view-dependent appearance for neural radiance fields. *arXiv preprint arXiv:2112.03907* (2021). 3
- [WLW\*23] WANG Z., LU C., WANG Y., BAO F., LI C., SU H., ZHU J.: Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *arXiv preprint arXiv:2305.16213* (2023). 3
- [WMLT07] WALTER B., MARSCHNER S. R., LI H., TORRANCE K. E.: Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques* (Goslar, DEU, 2007), EGSR'07, Eurographics Association, p. 195–206. 5
- [WSLG23] WU T., SUN J.-M., LAI Y.-K., GAO L.: De-nerf: Decoupled neural radiance fields for view-consistent appearance editing and high-frequency environmental relighting. In *ACM SIGGRAPH 2023 Conference Proceedings* (New York, NY, USA, 2023), SIGGRAPH '23, Association for Computing Machinery. URL: <https://doi.org/10.1145/3588432.3591483>, doi:10.1145/3588432.3591483. 3

- [XLSL22] XUE Y., LI Y., SINGH K. K., LEE Y. J.: Giraffe hd: A high-resolution 3d-aware generative model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 18440–18449. 1, 3, 8
- [XTS\*22] XIE Y., TAKIKAWA T., SAITO S., LITANY O., YAN S., KHAN N., TOMBARI F., TOMPKIN J., SITZMANN V., SRIDHAR S.: Neural fields in visual computing and beyond. In *Computer Graphics Forum* (2022), vol. 41, Wiley Online Library, pp. 641–676. 3
- [YLCLT15] YANG L., LUO P., CHANGE LOY C., TANG X.: A large-scale car dataset for fine-grained categorization and verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 3973–3981. 2, 7
- [ZLW\*21] ZHANG K., LUAN F., WANG Q., BALA K., SNAVELY N.: PhysSG: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021). 3
- [ZSD\*21] ZHANG X., SRINIVASAN P. P., DENG B., DEBEVEC P., FREEMAN W. T., BARRON J. T.: Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–18. 3
- [ZXNT21] ZHOU P., XIE L., NI B., TIAN Q.: Cips-3d: A 3d-aware generator of gans based on conditionally-independent pixel synthesis. *arXiv preprint arXiv:2110.09788* (2021). 3
- [ZZZ\*18] ZHU J.-Y., ZHANG Z., ZHANG C., WU J., TORRALBA A., TENENBAUM J., FREEMAN B.: Visual object networks: Image generation with disentangled 3d representations. *Advances in neural information processing systems* 31 (2018). 3