Neural Graphics Pipeline for Controllable Image Generation

Xuelin Chen Shandong University
Daniel Cohen-Or Tel Aviv University
Baoquan Chen Peking University
Niloy J. Mitra University College London, Adobe Research

Abstract

We present *Neural Graphics Pipeline* (NGP), a hybrid generative model that brings together neural and traditional image formation models. NGP generates coarse 3D models that are fed into neural rendering modules to produce view-specific interpretable 2D maps, which are then composited into the final output image using a traditional image formation model. Our approach offers control over image generation by providing direct handles controlling illumination and camera parameters, in addition to control over shape and appearance variations. The key challenge is to learn these controls through unsupervised training that links generated coarse 3D models with unpaired real images via neural and traditional (e.g., Blinn-Phong) rendering functions, without establishing an explicit correspondence between them. We evaluate our hybrid modeling framework, compare with neural-only generation methods (namely, DCGAN, LSGAN, WGAN-GP, VON, and SRNs), report improvement in FID scores against real images, and demonstrate that NGP supports direct controls common in traditional forward rendering. Code, data, and trained models will be released.

1 Introduction

Computer graphics produces images by *forward rendering* 3D scenes. While this traditional approach provides *controllability* in the form of direct manipulation of camera, illumination, and other rendering parameters, the main bottleneck of the classic approach is content creation, that is the explicit need to author detailed scenes. Neural networks have recently given raise to *neural rendering* as an alternative approach wherein specialized networks are trained end-to-end to operate on deep features stored on sparse geometry (e.g., voxels [1, 2, 3, 4, 5, 6, 7], points [8], surface patches [9]) to directly produce pixel colors. Neural rendering revolutionizes image synthesis workflow by bypassing the content creation stage, however, they lack the level of controllability supported in traditional rendering.

We introduce *Neural Graphics Pipeline* (NGP), a hybrid generative approach that uses neural network to produce coarse 3D content, decorated with view-specific interpretable 2D features, that can then be consumed by traditional image formation models – see Figure 1. The approach relaxes the need for modeling a fully detailed scene model, while retaining the same traditional direct control over illumination and camera, in addition to indirect control over shape and appearance variations.

NGP (see Figure 2) consists of four modules: (i) a GAN-based generation of a coarse 3D model, (ii) a projection module that renders the coarse geometry into a 2D depth map, (iii) a set of GANs to produce image-space interpretable appearance features (i.e., normal, diffuse albedo, specular map, roughness), and (iv) a 2D renderer that takes these appearance maps along with user-provided conventional illumination (i.e., light positions with intensity) to produce the final images.

Training NGP is challenging because there is no direct supervision available in terms of paired or unpaired input and corresponding 2D interpretable features. By interpretable features, we refer to

Preprint. Under review.

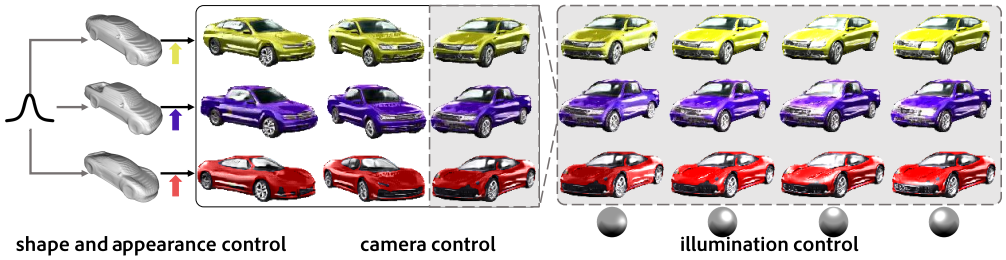

Figure 1: NGP, a GAN-based model, samples a coarse 3D model, and provides full control over camera and illumination, and responds to geometry and appearance edits. NGP is trained directly on unlabelled real images. Mirrored balls (right-bottom) indicate corresponding illumination setting.

2D feature maps that are used in traditional imaging models and, hence, can be combined with fixed and known image formation models (e.g., Blinn-Phong) along with illumination information to produce a final image. We present an *unsupervised* learning setup for the proposed neural modeling framework. Note that by generating interpretable intermediate maps, we link the 3D and 2D images without any explicit correspondence information between them. The core of NGP consists of a network that parameterically translates a depth image to an image with realistic appearance. These additional parameters, which disambiguate the translation, are in fact the handles that controls the image generation of the trained network. A notable feature of NGP, which is based on unsupervised unpaired training, is the ability of collectively learn from synthetic data and real images.

We extensively evaluate our hybrid modeling framework against several competing neural-only image generation approaches [1, 10, 11, 6, 7], rate the different methods using the established FID score [12, 13], and present ablation studies to show the importance of our design choices. Our tests demonstrate the superiority of our method (i.e., lower FID scores) compared to other state-of-the-art alternatives, on both synthetic and real data.

2 Related Work

GAN-based image generation. Since the introduction of Generative Adversarial Nets (GANs) [14], many GAN variants [1, 10, 11, 15] have been proposed to synthesize images conditioned on control variables sample from a Gaussian distribution. State-of-the-art GAN-based methods are now able to generate images with high level of realism [16, 17, 18]. While it is increasingly possible to provide guidance through conditional latent code [19, 20], structured latent space [21, 22, 23, 24], style example [25], or semantic specifications [26], it still remains difficult to directly control generated imagery by updating all of geometry, camera, illumination, or material parameters. We were particularly inspired by the recently proposed visual object network [6] that takes a generated rough shape and trains a 2D texture network for adding texture to synthesize images. Different from ours, they directly output final RGB images, and do not provide access to interpretable intermediate features, and thus, prevents direct illumination control. We use unsupervised training, avoiding associating images with attributes or tags to allow scaling in terms of variety, richness, and realism.

3D generative neural networks. Researchers have also developed various generative networks for automatic content creation, ranging from single object generation [27, 28, 29, 30, 31, 32, 33, 34], indoor scene synthesis [35, 36, 37, 38], urban landscape and terrain generation [39, 40]. The generated geometry, however, is still not sufficiently detailed and/or assigned with plausible materials, to be directly rendered by traditional forward rendering to produce high-quality images.

Neural rendering. A particularly exciting breakthrough is neural rendering, where deep features are learned on coarse geometry (e.g., voxels, points), and then neurally rendered to produce a final image. Most of the proposed approaches use supervised training and/or largely target novel view synthesis task [2, 41, 5, 3, 42, 8, 43, 7, 44, 45], with the output optionally conditioned using latent vectors (e.g., appearance vectors in [5]). In the unsupervised setting, GQN [46] and HoloGAN [4] allow camera manipulation and model complex background clutter. However, since the learned features are deep, they cannot, yet, be manipulated using traditional CG controls. For example, one cannot freely control illumination in such an image generation pipeline.

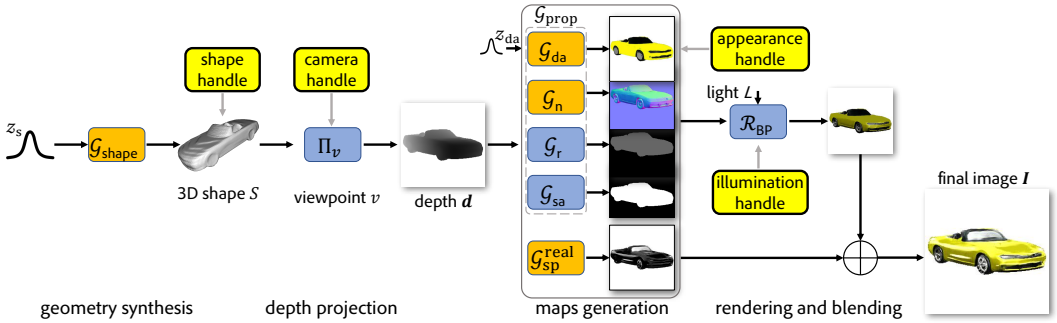

Figure 2: **NGP at inference time.** At test time, starting from a sampled noise vector z_{s} and a set of user control signals (marked in yellow), NGP uses a combination of learned networks (marked in mustard) and fixed functions (marked in blue) to produce a range of interpretable feature maps, which are then combined to produce a final image $\boldsymbol{I}$.

3 Formulation

Traditional computer graphics follows a *model-and-render* pipeline, where a 3D scene is first modeled, and an image is then produced by rendering the 3D scene via a conventional renderer, a process that simulates the flow of light in physical world. Neural Graphics Pipeline, Figure 2 presents an overview, follows a similar paradigm: we first sample a coarse 3D shape using a neural network, followed by a set of learned generators producing view-specific interpretable reflectance property maps, along with a neural-rendered specular map. We assume the reflectance of the viewed content in the scene is characterized by a set of property maps: diffuse albedo, surface normal, monochrome roughness and specular albedo, which are then combined using the *Blinn-Phong Reflection Model*.

A coarse shape S is generated from a latent shape code $S := \mathcal{G}_{\text{shape}}(z_{\text{s}})$, the shape is then projected from a viewpoint sample v to form a 2D depth map $\boldsymbol{d} := \Pi_v(S)$. The maps generation module then produces a set of intermediate maps, with z_{da} controlling the appearance (diffuse albedo). The generated reflectance maps from $\mathcal{G}_{\text{prop}} := (\mathcal{G}_{\text{da}}, \mathcal{G}_{\text{n}}, \mathcal{G}_{\text{r}}, \mathcal{G}_{\text{sa}})$ are then fed into a fixed function *Blinn-Phong* renderer $\mathcal{R}_{\text{BP}}$ (see Sec. A.1 in supplementary for details) to illuminate the viewed content under a given light setting L. Blending the resultant rendering image with the realistic specular map generated by $\mathcal{G}_{\text{sp}}^{\text{real}}$, our image formation flow generates the final image,

$$\boldsymbol{I} := \mathcal{R}_{\text{BP}}\left(\mathcal{G}_{\text{prop}}(\Pi_v(\mathcal{G}_{\text{shape}}(z_{\text{s}})), z_{\text{da}}), L\right) + \mathcal{G}_{\text{sp}}^{\text{real}}(\Pi_v(\mathcal{G}_{\text{shape}}(z_{\text{s}}))), \tag{1}$$

by sampling the space of (z_{s}, v, z_{da}, L) at inference time.

NGP provides the user with several handles (highlighted in yellow in Figure 2) to control the output image: (i) a *camera handle* offers direct control to rotate the camera view; (ii) a *illumination handle* offers direct control to specify the lights (position, intensity, and count); (iii) a *shape handle* to control the coarse geometry via direct editing and latent control; and (iv) an *appearance handle* to manipulate the appearance of the object via direct editing and latent control. The NGP is designed such that the output image meaningfully adapts to the user specifications. Next, we detail the individual modules in NGP and elaborate on how we train the networks without intermediate supervision.

Learning geometry synthesis. We start with a category-specific 3D shape prior to capture rough geometry of the object, *without* any reflectance properties. We adopt the recently proposed IM-GAN [28], which uses shape latent codes to produce implicit signed distance fields corresponding to realistic shapes, although alternate 3D generative models can be used.

More specifically, we pretrain a 3D autoencoder with the implicit field decoder to produce a compact shape latent space for representing 3D shape implicit fields and use latent-GAN [47] on the trained shape latent space to produce realistic shape latent codes. As a result, we learn a generator $\mathcal{G}_{\text{shape}}$ to map the Gaussian-sampled shape code z_{s} to a shape $S := \mathcal{G}_{\text{shape}}(z_{\text{s}})$.

Depth projection. Next, we project the coarse shape to 2D via a direct depth projection layer. Given a sampled shape S and a sampled viewpoint v, which is parameterized by an extrinsic matrix $\boldsymbol{E} := [\boldsymbol{R}|\boldsymbol{t}] \in \mathbb{R}^{3\times4}$ and camera intrinsics $\boldsymbol{K} \in \mathbb{R}^{3\times3}$, we obtain a coarse depth map $\boldsymbol{d}$ by projecting every *visible* point p (in homogeneous coordinates) on the surface S as, $\boldsymbol{d} := \boldsymbol{K}\boldsymbol{E}p, \forall p \in S$. We use

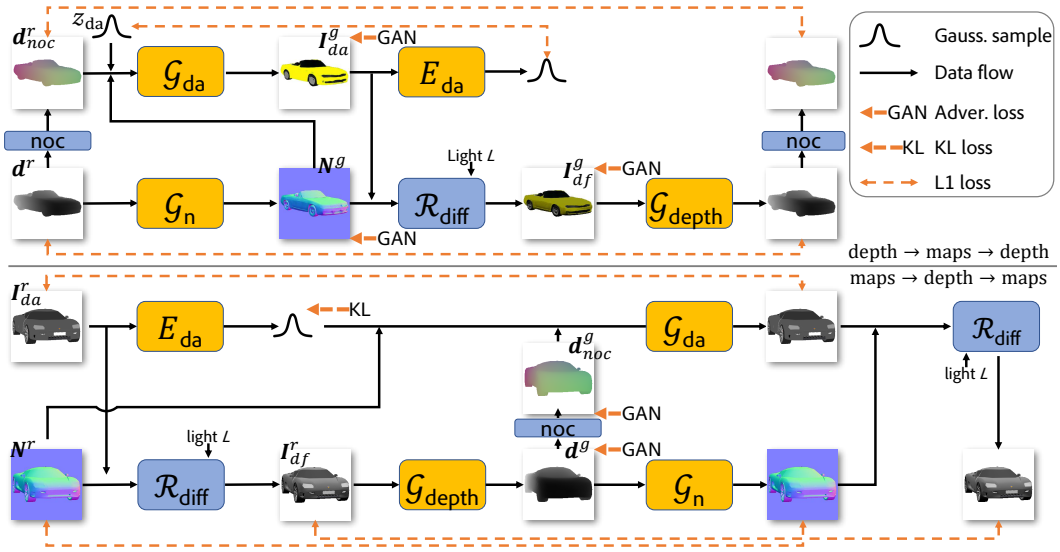

Figure 3: **Learning reflectance maps.** The proposed architecture for training to jointly generate reflectance property maps from depth images using adversarial losses and cycle consistency losses to enable unpaired training. Top: the cycle between the real depth image and the generated diffuse image composited from generated reflectance maps. Bottom: the cycle between real diffuse image composited from real reflectance maps, and the generated depth maps.

OpenGL calls for efficient depth rendering and rasterization. As we shall demonstrate, the original coarse depth map is itself sufficient for our end goal. Although we train $\mathcal{G}_{\text{shape}}$ separately, if desired, NGP can be linked to a differentiable depth rendering layer and trained end-to-end.

Learning reflectance maps generation. Next, we elaborate on the modules to generate reflectance maps from a coarse depth map $\boldsymbol{d}$, including two constant function modules ($\mathcal{G}_{\text{sa}}$ and $\mathcal{G}_{\text{r}}$, for specular albedo and roughness maps, respectively) and two learned networks ($\mathcal{G}_{\text{da}}$ and $\mathcal{G}_{\text{n}}$, for diffuse albedo and normal maps, respectively).

(i) Specular albedo and roughness maps generation. In absence of the diverse specular albedo and roughness data to learn the data prior, we simply realize $\mathcal{G}_{\text{sa}}$ and $\mathcal{G}_{\text{r}}$ as constant functions of the form: $\mathcal{G}_{\text{sa}}(\boldsymbol{d}) : \boldsymbol{I}^g_{sa} = c\boldsymbol{M}(\boldsymbol{d})$ and $\mathcal{G}_{\text{r}}(\boldsymbol{d}) : \boldsymbol{\alpha}^g = \alpha\boldsymbol{M}(\boldsymbol{d})$, where $\boldsymbol{I}^g_{sa}$ is the generated specular albedo map, $\boldsymbol{\alpha}^g$ the generated roughness map, $\boldsymbol{M}(\cdot)$ generates the mask of $\boldsymbol{d}$ by thresholding the depth, c is a constant specular albedo (set to white) and α is a constant roughness (set to 4.0).

(ii) Learning to generate diffuse albedo and normal maps. For learning $\mathcal{G}_{\text{da}}$ and $\mathcal{G}_{\text{n}}$, we only have access to 'real' reflectance maps that comprise of real diffuse albedo maps $\mathbb{I}^r_{da} = \{\boldsymbol{I}^r_{da}\}$ and detailed normal maps $\mathbb{N}^r = \{\boldsymbol{N}^r\}$, along with corresponding viewpoints. Note that, given the light setting L, each set of real reflectance maps, denoted by $(\boldsymbol{I}^r_{da}, \boldsymbol{N}^r)$, can be used to render a real diffuse image $\boldsymbol{I}^r_{df}$ using the diffuse reflection component (denoted as $\mathcal{R}_{\text{diff}}$) of the Blinn-Phong equation.

Given the coarse depth image $\boldsymbol{d}$ and the viewpoint v parameterized by $\boldsymbol{E}$ and $\boldsymbol{K}$, the task is then to synthesize a pair of generated reflectance maps $(\boldsymbol{I}^g_{da}, \boldsymbol{N}^g)$ that can be used to render a diffuse image $\boldsymbol{I}^g_{df}$. Training with supervision would be relatively easy, and can be seen as a standard task. However, we do not have access to ground truth maps for supervision, i.e., the shape generated from the shape network comes *without* any ground truth reflectance properties. Hence, we treat this as an unpaired image-to-image translation problem. Our key idea is to do a cycle translation between the depth map and the diffuse image (i.e., product of the diffuse albedo and detailed normal map), via the fixed rendering function $\mathcal{R}_{\text{diff}}$. Specifically, we design a cycle-consistent adversarial network that *jointly* generates $(\boldsymbol{I}^g_{da}, \boldsymbol{N}^g)$ from $\boldsymbol{d}$. Figure 3 shows the proposed architecture.

Given 'real' depth maps $\mathbb{D}^r = \{\boldsymbol{d}^r\}$ produced from the depth projection, we train a network $\mathcal{G}_{\text{n}}$ to generate a detailed normal map $\boldsymbol{N}^g = \mathcal{G}_{\text{n}}(\boldsymbol{d}^r)$ that fools a discriminator trained to differentiate the real and generated detailed normal maps, and another network $\mathcal{G}_{\text{da}}$ to generate a diffuse albedo map $\boldsymbol{I}^g_{da}$ that fools a diffuse albedo map discriminator (Figure 3-top). Note that we do *not* enforce

one-to-one mapping from depth maps to diffuse albedo maps, but rather condition the generation using random Gaussian sample code z_{da}. In practice, we found the network $\mathcal{G}_{\text{da}}$ difficult to train in the absence of 3D object-space coordinates, as opposed to the view-dependent camera-space coordinates provided by the depth map. Hence, we use the intrinsic $\boldsymbol{K}$ and extrinsic $\boldsymbol{E}$ camera parameters, to enrich $\boldsymbol{d}^r$ to the normalized object coordinates (NOC) [48] system to obtain $\boldsymbol{d}^r_{\text{noc}} := noc(\boldsymbol{d}^r, \boldsymbol{K}, \boldsymbol{E})$. Further, we found that the generated normal map $\boldsymbol{N}^g$ helps generating the diffuse albedo, as the detailed normal map provides more detailed geometry information. Therefore, we give $\mathcal{G}_{\text{da}}$ as input $\boldsymbol{d}^r_{\text{noc}}$, $\boldsymbol{N}^g$, and z_{da} resulting in: $\boldsymbol{I}^g_{da} := \mathcal{G}_{\text{da}}(\boldsymbol{d}^r_{\text{noc}}, \boldsymbol{N}^g, z_{\text{da}})$. Following these two generation networks, a differentiable diffuse renderer $\mathcal{R}_{\text{diff}}$ takes as input $\boldsymbol{N}^g$ and $\boldsymbol{I}^g_d$ to generate a diffuse image $\boldsymbol{I}^g_{df} := \mathcal{R}_{\text{diff}}(\boldsymbol{N}^g, \boldsymbol{I}^g_{da}, L)$.

On the other end (Figure 3-bottom), given the 'real' diffuse albedo map $\boldsymbol{I}^r_{da}$ and detailed normal map $\boldsymbol{N}^r$, we introduce an encoder E_{da} to estimate a Gaussian-distributed diffuse albedo code from the real diffuse albedo map $\boldsymbol{I}^r_d$. A 'real' diffuse image is rendered via $\boldsymbol{I}^r_{df} := \mathcal{R}_{\text{diff}}(\boldsymbol{N}^r, \boldsymbol{I}^r_{da}, L)$, taken as input to the depth network $\mathcal{G}_{\text{depth}}$ to generate a coarse depth map $\boldsymbol{d}^g = \mathcal{G}_{\text{depth}}(\boldsymbol{I}^r_{df})$ that fools a coarse depth map discriminator.

We jointly train all the networks $\mathcal{G}_{\text{n}}$, $\mathcal{G}_{\text{da}}$, E_{da}, $\mathcal{G}_{\text{depth}}$ with a set of adversarial losses and cycle-consistency losses, as illustrated with the dashed arrows in Figure 3. We also simultaneously train corresponding discriminators to classify the real from the generated maps/images. More details about the training losses can be found in the supplementary (Sec. A.2). We use fixed light setting L during training, placing multiple overhead lights to light the scene. Note that the light setting can be dynamically changed at inference time, resulting in illumination control in the generated images.

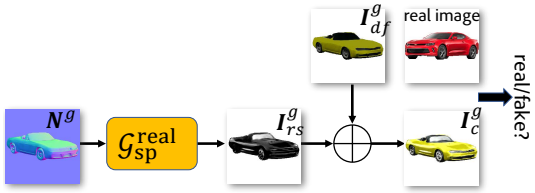

Figure 4: **Learning realistic specular maps.** The cycle-consistent adversarial network for learning to generate realistic specular maps from normal maps using adversarial and cycle consistency losses. For simplicity, other essential modules in the training cycles are omitted.

Learning realistic specular generation. To add further realism of the final image, we learn a realistic specular network $\mathcal{G}^{\text{real}}_{\text{sp}}$, which takes as input the generated detailed normal map $\boldsymbol{N}^g$, derived from the input depth, to generate a realistic specular map $\boldsymbol{I}^g_{rs}$. Blending this generated realistic specular map with the generated diffuse image $\boldsymbol{I}^g_{df}$ leads to a composite image $\boldsymbol{I}^g_c$ that fools a realistic images discriminator (see Figure 4). To enable training without paired data, we designed a cycle-consistent adversarial network for learning $\mathcal{G}^{\text{real}}_{\text{sp}}$. Note the realistic specular generator can be linked to the networks of training reflectance map generators, making the setup end-to-end trainable. Also, this realistic specular generation is only conditioned on the view-specific input $\boldsymbol{N}^g$ and $\boldsymbol{I}^g_{df}$ but remains unaffected by illumination specifications.

4 Experiments

We introduce the datasets, evaluation metrics, and compare Neural Graphics Pipeline against competing GAN-based and/or neural rendering baselines. Further details can be found in the supplementary (Sec. A.4 and Sec. A.5). We evaluate our generated images, both qualitatively and quantitatively, on publicly-available datasets. For comparisons, at test time, we use two variants of our method: (i) *NGP*: as the default option, the final image is generated by blending the diffuse rendering of $\mathcal{R}_{\text{diff}}$ under 4 base overhead lights (same setting as in training) with the realistic specular map; and (ii) *NGP-BP*: as a depleted option, where we use the full Blinn-Phong renderer $\mathcal{R}_{\text{BP}}$ under 4 base lights overhead (same setting as in training), along with randomly sampled lights, but *without* blending with the realistic specular map.

4.1. Evaluations. ***Datasets.*** Our 3D dataset consists of chairs and cars from ShapeNet [49]; as 2D datasets, we render each ShapeNet model in Blender [50] to collect example real reflectance maps for training the reflectance map generators, while we use the real-world images dataset from VON [6], which contains 2605 car images and 1963 chair images, for training the realistic specular generator.

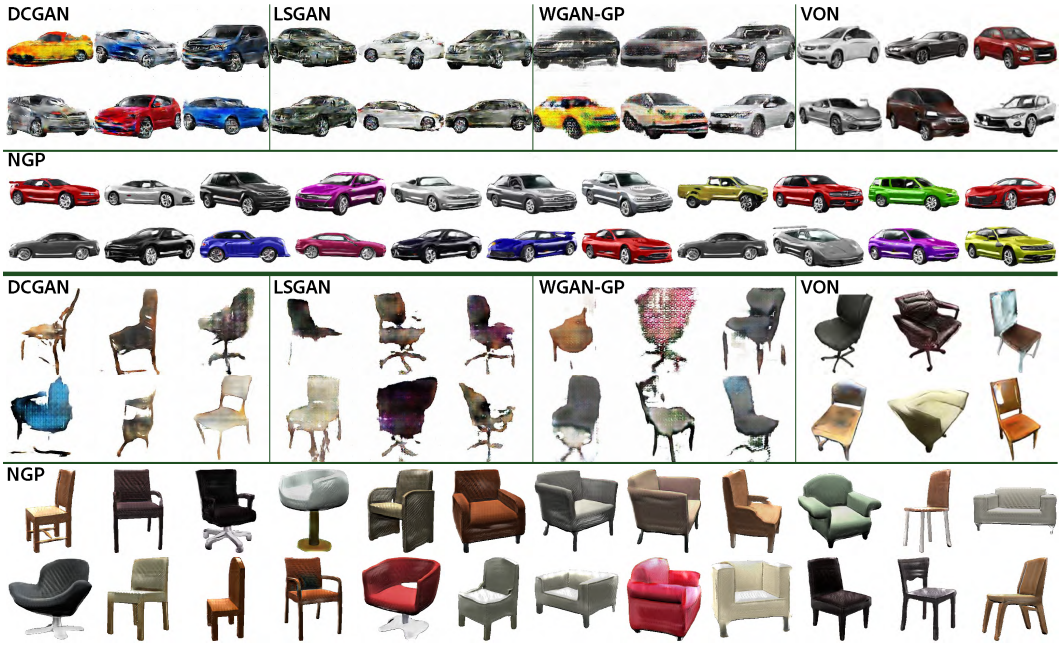

Figure 5: **Qualitative comparison with baselines.** NGP versus DCGAN [1], LSGAN [10], WGAN-GP [11], and VON [6]. All the models were trained on the same set of real-world images.

Baselines. We compare our method against the following baseline methods: DCGAN [1], LSGAN [10], WGAN-GP [11], VON [6], and SRNs [7], of which the details can be found in the supplementary. Since SRNs assumes training images with full camera parameters, we train SRNs on Blinn-Phong rendered images with varying lighting. For a fair comparison, we compare separately to SRNs with NGP-BP, reporting the FID computed against Blinn-Phong rendered images.

Metrics. Fréchet Inception Distance (FID) is an established measure comparing inception similarity score between distributions of generated and real images [12, 13]. To evaluate an image generation model, we calculate FID between the generated images set and a target real images set. Specifically, each set of images are fed to the Inception network [51] trained on ImageNet [52], then the features with length 2048 from the layer before the last fully-connected layer are used to calculate the FID. Lower FID score indicates image generation with better quality.

Implementation details. Hyperparameters, full network architectures, and rendering models for NGP and the variants are detailed in the supplementary (Sec. A.3). Training of the presented models took 5 days per class on a single Nvidia GeForce GTX 1080. A single forward pass takes around 180 ms and 1 GB of GPU memory. Note that while training on real images, we found accurately modeling perspective effects, instead of an orthogonal camera assumption, to be important.

Ablation study. Table 1 shows the ablation result of our realistic specular blending network $\mathcal{G}_{\text{sp}}^{\text{real}}$ using NGP-BP *v.s.* NGP. We refer to the supplementary (Sec. A.6) for additional ablation studies.

Results. We first compare our method against baseline methods (excluding SRNs) on the real-world images data. Our method variants consistently outperform these baselines qualitatively and quantitatively. In Table 1, both NGP and NGP-BP have the two best FID scores, outperforming other baseline methods by large margins. Qualitative comparisons on real images are presented in Figure 5. Note the realism of specular highlights, the wheels and windscreens of the cars, or the varying illumination on the chairs. The GAN variants (i.e., DCGAN, LSGAN, and WGAN-GP) suffer

Table 1: FID comparison on real images data. Note that FIDs are computed against real images data.

	DCGAN	LSGAN	WGAN-GP	VON	NGP-BP	NGP
car	130.5	171.4	123.4	83.3	**67.2**	**58.3**
chair	225.0	225.3	184.9	51.8	**47.9**	**51.2**

Table 2: SRNs comparison. Note FIDs are computed against Blinn-Phong images data.

	SRNs	NGP-BP
car	167.0	**30.0**
chair	50.3	**32.0**

Figure 6: **Comparison with SRNs.** For fair comparison, we give SRNs [7] full camera information and use the depleted NGP-BP option. Please refer to the supplementary (Sec. A.5) for details.

from lower visual quality as they seek to directly map the Gaussian samples to final images, only producing results with roughly plausible content. Among these variants, VON produces the closest results compared to NGP. Note that although our method provides control over illumination and camera, we do not see any performance degradation, but on the contrary, our method still produces slightly better visual results over VON. Interestingly, observe that by imposing inductive bias on the image formation model used in traditional rendering, NGP-BP results in superior quality results even when trained on the same dataset. We also conduct qualitative comparison with NGP-BP against SRNs, as shown in Figure 6, with quantitative numbers in Table 2.

4.2. Controllable Image Generation. The key advantage of NGP is retaining the controls available in traditional modeling-and-rendering based image generation. In the following, we demonstrate the various controls supported by our method. See Figure 1 and supplemental video.

Shape control. NGP generates images of diverse shapes with ease via simply changing the shape code z_s. Additionally, the user can directly edit the coarse geometry, as shown in the video.

Camera control. We also support full camera view control for the final generated image while keeping all other factors. Figure 8 illustrates the effect of changing the camera view for generating different final images. Note that earlier works including VON [6], SRNs [7], and Hologan [4] also support various levels of camera control.

Illumination control. Our method models detailed normal maps in the reflectance property maps generation stage, so that additional lights can be added on top with explicit control of the illumination (see Figures 2, 8). We call this more versatile option NGP-plus (see details in Sec. A.5). Such level of control (i.e., explicit light count, position, and intensity) is not supported by VON [6] and Hologan [4]. Figure 8 shows the effect of generating various images with different additional light settings.

Appearance control. The overall appearance, particularly the colorization, of the content in the generated images can be easily changed by providing an exemplar image as guidance, leading to controllable and various appearance in generated images (see Figure 7). Further, this allows the user to simply edit the diffuse albedo, akin to traditional control, using existing imaging tools, and render the final image using NGP, thus benefiting from the appearance disentanglement.

4.3. Limitations. While we presented an improved image generation pipeline, the output quality is nevertheless below the current limits of traditional computer graphics (e.g., using Blender). While we acknowledge this shortcoming, we believe this will rapidly change, as we have witnessed in the context of GANs in general or neural rendering in particular. One axis of improvement, will be targeting larger image sizes (e.g., 1024×1024 instead of current 256×256), possibly using a progressive GAN setup. Another limitation is that our models are class-specific, thus, currently separate networks need to be trained for new shape categories. However, conceptually, being unsupervised, we believe that Neural Graphics Pipeline can be used across many classes, as long as

Figure 7: Appearance control via exemplar diffuse albedo images (top rows). Note that the specular highlights on the surface are preserved even under changes to the color of the cars/chairs.

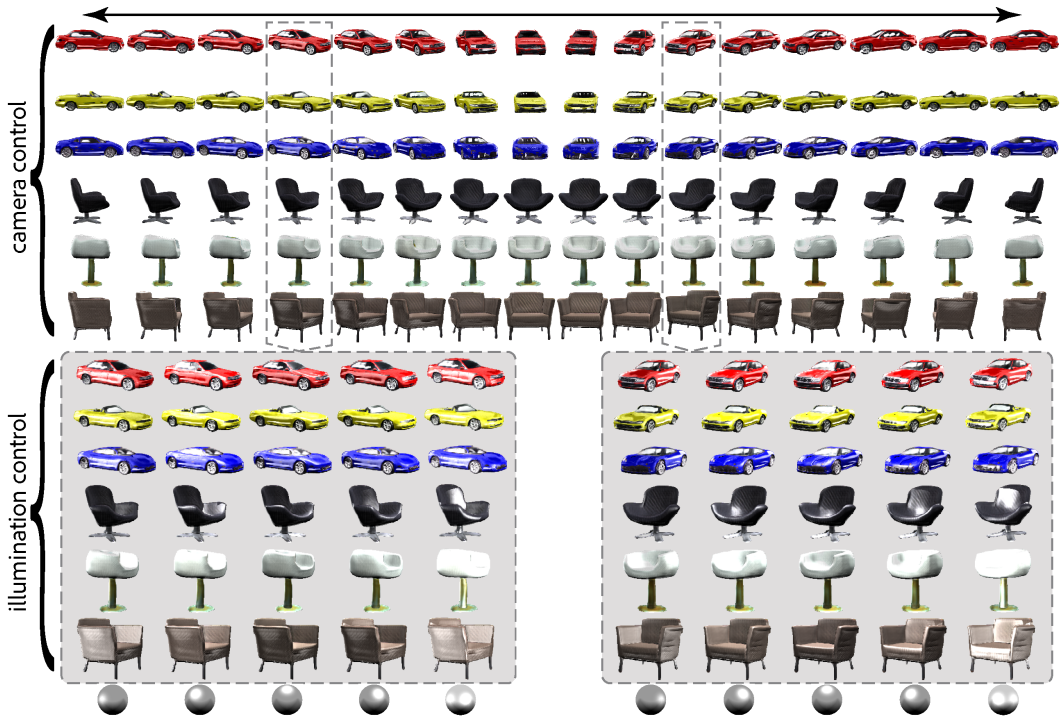

Figure 8: **Camera and illumination control.** (Top) Camera control with object shape and appearance held fixed along rows. For selected camera views (marked at the top), we show (at the bottom) the corresponding generations under changing illumination (intensity, location, number of lights) as shown on the mirrored ball. Note the granularity of camera and illumination control enabled by ours.

we can get sufficient data volumes for training, and stronger networks with larger capacities, and can be boosted by handling more advanced rendering models.

5 Conclusions and Future Work

We have presented a novel graphics pipeline that combines the strength of neural networks with the effective direct control offered in traditional graphics pipeline. We enable this by designing neural blocks that generate coarse 3D geometry and produce interpretable 2D feature layers, that can then be directly handled by a fixed rendering function to produce a final image. It is important to emphasize that our training is completely unsupervised and no attributes like texture or reflectance indices are associated with the images. This allows using both real and synthetic images.

As we have presented, the unsupervised training of a parametric translation is a key technical contribution. It involves two carefully designed architectures with cycle consistency losses that make up for the lack of supervision in the form of any paired data. While our current implementation supports four interpertable maps, the design is scalable and can include additional maps, which in turn may unlock more control handles using advanced rendering setups.

Our Neural Graphics Pipeline, and neural rendering in general, questions *when do we really need 3D models?* In computer graphics, the production or the fabrication of a physical 3D model is of less important as the ultimate goal of the pipeline is to 'merely' produce images. Then, it is questionable whether the 3D representation is needed at all, or some abstract or latent representation could well instead be used. In our current work, we explored this avenue and reduced the requirement from detailed 3D models and/or explicit material assignment. In the future, we would like to continue and search for new means to further reduce or totally avoid the use of explicit 3D, without losing any control over the quality or manipulation of the generated image. As the quality of neural image generation continue to rapidly improve, we believe that this work is an important step forward towards a fully neural workflow, without sacrificing users' ability to control the underlying scene attributes.

Broader Impact

Our work benefits the GAN-based production of images with effective direct controls, as in traditional graphics pipeline, with users being able to control the image generation process without elaborately authoring the fully detailed 3D scenes. We believe the main impact of our work is a step forward towards the combination of the neural generative models and the traditional graphics imaging formulations, incorporating the merits of each.

Any image generation work that learns from unlabelled real-world images data, particularly from object-specific (e.g., face) images data, runs the risk of data hacking or offensive content reflective of the training data. Our work, which learns a image generation model from real-world images, may not be an exception. Nevertheless, our work requires to collect the underlying reflectance property maps of the content in the image for learning, which makes it, for the time being, unpractical for the growing identity hacking. In our implementation, the risk is also moderated by only demonstrating our work on object-centric image generation.

Overall, we believe that the impact of our work is mainly to offer traditional control over neural rendering. In general, it broadens the understanding of neural rendering and offers means to train it without supervision, which then allows training it with both real and synthetic data. We do not believe that our work has any ethical or social implications beyond those stated above.

References

[1] Radford Alec, Metz Luke, and Chintala Soumith. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2016.

[2] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2437–2446, 2019.

[3] Thu H Nguyen-Phuoc, Chuan Li, Stephen Balaban, and Yongliang Yang. Rendernet: A deep convolutional network for differentiable rendering from 3d shapes. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 7891–7901, 2018.

[4] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *International Conference on Computer Vision (ICCV)*, Nov 2019.

[5] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6878–6887, 2019.

[6] Jun-Yan Zhu, Zhoutong Zhang, Chengkai Zhang, Jiajun Wu, Antonio Torralba, Josh Tenenbaum, and Bill Freeman. Visual object networks: image generation with disentangled 3d representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 118–129, 2018.

[7] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1119–1130, 2019.

[8] Kara-Ali Aliev, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. *arXiv preprint arXiv:1906.08240*, 2019.

[9] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[10] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2017.

[11] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning (ICML)*, pages 214–223, 2017.

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6626–6637, 2017.

[13] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 700–709, 2018.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2672–2680, 2014.

[15] Jianwei Yang, Anitha Kannan, Dhruv Batra, and Devi Parikh. Lr-gan: Layered recursive generative adversarial networks for image generation. *International Conference on Learning Representations (ICLR)*, 2017.

[16] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations (ICLR)*, 2019.

[17] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8798–8807, 2018.

[18] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[19] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[20] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *International Conference on Machine Learning (ICML)*, pages 2642–2651, 2017.

[21] Thu Nguyen-Phuoc, Christian Richardt, Long Mai, Yong-Liang Yang, and Niloy Mitra. Blockgan: Learning 3d object-aware scene representations from unlabelled images. *arXiv preprint arXiv:2002.08988*, 2020.

[22] Martin Engelcke, Adam R Kosiorek, Oiwi Parker Jones, and Ingmar Posner. Genesis: Generative scene inference and sampling with object-centric latent representations. *International Conference on Learning Representations (ICLR)*, 2020.

[23] Sjoerd van Steenkiste, Karol Kurach, and Sylvain Gelly. A case for object compositionality in deep generative models of images. *CoRR*, abs/1810.10340, 2018.

[24] Takuhiro Kaneko, Kaoru Hiramatsu, and Kunio Kashino. Generative adversarial image synthesis with decision tree latent controller. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[25] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016.

[26] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2337–2346, 2019.

[27] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 82–90, 2016.

[28] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5939–5948, 2019.

[29] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[30] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. GRASS: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics (TOG)*, 36(4):Article 52, 2017.

[31] Chenyang Zhu, Kai Xu, Siddhartha Chaudhuri, Renjiao Yi, and Hao Zhang. SCORES: Shape composition with recursive substructure priors. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 37(6), 2018.

[32] Lin Gao, Jie Yang, Tong Wu, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. SDM-NET: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 38(6), 2019.

[33] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas Guibas. Structurenet: Hierarchical graph networks for 3d shape generation. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 38(6):Article 242, 2019.

[34] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *International Conference on Computer Vision (ICCV)*, pages 4541–4550, 2019.

[35] Rui Ma, Akshay Gadi Patil, Matt Fisher, Manyi Li, Soren Pirk, Binh-Son Hua, Sai-Kit Yeung, Xin Tong, Leonidas J. Guibas, and Hao Zhang. Language-driven synthesis of 3d scenes using scene databases. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 37(6), 2018.

[36] Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X Chang, and Daniel Ritchie. Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Transactions on Graphics (SIGGRAPH)*, 38(4):1–15, 2019.

[37] Daniel Ritchie, Kai Wang, and Yu-an Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6182–6190, 2019.

[38] Yang Yang, Shi Jin, Ruiyang Liu, Sing Bing Kang, and Jingyi Yu. Automatic 3d indoor scene modeling from single panorama. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3926–3934, 2018.

[39] Tom Kelly, Paul Guerrero, Anthony Steed, Peter Wonka, and Niloy J. Mitra. Frankengan: Guided detail synthesis for building mass models using style-synchonized gans. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 37(6):1:1–1:14, 2018.

[40] Yiwei Zhao, Han Liu, Igor Borovikov, Ahmad Beirami, Maziar Sanjabi, and Kazi Zaman. Multi-theme generative adversarial terrain amplification. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 38(6):200, 2019.

[41] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (SIGGRAPH)*, 38(4):1–12, 2019.

[42] Aliaksandra Shysheya, Egor Zakharov, Kara-Ali Aliev, Renat Bashirov, Egor Burkov, Karim Iskakov, Aleksei Ivakhnenko, Yury Malkov, Igor Pasechnik, Dmitry Ulyanov, et al. Textured neural avatars. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2387–2397, 2019.

[43] Zaiwei Zhang, Zhenpei Yang, Chongyang Ma, Linjie Luo, Alexander Huth, Etienne Vouga, and Qixing Huang. Deep generative modeling for scene synthesis via hybrid representations. *arXiv preprint arXiv:1808.02084*, 2018.

[44] Sai Bi, Kalyan Sunkavalli, Federico Perazzi, Eli Shechtman, Vladimir G Kim, and Ravi Ramamoorthi. Deep cg2real: Synthetic-to-real translation via image disentanglement. In *International Conference on Computer Vision (ICCV)*, pages 2730–2739, 2019.

[45] Kyle Olszewski, Sergey Tulyakov, Oliver Woodford, Hao Li, and Linjie Luo. Transformable bottleneck networks. *International Conference on Computer Vision (ICCV)*, Nov 2019.

[46] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018.

[47] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Latent-space gans for 3D point clouds. In *International Conference on Machine Learning (ICML), Workshop on Implicit Models*, 2017.

[48] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[49] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[50] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2020.

[51] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[52] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[53] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *International Conference on Computer Vision (ICCV)*, pages 2223–2232, 2017.

[54] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *European Conference on Computer Vision (ECCV)*, pages 172–189, 2018.

[55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[56] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

A.1 Blinn-Phong rendering function

For rendering the final images, we assume that the reflectance of the content in the scene under camera view v is characterized by a set of property maps: a surface normal map $\boldsymbol{N}$, a diffuse albedo map $\boldsymbol{I}_d$, a specular albedo map $\boldsymbol{I}_s$, and a monochrome specular roughness map $\boldsymbol{\alpha}$. We use a classical rendering model - *Blinn-Phong Reflection Model* - as our rendering equation, which, for a given light $\boldsymbol{L}$, computes intensity as:

$$\begin{aligned} \boldsymbol{I} &= k_d(\boldsymbol{N}\cdot\boldsymbol{L})\boldsymbol{I}_d + k_s(\boldsymbol{N}\cdot\boldsymbol{H})^{\boldsymbol{\alpha}}\boldsymbol{I}_s \\ \boldsymbol{H} &= \frac{\boldsymbol{L}+\boldsymbol{V}}{\|\boldsymbol{L}+\boldsymbol{V}\|} \end{aligned} \tag{A.1}$$

wherek_d and k_s are diffuse reflection constant and specular reflection constant, respectively. $\boldsymbol{V}$ is the direction to the viewer, and hence is set to the view direction of v for approximation.

A.2 Training losses for reflectance maps generation

We train the 2D networks for generating reflectance maps with a set of adversarial losses and cycle consistency losses. Each loss described in the following corresponds to a dashed arrow in the architecture figure in the main paper.

Adversarial losses For translating depth images to final composition images, we use the following adversarial loss for the detailed normal map generation:

$$L_{\text{n}}^{GAN} = \mathbb{E}_{\boldsymbol{N}^r}[\log D_{\text{n}}(\boldsymbol{N}^r)] + \mathbb{E}_{\boldsymbol{d}^r}[\log(1 - D_{\text{n}}(\mathcal{G}_{\text{n}}(\boldsymbol{d}^r))], \tag{A.2}$$

where D_{n} learns to classify the real and generated normal maps. For the adversarial loss on the diffuse albedo maps generation:

$$L_{\text{da}}^{GAN} = \mathbb{E}_{\boldsymbol{I}_d^r}[\log D_{\text{da}}(\boldsymbol{I}_d^r)] + \mathbb{E}_{(\boldsymbol{d}^r, z_{\text{da}})}[\log(1 - D_{\text{da}}(\mathcal{G}_{\text{da}}(\boldsymbol{d}_{\text{noc}}^r, \mathcal{G}_{\text{n}}(\boldsymbol{d}^{\boldsymbol{r}}), z_{\text{da}})))], \tag{A.3}$$

where $\boldsymbol{d}_{\text{noc}}^r = noc(\boldsymbol{d}^r)$ and D_{da} learns to classify the real and generated diffuse albedo maps. We also apply the adversarial loss on the diffuse images:

$$L_{\text{df}}^{GAN} = \mathbb{E}_{\boldsymbol{I}_{df}^r}\left[\log D_{\text{df}}(\boldsymbol{I}_{df}^r)\right] + \mathbb{E}_{\boldsymbol{d}^r}[\log(1 - D_{\text{df}}(\mathcal{R}_{\text{diff}}(\mathcal{G}_{\text{n}}(\boldsymbol{d}^{\boldsymbol{r}}), \mathcal{G}_{\text{da}}(\boldsymbol{d}_{\text{noc}}^r, \mathcal{G}_{\text{n}}(\boldsymbol{d}^{\boldsymbol{r}}), z_{\text{da}}), L)))], \tag{A.4}$$

where L is the light setting, $\boldsymbol{I}^r_{df}$ is the real diffuse image produced from real diffuse albedo and normal maps, and D_{df} learns to classify the real and generated diffuse images.

For the translation from diffuse images to depth images, we use the following adversarial loss:

$$L^{GAN}_{\text{depth}} = \mathbb{E}_{\boldsymbol{d}^r}[\log D_{\text{depth}}(\boldsymbol{d}^r)] + \mathbb{E}_{\boldsymbol{I}^r_{df}}\left[\log(1 - D_{\text{depth}}(\mathcal{G}_{\text{depth}}(\boldsymbol{I}^r_{df})))\right], \tag{A.5}$$

where D_{depth} learns to classify the real and generated depth images. Furthermore, as we observed that the task of classifying the real depth images and generated ones is rather easier for D_{depth}, we also add the adversarial loss on the NOC image derived from the depth image to balance the network training:

$$L^{GAN}_{\text{noc}} = \mathbb{E}_{\boldsymbol{d}^r}[\log D_{\text{noc}}(noc(\boldsymbol{d}^r))] + \mathbb{E}_{\boldsymbol{I}^r_{df}}\left[\log(1 - D_{\text{noc}}(noc(\mathcal{G}_{\text{depth}}(\boldsymbol{I}^r_{df}))))\right], \tag{A.6}$$

where D_{noc} learns to classify the real and generated NOC images.

Cycle-consistency losses We further add the following cycle consistency losses to enforce the bijective relationship between each two domains.

Cycle-consistency loss on the depth map:

$$L^{cyc}_{\text{depth}} = \mathbb{E}_{(\boldsymbol{d}^r, z_{\text{da}})}\left[\left\|\mathcal{G}_{\text{depth}}(\mathcal{R}_{\text{diff}}(\mathcal{G}_{\text{n}}(\boldsymbol{d}^r), \mathcal{G}_{\text{da}}(\boldsymbol{d}^r_{\text{noc}}, \mathcal{G}_{\text{n}}(\boldsymbol{d}^r), z_{\text{da}}), L)) - \boldsymbol{d}^r\right\|_1\right]; \tag{A.7}$$

Cycle-consistency loss on the NOC map:

$$L^{cyc}_{\text{noc}} = \mathbb{E}_{(\boldsymbol{d}^r, z_{\text{da}})}\left[\left\|noc(\mathcal{G}_{\text{depth}}(\mathcal{R}_{\text{diff}}(\mathcal{G}_{\text{n}}(\boldsymbol{d}^r), \mathcal{G}_{\text{da}}(\boldsymbol{d}^r_{\text{noc}}, \mathcal{G}_{\text{n}}(\boldsymbol{d}^r), z_{\text{da}}), L))) - \boldsymbol{d}^r_{\text{noc}}\right\|_1\right], \tag{A.8}$$

where $\boldsymbol{d}^r_{\text{noc}} = noc(\boldsymbol{d}^r)$; Cycle-consistency loss on the normal map:

$$L^{cyc}_{\text{n}} = \mathbb{E}_{(\boldsymbol{N}^r, \boldsymbol{I}^r_d)}\left[\left\|\mathcal{G}_{\text{n}}(\mathcal{G}_{\text{depth}}(\mathcal{R}_{\text{diff}}(\boldsymbol{N}^r, \boldsymbol{I}^r_d, L))) - \boldsymbol{N}^r\right\|_1\right]; \tag{A.9}$$

And cycle-consistency loss on the diffuse albedo map:

$$L^{cyc}_{\text{da}} = \mathbb{E}_{(\boldsymbol{N}^r, \boldsymbol{I}^r_d, \boldsymbol{I}^r_{df})}\left[\left\|\mathcal{G}_{\text{da}}(\mathcal{G}_{\text{depth}}(\boldsymbol{I}^r_{df}), \boldsymbol{N}^r, E_{\text{da}}(\boldsymbol{I}^r_d)) - \boldsymbol{I}^r_d\right\|_1\right]; \tag{A.10}$$

Cycle-consistency loss for the diffuse image:

$$L^{cyc}_{\text{df}} = \mathbb{E}_{(\boldsymbol{N}^r, \boldsymbol{I}^r_d, \boldsymbol{I}^r_{df})}\left[\left\|\mathcal{R}_{\text{diff}}(\mathcal{G}_{\text{n}}(\mathcal{G}_{\text{depth}}(\boldsymbol{I}^r_{df})), \mathcal{G}_{\text{da}}(\mathcal{G}_{\text{depth}}(\boldsymbol{I}^r_{df}), \boldsymbol{N}^r, E_{\text{da}}(\boldsymbol{I}^r_d)), L) - \boldsymbol{I}^r_{df}\right\|_1\right], \tag{A.11}$$

where $\boldsymbol{I}^r_{df} = \mathcal{R}_{\text{diff}}(\boldsymbol{N}^r, \boldsymbol{I}^r_d, L)$.

In addition, similar to the latent space reconstruction in other unconditional GANs and image-to-image translation works, we also introduce a latent space cycle-consistency loss to encourage $\mathcal{G}_{\text{da}}$ to use the diffuse albedo code z_{da}:

$$L^{cyc}_{z_{\text{da}}} = \mathbb{E}_{(\boldsymbol{d}^r, z_{\text{da}})}[\|E_{\text{da}}(\mathcal{G}_{\text{da}}(\boldsymbol{d}^r_{noc}, \mathcal{G}_{\text{n}}(\boldsymbol{d}^r), z_{\text{da}})) - z_{\text{da}}\|_1]. \tag{A.12}$$

At last, to enable sampling at test time, we force $E_{\text{da}}(\boldsymbol{I}^r_d)$ to be close to the standard Gaussian distribution, by adding a Kullback-Leibler (KL) loss on the z_{da} latent space:

$$L_{\text{KL}} = \mathbb{E}_{\boldsymbol{I}^r_d}[\mathcal{D}_{KL}(E_{\text{da}}(\boldsymbol{I}^r_d)\|\mathcal{N}(0, \mathcal{I}))], \tag{A.13}$$

where $\mathcal{D}_{KL}(p\|q) = -\int_z p(z)\log\frac{p(z)}{q(z)}dz$.

Finally, we write the final 2D modeling loss as:

$$\begin{aligned} L^{\text{2D}}_{\text{modeling}} = {} & L^{GAN}_{\text{n}} + L^{GAN}_{\text{da}} + L^{GAN}_{\text{df}} + L^{GAN}_{\text{depth}} + L^{GAN}_{\text{noc}} \\ & + \lambda^{cyc}_{\text{n}} L^{cyc}_{\text{n}} + \lambda^{cyc}_{\text{da}} L^{cyc}_{\text{da}} + \lambda^{cyc}_{\text{df}} L^{cyc}_{\text{df}} \\ & + \lambda^{cyc}_{\text{depth}} L^{cyc}_{\text{depth}} + \lambda^{cyc}_{\text{noc}} L^{cyc}_{\text{noc}} + \lambda^{cyc}_{z_{\text{da}}} L^{cyc}_{z_{\text{da}}} + \lambda_{\text{KL}} L_{\text{KL}}, \end{aligned} \tag{A.14}$$

where λ^{cyc}_{n}, $\lambda^{cyc}_{\text{da}}$, $\lambda^{cyc}_{\text{df}}$, $\lambda^{cyc}_{\text{depth}}$, $\lambda^{cyc}_{\text{noc}}$, $\lambda^{cyc}_{z_{\text{da}}}$ and λ_{KL} control the importance of each cycle consistency loss.

A.3 Implementation details

3D shape network For the coarse shape synthesis network, we adopt the IM-GAN architecture from [28]. Both generator and the discriminator are constructed by two hidden fully-connected layers, and the Wasserstein GAN loss with gradient penalty is adopted to train the latent-GAN.

2D detailing networks We use a perspective camera with a focal length of 50mm (35 film equivalent). The 2D networks takes as input depth images of 256×256, which is also the size for all reflectance maps. For 2D maps generation networks, we use the ResNet encoder-decoder [53, 54] for all map generators. In addition, we concatenate the diffuse code z_{da} to all intermediate layers in the encoder of $\mathcal{G}_{\text{da}}$ [6], the generated detailed normal map N^g is fused to the first layer of the encoder of $\mathcal{G}_{\text{da}}$ by concatenation. The ResNet encoder [55] is used for constructing E_{da}. We use mid (70×70) and large (140×140) receptive field size (RFS) for all discriminators (except the diffuse albedo discriminator), as the generation of these maps relies on the mid-level and global-structure features extracted by the corresponding discriminators; we use small (34×34) RFS for the diffuse albedo discriminator, as the generation of the diffuse albedo needs only low-level features extracted by the corresponding discriminator, such as *local* smoothness, purity, repetitive pattern and etc. of the albedo color, paying less attention to the global structure of the generated diffuse albedo. Finally, we use the least square objective as in LS-GAN [10] for stabilizing the training.

Training details The 3D generation network is trained as described in the original IM-NET paper. The z_{s} is sampled from the standard Gaussian distribution $\mathcal{N}(0, \mathcal{I})$, with the code dimension $|z_{\text{s}}| = 200$. The generated implicit fields are converted to meshes by using $128 \times 128 \times 128$ grid samplings and Marching Cubes. The diffuse code z_{da} is also sampled from the standard Gaussian distribution $\mathcal{N}(0, \mathcal{I})$, with the code length $|z_{\text{da}}| = 8$. We set the hyperparameters in Eq. A.14 as, $\lambda^{\text{cyc}}_{\text{depth}} = \lambda^{cyc}_{\text{noc}} = 10$, $\lambda^{cyc}_{\text{n}} = \lambda^{cyc}_{\text{da}} = \lambda^{cyc}_{\text{df}} = 25$, $\lambda^{cyc}_{z_{\text{da}}} = 1$, $\lambda_{\text{KL}} = 0.001$. We use Adam optimizer [56] with a learning rate of 0.0001 for training all 2D networks. We first train the reflectance maps generation networks for 300,000 samples, and then train the realistic specular generation networks for 200,000 samples, at last fine-tune the whole 2D setup by joint training. The diffuse reflectance constant k_d in Equation A.1 to 0.6 for cars and 0.8 for chairs. At the inference time, the specular reflection constant k_s in Equation A.1 is set to 0.4 for cars and 0.2 for chairs, if applicable.

A.4 Details of datasets

Real reflectance property map sets For training reflectance property map generators, we render each model in Blender to collect the real reflectance property maps. Each model is fit into a unit sphere placed at the origin. The camera view is randomly sampled from the camera view distribution described next. For the dataset of real reflectance property maps, we random sample camera views and render the models in Blender, obtaining around 10k sets of reflectance property maps for car category and around 40k sets of reflectance property maps for chair category.

Camera view distribution We assume the camera is at a fixed distance of 2m to the origin and use a focal length of 50mm (35mm film equivalent). The camera location is restricted on a sphere, which can be parameterized as ($\rho = 2$, θ, ϕ), where θ is the counter-clockwise vertical angle from the object face-direction base and ϕ is the horizontal angle from the object face direction base. By default, we set the range of θ to be $[0°, 20°]$ and the range of ϕ to be $[-90°, 90°]$. In addition, we constrain the camera to look at the origin and disable camera in-plane rotation.

Real images For training the realistic specular generator, we use the real-world images dataset from VON [6], which contains 2605 car images and 1963 chair images. The images are randomly flipped during the training for data augmentation.

A.5 Details of baseline methods and NGP variants

In the following, we describe the details of the baseline methods and the variants of our method appeared in the paper:

(i) DCGAN [1] proposed specific generator and discriminator architectures that significantly improve the training of generative adversarial networks. We use DCGAN with the standard cross-entropy loss.

(ii) LSGAN [10] adopted least square loss for stabilizing the GAN training. We use the same DCGAN generator and discriminator architectures for LSGAN.

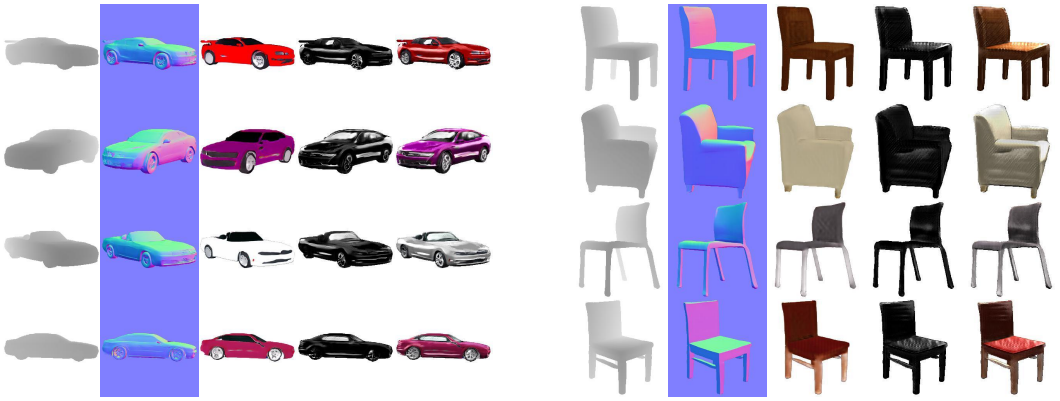

Figure A.1: More results with generated intermediate maps. From left to right: input coarse depth map, generated detailed normal map, generated detailed diffuse albedo map, generated realistic specular map, and the final generated image of our method. The generated specular albedo map and roughness map by constant function generators are not shown here.

(iii) WGAN-GP [11] adopted Wasserstein metric and gradient penalty in training. We also use the same DCGAN generator and discriminator architectures for WGAN-GP. In addition, we replace the default BatchNorm by InstanceNorm in the discriminator, and train the discriminator 5 times per generator iteration.

(iv) VON [6] also generates 3D rough shapes first but instead trains a network to add texture from a specific view to generate images. The VON results are obtained by the released models from the authors.

(v) SRNs [7] formulates the image formation as a neural, 3D-aware rendering algorithm. SRNs assume having images with full camera parameters as training data, thus it can only be trained on composite images obtained by rendering the ShapeNet models using Blinn-Phong renderer. After trained, we make SRNs a generative model for image generation task by randomly *pick* scene codes generated from the training and randomly sample camera viewpoints, similarly to the novel view synthesis application as described in the original paper.

(vi) NGP, as the default option, the final image is generated by blending the diffuse rendering of $\mathcal{R}_{\text{diff}}$ under 4 base overhead lights (same setting as in training) with the realistic specular map. Note that only $\mathcal{R}_{\text{diff}}$ is used to light the scene under the base lights, thus these base lights only result in diffuse reflection but no specular highlights in the final image.

(vii) NGP-BP, as a depleted option, where we use the full Blinn-Phong renderer $\mathcal{R}_{\text{BP}}$ under 4 base lights overhead (same setting as in training), along with randomly sampled lights, but *without* blending with the realistic specular map.

(viii) NGP-plus, as a more versatile option that combines NGP and NGP-BP for illumination control of additional lights. The output image of NGP is first formed, on top of which the diffuse reflection and specular reflection yielded by the additional lights via $\mathcal{R}_{\text{BP}}$ are added for producing the final image.

A.6 Further Evaluations

More Results with intermediate Maps In Figure A.1, we present more qualitative results of generated images using our method, along with the intermediate maps used to composite final images.

Evaluation on NGP variants In table A.1, we show the FID scores of the three NGP variants. We can see that, in general, all the three NGP variants consistently outperforms the other methods. NGP-plus even yields slightly better results than NGP with additional illumination control. Interestingly, the NGP-BP produces the best results on chairs even with a biased traditional rendering model (Blinn-Phong model).

Table A.1: FID comparison on real images data. Note that FIDs are computed against real images data.

	DCGAN	LSGAN	WGAN-GP	VON	NGP-BP	NGP	NGP-plus
car	130.5	171.4	123.4	83.3	**67.2**	**58.3**	**54.8**
chair	225.0	225.3	184.9	51.8	**47.9**	**51.2**	**50.3**

Ablation study In addition, based on the default option NGP, we also conduct ablation studies to show the importance of the detailed normal map generator, the diffuse albedo map generator and the realistic specular generator in generating the final images. The quantitative results are presented in Table A.2, where:

(i) NGP-w/o-$\mathcal{G}_{\text{n}}$ disables the detailed normal map generator in NGP, and uses the coarse normal map derived from the input coarse shape for the final image generation.

(ii) NGP-w/o-$\mathcal{G}_{\text{da}}$ disables the diffuse albedo generator in NGP, and uses a white diffuse albedo map for the final image generation.

(iii) NGP-w/o-$\mathcal{G}_{\text{sp}}^{\text{real}}$ disables the realistic specular generator in NGP, such that the final image is produced without blending with the realistic specular map.

Table A.2: Ablation study shows the importance of each generator in generating the final images.

	NGP	NGP-w/o-$\mathcal{G}_{\text{n}}$	NGP-w/o-$\mathcal{G}_{\text{da}}$	NGP-w/o-$\mathcal{G}_{\text{sp}}^{\text{real}}$
car	**58.3**	64.6	114.1	74.8
chair	**51.2**	55.7	71.3	52.9

A.7 Video results

Please see the supplementary video for demonstration of camera/illumination control supported by NGP. Note that our generators, being view-specific, can lead to small changes across camera variations.