Meta 3D AssetGen: Text-to-Mesh Generation with High-Quality Geometry, Texture, and PBR Materials

Yawar Siddiqui[†] **Tom Monnier*** **Filippos Kokkinos*** **Mahendra Kariya**
Yanir Kleiman **Emilien Garreau** **Oran Gafni** **Natalia Neverova**
Andrea Vedaldi **Roman Shapovalov*** **David Novotny***
GenAI, Meta [†]TU Munich; intern with Meta *core technical contributors

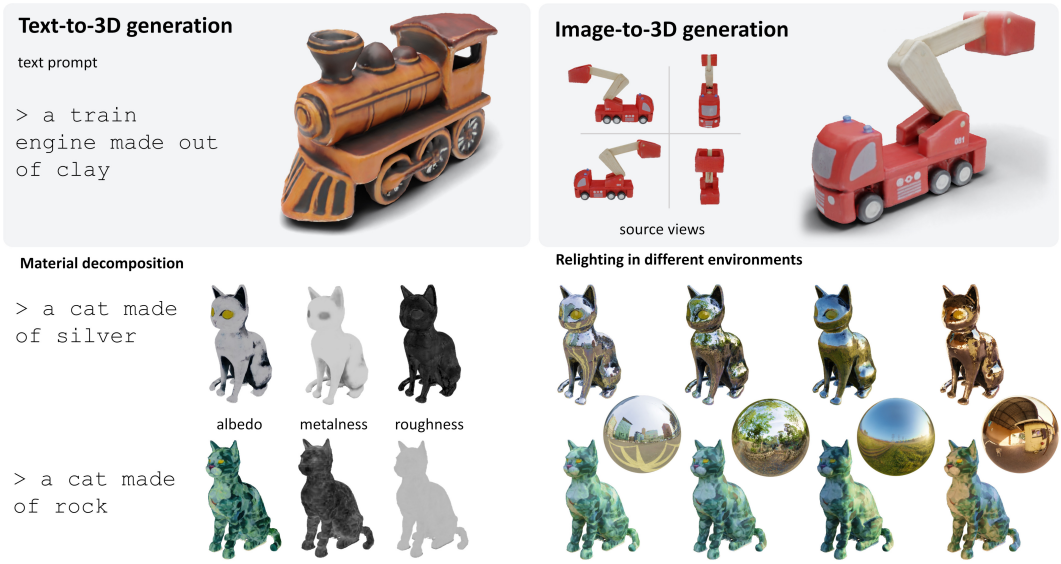


Figure 1: We present **Meta 3D AssetGen**, a novel text- or image-conditioned generator of 3D meshes with physically-based rendering materials (top). Meta 3D AssetGen produces meshes with detailed geometry and high-quality textures, and decomposes materials into albedo, metalness, and roughness (bottom left), which allows to realistically relight objects in new environments (bottom right).

Abstract

We present Meta 3D AssetGen (AssetGen), a significant advancement in text-to-3D generation which produces faithful, high-quality meshes with texture and material control. Compared to works that bake shading in the 3D object's appearance, AssetGen outputs physically-based rendering (PBR) materials, supporting realistic relighting. AssetGen generates first several views of the object with separate shaded and albedo appearance channels, and then reconstructs colours, metalness and roughness in 3D, using a deferred shading loss for efficient supervision. It also uses a sign-distance function to represent 3D shape more reliably and introduces a corresponding loss for direct shape supervision. This is implemented using fused kernels for high memory efficiency. After mesh extraction, a texture refinement transformer operating in UV space significantly improves sharpness and details. AssetGen achieves 17% improvement in Chamfer Distance and 40% in LPIPS over the best concurrent work for few-view reconstruction, and a human preference of 72% over the best industry competitors of comparable speed, including those that support PBR. Project page with generated assets: `https://assetgen.github.io`

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-0303

1 Introduction

Generating 3D objects from text or image prompts has enormous potential for 3D graphics, with applications in animation, gaming and virtual reality. However, while image and video generators have improved dramatically [68, 44, 55, 42, 97, 103], 3D generators are not ready yet for professional use. In fact, 3D generators are often slow and produce artifacts in the generated 3D meshes and textures. Many 3D generators, furthermore, "bake" appearance as albedo, ignoring how materials respond to variable environmental illumination. This results in visually unattractive outputs, especially for reflective materials, which look out of place when put in novel environments.

In this paper, we introduce *Meta 3D AssetGen*, a significant step-up in text-conditioned 3D generation. AssetGen generates assets in under 30 seconds while outperforming prior methods of comparable speed in faithfulness, in quality of the generated 3D meshes and, especially, in quality and control of materials, by supporting *Physically-Based Rendering* (PBR) [87]. The model generates albedo, metalness, and roughness so that rendered scenes can accurately reflect environmental illumination. In addition, we focus on meshes as the output representation due to their prevalence in applications and compatibility with PBR.

AssetGen uses the two-stage design epitomized by [42]. The first stage *stochastically* generates four images of the object from four canonical viewpoints, and the second stage *deterministically* reconstructs the 3D shape, appearance and materials of the object from these views (Fig. 1). The two-stage approach is faster and more robust than SDS-based techniques that perform test-time optimization [68] and, so far, produces better results than single-stage 3D generators [37, 61, 94, 79].

The first question we ask is how this design should be extended to support PBR. We show that it is difficult for the image-to-3D stage to predict PBR channels from an image as this problem is ambiguous and the model is deterministic. However, we also show that it is difficult offload PBR prediction to the text-to-image model; while this is stochastic, which handles ambiguity, the PBR channels are statistically different from the natural images used for pre-training, which makes fine-tuning difficult. Our solution is to give the text-to-image model the simpler task of outputting shaded appearance and albedo only, and task the image-to-3D stage with inferring the PBR channels from these. This reduces the statistical gap for the text-to-image model and still removes most of the ambiguity for the image-to-3D model.

We also note that the quality of 3D shapes and meshes is crucial for PBR modelling. Hence, the second question we study is how to improve 3D quality. We do so by learning a reconstruction network, *MetaILRM*, which outputs directly a signed-distance field (SDF). SDFs are better than opacity fields for meshing, as the zero level set of an SDF traces the object's surface more reliably. Furthermore, the SDF can be directly supervised using ground-truth depth maps, which is not immediately possible for opacity. The crucial contribution here is to add SDF support, including the VolSDF [108] formulation for differentiable rendering, to the memory-efficient Lightplane kernels [6]. In this way, we can use the stronger SDF representation together with larger batches and photometric loss supervision on high-resolution renders, improving both shapes and textures.

Finally, we note that much of the quality of the final asset depends on texture quality. MetaILRM's textures can still be slightly blurrier than the input image due to the limited resolution of the volumetric representation. The third question we investigate is how to maximize the texture quality. To this end, we introduce a new texture refiner network which upgrades the extracted albedo and materials by fusing information extracted from the original views, resolving possible conflicts between them.

We demonstrate the effectiveness of AssetGen on the image-to-3D and text-to-3D tasks. For image-to-3D, we attain state-of-the-art performances among existing few-view mesh-reconstruction methods when measuring the accuracy of the recover shaded and PBR texture maps. For text-to-3D, we conduct extensive user studies to compare the best methods from academia and industry that have comparable inference time, and outperform them in terms of visual quality and text alignment.

2 Related Work

Text-to-3D. Inspired by text-to-image models, early text-to-3D approaches [39, 31, 64, 34, 26, 110] train 3D diffusion models on datasets of captioned 3D assets. Yet, the limited size and diversity of 3D data prevents generalization to open-vocabulary prompts. Recent works thus pivoted into basing

such generators on text-to-image models that are trained on billions of captioned images. Among these, works like [75, 56] finetune 2D diffusion models to output 3D representations, but the quality is limited due to the large 2D-3D domain gap. Other approaches can be dived into two groups.

The first group contains methods that build on DreamFusion, a seminal work by [68], and distill 3D objects by optimizing NeRF via the SDS loss, matching its renders to the belief of a pre-trained text-to-image model. Extensions have considered: (i) other 3D representations like hash grids [44, 69], meshes [44] and 3D Gaussians (3DGS) [81, 111, 12]; (ii) improved SDS [91, 95, 119, 30]; (iii) monocular conditioning [69, 82, 113, 78]; (iv) predicting additional normals or depth for better geometry [70, 78]. Yet, distillation methods are prone to issues such as the Janus effect (duplicating object parts) and content drift [74]. A common solution is to incorporate view-consistency priors into the diffusion model, by either conditioning on cameras [47, 73, 32, 11, 69] or by generating multiple object views jointly [74, 98, 93, 40, 118]. Additionally, SDS optimization is slow and requires minutes to hours per assets; this issue is partly addressed in [52, 101] with amortized SDS.

The second group of methods includes faster two-stage approaches [46, 51, 49, 107, 106, 8, 83, 28, 23] that start by generating multiple views of the object using a text-to-image or text-to-video model [54, 13] tuned to output multiple views of the object followed by per-scene optimization using NeRF [58] or 3DGS [38]. However, per-scene optimization requires several highly-consistent views which are difficult to generate reliably. Instant3D [42] improves speed and robustness by generating a grid of just four views followed by a feed-forward network (LRM [29]) that reconstructs the object from these. One-2–3–45++ [45] replaces the LRM with a 3D diffusion model. Our AssetGen builds on the Instant3D paradigm and upgrades the LRM to output PBR materials and an to use a SDF-based representation of 3D shape. Furthermore, it starts from grids of four views with shaded and albedo channels, key to predicting accurate 3D shape and materials from images.

3D reconstruction from images. 3D scene reconstruction, in its traditional *multi-view stereo* (MVS) sense, assumes access to a dense set of scene views. Recent reconstruction methods such as NeRF [58] optimize a 3D representation by minimizing multi-view rendering losses. There are two popular classes of 3D representation: (i) explicit representations like meshes [22, 114, 24, 63, 59, 76] or 3D points/Gaussians [38, 25], and (ii) implicit representations like occupancy fields [65], radiance fields [58, 62] and signed distance functions (SDF) [109]. Compared to occupancy fields, SDF [66, 108, 92, 17, 21] simplifies surface constraints integration, improving scene geometry. For this reason, we also use SDFs and demonstrate that they outperform occupancy fields.

Sparse-view reconstruction instead assumes few input views (usually 1 to 8). An approach to mitigate the lack of dense multiple views is to leverage 2D diffusion priors in optimization [55, 99], but this is often slow and fragile. More recently, authors have focused on training feed-forward reconstructors on large datasets [14, 36, 57, 48, 100, 60, 94]. In particular, [29] trains a large Transformer [89] to predict NeRF using a triplane representation [7, 9]. Followups study 3D representations like meshes [103, 97] and 3DGS [120, 105, 80, 115], improved backbones [96, 97] and training protocols [86, 35]. We introduce three extensions to LRM: (i) an SDF formulation for improved geometry, (ii) PBR material prediction for relighting, and (iii) a texture refiner for better texture details.

3D modeling with PBR materials. Most 3D generators output 3D objects with baked illumination, either view-dependent [58, 38] or view-independent [29]. Since baked lighting ignores the model's response to environmental illumination, it is unsuitable for graphics pipelines that simulate lighting. Physically-based rendering (PBR) defines material properties so that a suitable shader can account for illumination realistically. Several MVS works have considered estimating PBR materials using NeRF [4, 3, 102], SDF [116], differentiable meshes [63, 27] or 3DGS [33, 43]. In generative modelling, [10, 70, 50, 104] augment the text-to-3D SDS optimization [68] with a PBR model. Differently from them, we integrate PBR modeling in our feed-forward text-to-3D network, unlocking for the first time fast text-based generation of 3D assets with controllable PBR materials.

3 Method

AssetGen is a two-stage pipeline (Fig. 2). The first stage, text-to-image (Sec. 3.1), maps text to an image grid containing four object views with material information. The second stage, image-to-3D, comprises a novel PBR-based sparse-view reconstruction model (Sec. 3.2) and a new texture refiner (Sec. 3.3). As such, AssetGen is applicable to two tasks: text-to-3D (stage 1+2) and image-to-3D (stage 2 only).

 https://doi.org/10.52202/079017-0303

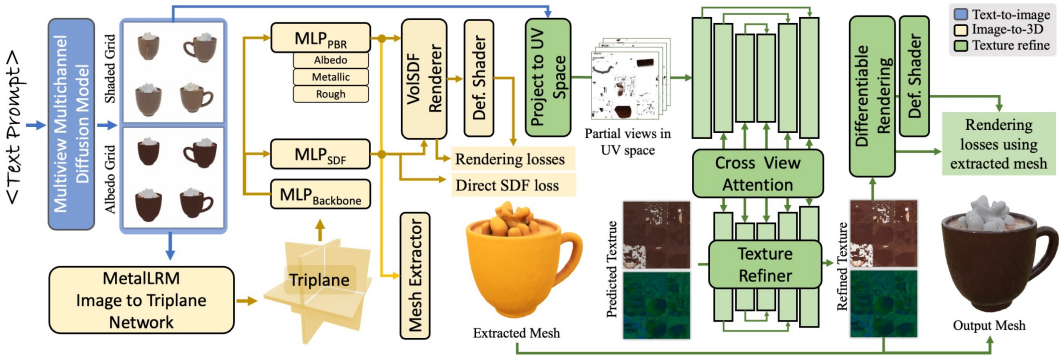


Figure 2: **Overview.** Given a text prompt, AssetGen generates a 3D mesh with PBR materials in two stages. The first text-to-image stage (blue) predicts a 6-channel image depicting 4 views of the object with shaded and albedo colors. The second image-to-3D stage includes two steps. First, a 3D reconstructor (dubbed MetaILRM) outputs a triplane-supported SDF field converted into a mesh with textured PBR materials (orange). Then, PBR materials are enhanced with our texture refiner which recovers missing details from the input views (green).

3.1 Text-to-image: Generating shaded and albedo images from text

The goal of the text-to-image module is to generate several views of the generated 3D object. To this end, we employ an internal text-to-image diffusion model pre-trained on billions of text-annotated images, with an architecture similar to Emu [16]. Similar to [74, 42], we finetune the model to predict a grid of four images I_i, $i = 1, \dots, 4$, each depicting the object from canonical viewpoints π_i. Note that I_i are RGB images of the shaded object. We tried deferring the PBR parameter extraction to the image-to-3D stage, but this led to suboptimal results. This is due to the determinism of the image-to-3D stage, which fails to model ambiguities when assigning materials to surfaces.

A natural solution, then, is to predict the PBR parameters directly in the text-to-image stage. These consists of the albedo ρ_0 (by which we mean the base color, which is the same as albedo only for zero metalness), the metalness γ, and the roughness α. However, we found this to be ineffective too because the metalness and roughness maps deviate from the distribution of natural images making them a hard target for finetuning. Our novel solution is to train the model to generate instead a **4-view grid with 6 channels**, 3 for the shaded appearance I and 3 more for the albedo ρ_0. This reduces the finetuning gap, and removes enough ambiguity for accurate PBR prediction in the image-to-3D stage.

3.2 Image-to-3D: A PBR-based large reconstruction model

We now describe the image-to-3D stage, which solves the reconstruction tasks given either a small number of views I_i (few-view reconstruction), or the 4-view 6-channel grid of Sec. 3.1.

At the core of our method is a new PBR-aware reconstruction model, MetaILRM, that reconstructs the object given $N \geq 1$ *posed* images $(I_i, \pi_i)_{i=1}^N$, where $I_i \in \mathbb{R}^{H \times W \times D}$ and $\pi_i \in \Pi$ is the camera viewpoint. As noted in Sec. 3.1, we consider $N = 4$ canonical viewpoints $\pi_1, \dots, \pi_4$ (fixed to 20° elevation and 0°, 90°, 180°, 270° azimuths) and $D = 6$ input channels. The output is a 3D field representing the shape and PBR materials of the object as an SDF $s : \mathbb{R}^3 \to \mathbb{R}$, where $s(\boldsymbol{x})$ is the signed distance from the 3D point $\boldsymbol{x}$ to the nearest object surface point, and a PBR function $k : \mathbb{R}^3 \to \mathbb{R}^5$, where $k(\boldsymbol{x}) = (\rho_0, \gamma, \alpha)$ are the albedo, metalness and roughness.

The key to learning the model is the *differentiable rendering* operator $\mathcal{R}$. This takes as input a field $\ell : \mathbb{R}^3 \to \mathbb{R}^D$, the SDF s, the viewpoint π, and a pixel $u \in U = [0, W) \times [0, H)$, and outputs the projection of the field on the pixel according to the rendering equation [58], which has the same number of channels D as the rendered field ℓ:

$$\mathcal{R}(u \mid \ell, s, \pi) = \int_0^\infty \ell(\boldsymbol{x}_t)\sigma(\boldsymbol{x}_t \mid s) e^{-\int_0^t \sigma(\boldsymbol{x}_\tau \mid s)\, d\tau} dt. \tag{1}$$

Here $\boldsymbol{x}_t = \boldsymbol{x}_0 - t\boldsymbol{\omega}_o$, $t \in [0, \infty)$ is the ray that goes from the camera center $\boldsymbol{x}_0$ through the pixel u along direction $-\boldsymbol{\omega}_o \in \mathbb{S}^2$. The function $\sigma(\boldsymbol{x} \mid s)$ is the opacity of the 3D point $\boldsymbol{x}$ and is obtained

from the SDF value $s(\boldsymbol{x})$ using the VolSDF [108] formula

$$\sigma(\boldsymbol{x} \mid s) = \frac{a}{2}\Big(1 + \operatorname{sign} s(\boldsymbol{x})\,\big(1 - e^{-|s(\boldsymbol{x})|/b}\big)\Big), \tag{2}$$

where a, b are the hyper-parameters. We use Eq. (1) to render several different types of fields ℓ, the most important of which is the radiance field, introduced next along with the material model.

Reflectance model. The appearance of the object $\hat{I}(u) = \mathcal{R}(u \mid L, s, \pi)$ in a shaded RGB image $\hat{I}$ is obtained by rendering its *radiance field* $\ell(\boldsymbol{x}) = L(\boldsymbol{x}, \boldsymbol{\omega}_{\text{o}} \mid k, \boldsymbol{n})$, where $\boldsymbol{n}$ is the field of unit normals. The radiance is the light reflected by the object in the direction $\boldsymbol{\omega}_{\text{o}}$ of the observer (see App. A.8 for details), which in PBR is given by:

$$L(\boldsymbol{x}, \boldsymbol{\omega}_{\text{o}} \mid k, \boldsymbol{n}) = \int_{H(\boldsymbol{n})} f(\boldsymbol{\omega}_{\text{i}}, \boldsymbol{\omega}_{\text{o}} \mid k(\boldsymbol{x}), \boldsymbol{n}(\boldsymbol{x})) L(\boldsymbol{x}, -\boldsymbol{\omega}_{\text{i}})(\boldsymbol{n}(\boldsymbol{x}) \cdot \boldsymbol{\omega}_{\text{i}})\, d\Omega_{\text{i}}, \tag{3}$$

where $\boldsymbol{\omega}_{\text{o}}, \boldsymbol{\omega}_{\text{i}} \in H(\boldsymbol{n}) = \{\boldsymbol{\omega} \in \mathbb{S}^2 : \boldsymbol{n} \cdot \boldsymbol{\omega} \geq 0\}$ are two unit vectors pointing outside the object and $L(\boldsymbol{x}, -\boldsymbol{\omega}_{\text{i}})$ is the radiance incoming from the environment at $\boldsymbol{x}$ from direction $\boldsymbol{\omega}_{\text{i}}$ in the solid angle $d\Omega_{\text{i}}$. The *Bidirectional Reflectance Distribution Function* (BRDF) f tells how light received from direction $-\boldsymbol{\omega}_{\text{i}}$ (incoming) is scattered into different directions $\boldsymbol{\omega}_{\text{o}}$ (outgoing) by the object [20].

In PBR, we consider a physically-inspired model for the BRDF, striking a balance between realism and complexity [2, 87, 77, 15, 90]; specifically, we use the Disney GGX model [90, 5], which depends on parameters ρ_0, γ, and α only (see App. A.12.1 for the parametric form of f). Hence, the MetaI LRM predicts the triplet $k(\boldsymbol{x}) = (\rho_0, \gamma, \alpha)$ at each 3D point $\boldsymbol{x}$.

Deferred shading. In practice, instead of computing $\hat{I}(u) = \mathcal{R}(u \mid L, s, \pi)$ using Eqs. (1) and (3), we use the process of *deferred shading* [20]:

$$\hat{I}(u) = \mathcal{R}_{\text{def}}(u \mid k, s, \pi) = \int_{H(\boldsymbol{n})} f(\boldsymbol{\omega}_{\text{i}}, \boldsymbol{\omega}_{\text{o}} \mid \bar{k}, \bar{\boldsymbol{n}}) L_{\text{env}}(-\boldsymbol{\omega}_{\text{i}})(\bar{\boldsymbol{n}} \cdot \boldsymbol{\omega}_{\text{i}})\, d\Omega_{\text{i}}, \tag{4}$$

where L_{env} is the environment radiance (assumed to be the same for all $\boldsymbol{x}$), $\bar{k} = \mathcal{R}(u \mid k, s, \pi)$ and $\bar{\boldsymbol{n}} = \mathcal{R}(u \mid \boldsymbol{n}, s, \pi)$ are rendered versions of the material and normal fields. The advantage of Eq. (4) is that the BRDF f is evaluated only once per pixel, which is much faster and less memory intensive than doing so for each 3D point during the evaluation of Eq. (1), particularly for training/backpropagation. During training, furthermore, the environment light is assumed to be a single light source at infinity, so the integral (4) reduces to evaluating a single term.

Training formulation and losses. MetaILRM is thus a neural network that takes as input a set of images $(I_i, \pi_i)_{i=1}^N$ and outputs estimates $\hat{s}$ and $\hat{k}$ for the SDF and PBR fields. We train it from a dataset of mesh surfaces $M \subset \mathbb{R}^3$ with ground truth PBR materials $k : M \to \mathbb{R}^5$.

Reconstruction models are typically trained via supervision on renders [29, 97]. However, physically accurate rendering via Eq. (1) is very expensive. We overcome this hurdle in two ways. First, we render the raw ground-truth PBR fields k and use them to supervise their predicted counterparts with the MSE loss, skipping Eq. (1). For the rendered albedo ρ_0 — which is similar enough to natural images — we also use the LPIPS [117] loss:

$$\mathcal{L}_{\text{pbr}} = \operatorname{LPIPS}\Big(\mathcal{R}(\cdot \mid \hat{\rho}_0, \hat{s}, \pi), \mathcal{R}(\cdot \mid \rho_0, M, \pi)\Big) + \Big\|\mathcal{R}(\cdot \mid \hat{k}, \hat{s}, \pi) - \mathcal{R}(\cdot \mid k, M, \pi)\Big\|^2. \tag{5}$$

We further supervise the PBR field by adding a computationally-efficient deferred shading loss:

$$\mathcal{L}_{\text{def}} = \|\sqrt{w} \odot (\mathcal{R}_{\text{def}}(\cdot \mid \hat{k}, \hat{s}, \pi) - \mathcal{R}_{\text{def}}(\cdot \mid k, M, \pi))\|^2. \tag{6}$$

The weight $w(u) = \hat{\boldsymbol{n}}(u) \cdot \boldsymbol{n}(u)$ is the dot product of the predicted and ground-truth normals at pixel u. It discounts the loss where the predicted geometry is not yet learnt. Fig. 14 (b) visualizes deferred shading and the rendering loss.

Finally, we also supervise the SDF field with a direct loss $\mathcal{L}_{\text{sdf}}$ (implemented as in [1]), a depth-MSE loss $\mathcal{L}_{\text{depth}}$ between the depth renders and the ground truth, and with a binary cross-entropy $\mathcal{L}_{\text{mask}}$ between the alpha-mask renders and the ground-truth masks. Refer to App. A.6.2 for more details.

LightPlane implementation. We base MetaILRM on LightplaneLRM [6], a variant of LRM [29] exploiting memory and compute-efficient Lightplane splatting and rendering kernels, offering better quality reconstructions. However, since LightplaneLRM uses density fields, which are suboptimal for mesh conversion [92, 66, 1], we extend the Lightplane rendering GPU kernel with a VolSDF [108] renderer using Eq. (2). Additionally, we also fuse into the kernel the direct SDF loss $\mathcal{L}_{\text{sdf}}$ since a naive autograd implementation is too memory-heavy.

https://doi.org/10.52202/079017-0303

3.3 Mesh extraction and texture refiner

The MetaILRM module of Sec. 3.2 outputs a sign distance function s, implicitly defining the object surface $A = \{\boldsymbol{x} \in \mathbb{R}^3 \mid s(\boldsymbol{x}) = 0\}$ as a level set of s. We use the Marching Tetrahedra algorithm [18] to trace the level set and output a mesh $M \approx A$. Then, xAtlas [112] extracts a UV map $\phi : [0, V]^2 \to M$, mapping each 2D UV-space point $v = \phi(\boldsymbol{x})$ to a point $\boldsymbol{x} \in M$ on the mesh.

Next, the goal is to extract a high-quality 5-channel PBR texture image $\bar{K} \in \mathbb{R}^{V \times V \times 5}$ capturing the albedo, metalness, and roughness of each mesh point. The texture image K can be defined directly by sampling the predicted PBR field $\hat{k}$ as $K(v) \leftarrow \hat{k}(\phi(v))$, but this often yields blurry results due to the limited resolution of MetaILRM. Instead, we design a texture refiner module which takes as input the coarse PBR-sampled texture image as well as the N views representing the object and outputs a much sharper texture $\bar{K}$. In essence, this modules leverages the information from the different views to refine the coarse texture image. The right part of Fig. 2 illustrates this module.

More specifically, it relies on a network Φ which is fed $N + 1$ texture images $\{K_i\}_{i=0}^N$. First, each pixel $v \in [0, V]^2$ of $K_0 \in \mathbb{R}^{V \times V \times 11}$ is annotated with the concatenation of the normal, the 3D location, and the output of MetaILRM's PBR field $k(\phi(v))$ evaluated at v's 3D point $\phi(v)$. The remaining $K_1, \ldots, K_N$ correspond to partial texture images with 6 channels (for the base and shaded colors) which are obtained by back-projecting the object views to the mesh surface. The network Φ utilises two U-Nets to fuse $\{K_i\}_{i=0}^N$ into the enhanced texture $\bar{K}$. Φ's goal is to select, for each UV point v, which of the N input views provides the best information. Specifically, each partial texture image K_i is processed in parallel by a first U-Net, and the resulting information is communicated via cross attention to a second U-Net whose goal is to refine K_0 into the enhanced texture $\bar{K}$. Please refer to App. A.7 for further details.

Such a network is trained on the same dataset and supervised with the PBR and albedo rendering losses as MetaILRM. The only difference is meshes (whose geometry is fixed) are rendered differentiably using PyTorch3D's [71] mesh rasterizer instead of the Lightplane SDF renderer.

4 Experiments

Our **training data** consists of 140,000 meshes of diverse semantic categories created by 3D artists. For each asset, we render 36 views at random elevations within the range of $[-30°, 50°]$ at uniform intervals of $30°$ around the object, lit with a randomly selected environment map. We render the shaded images, albedo, metalness, roughness, depth maps, and foreground masks from each viewpoint. The text-to-image stage is based on an internal text-to-image model architecturally similar to Emu [16], fine-tuned on a subset of 10,000 high-quality 3D samples, captioned by a Cap3D-like pipeline [53] that uses Llama3 [88]. The other stage utilizes the entire 3D dataset instead.

For **evaluation**, following [105, 103, 42], we assess visual quality using PSNR and LPIPS [117] between the rendered and ground-truth images. PSNR is computed in the foreground region to avoid metric inflation due to the empty background. Geometric quality is measured by the L1 error between the rendered and ground-truth depth maps (of the foreground pixels), as well as the IoU of the object silhouette. We further report Chamfer Distance (CD) and Normal Correctness (NC) for 20,000 points uniformly sampled on both the predicted and ground-truth shapes. Material decomposition

Table 1: **Four-view reconstruction with PBR** evaluating the accuracy of the PBR renders for MetaI LRM and ablations. Methods in top / bottom accept 4 views with shaded / shaded&albedo color channels.

Method	LPIPS↓ albedo	PSNR↑ albedo	metal	rough
C = LightplaneLRM w/ SDF	0.117	17.14	12.39	15.25
E = C + Material prediction	0.097	20.66	15.99	20.25
F = E + Deferred shading loss	0.093	21.12	18.64	20.66
G = F + Texture refinement	**0.087**	**21.97**	**22.19**	**20.85**
H = F + Albedo & shaded input	0.084	23.02	20.43	**21.18**
I = H + Texture refinement	**0.069**	**24.39**	**27.28**	20.63

Table 2: **Win-rate of AssetGen in text-to-3D user study** evaluating visual quality and the alignment between the prompt and the generated meshes. AssetGen beats all baselines at 30 sec budget (on an A100 GPU).

Method	Visual quality	Text fidelity	PBR
GRM [105]	96.7 %	93.3 %	✗
InstantMesh [103]	99.3 %	97.3 %	✗
LightplaneLRM [6]	66.6 %	N/A	✗
Meshy v3 [85]	94.6 %	91.3 %	✓
Luma Genie 1.0 [84]	72.3 %	72.8 %	✓

is evaluated with LPIPS and PSNR on the albedo image, and PSNR alone for the metalness and roughness channels. All metrics are calculated on meshified outputs rather than on neural renders.

4.1 Sparse-view reconstruction

We tackle the sparse-view reconstruction task of predicting a 3D mesh from 4 posed images of an object on a subset of 332 meshes from Google Scanned Objects (GSO) [19]. We compare against state-of-the-art Instant3D-LRM [42], GRM [105], InstantMesh [103], and MeshLRM [97]. We also include LightplaneLRM [6], an improved version of Instant3D-LRM, which serves as our base model. MeshLRM [97] has not been open-sourced so we compare only qualitatively to meshes from their webpage. All methods are evaluated using the same input views at 512^2 resolution. Since none of the latter predict PBR materials and since GSO lacks ground-truth PBR materials, for fairness, we use a variant of our model that predicts shaded object textures.

Table 3: **Four-view reconstruction on GSO** comparing the appearance and geometry of MetaILRM (outputting baked-light texture) to baselines (top) and ablations (bottom). CD values multiplied by 10^{-2}.

Method	LPIPS↓	PSNR↑	Depth↓	IoU↑	CD↓	NC↑
Instant3D-LRM [29]	0.124	18.54	0.325	0.930	1.630	0.844
GRM [105]	0.100	19.87	0.364	0.949	1.490	0.873
InstantMesh [103]	0.113	20.63	0.334	0.937	1.364	0.848
MetaILRM (ours)	**0.057**	**22.49**	**0.173**	**0.968**	**1.137**	**0.885**
A = LightplaneLRM [6]	0.095	18.60	0.456	0.953	1.313	0.872
B = A + VolSDF rendering	0.094	20.91	0.201	0.957	1.212	0.875
C = B + Direct SDF loss	0.083	21.75	**0.173**	**0.968**	**1.137**	**0.885**
D = C + Texture refinement	**0.057**	**22.49**	**0.173**	**0.968**	**1.137**	**0.885**

As shown in Figs. 4 and 9 and Tab. 3, our method outperforms all baselines across all metrics. GRM captures texture detail well but struggles with fine geometric structures when meshified. InstantMesh and LightplaneLRM improve geometry but fall short on finer details and texture quality. Our approach excels in reconstructing shapes with detailed geometry and high-fidelity textures.

Ablations in Tab. 3 and Fig. 4 show that incorporating our scalable SDF-based rendering and direct SDF loss into the base LightplaneLRM model enhances geometric quality. Adding texture refinement further brings fine texture details.

Next, we consider the task of **sparse-view reconstruction with PBR materials**, where the goal is to reconstruct the 3D geometry and texture properties (albedo, metalness, and roughness) from four posed shaded 2D views of an object. This is done on an internal dataset of 256 artist-created 3D meshes, curated for high-quality materials. Since there are no existing few-view feed-forward PBR reconstructors, we conduct an ablation study in Tab. 1 and Figs. 3 and 13.

While adding material prediction with additional MLP heads provides some improvements, we observe that incorporating the deferred shading loss and texture refinement is essential for high-quality PBR decomposition. Example PBR predictions are shown in Fig. 8.

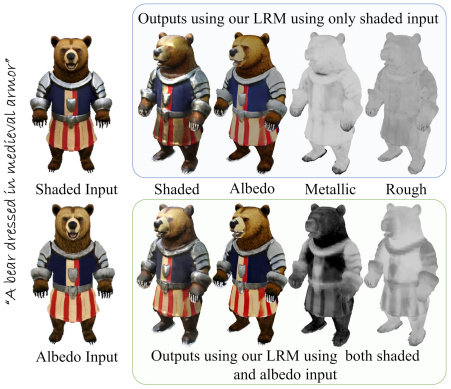


Figure 3: **Qualitative ablation on albedo generation.** In text-to-3D, generating 4 views representing albedo colors alongside shaded RGB colors improves material estimation for our 3D reconstructor. With both inputs, the model accurately predicts the armor as metallic and smooth, while the bear's fur is rough.

4.2 Text-to-3D generation

Finally, we evaluate text-to-3D with PBR materials. We compare against state-of-the-art feed-forward methods that generate assets at comparable speed ($\approx$ 10 to 30 s per asset). This includes text-to-3D variants of GRM [105], InstantMesh [103], and LightplaneLRM [6]. GRM uses Instant3D's 4-view grid generator, InstantMesh receives the first view from our 2D diffusion model and subsequently generates 6 views, while LightplaneLRM accepts 4 views from our grid generator. Since these methods bake lighting instead of generating PBR materials, for evaluation we apply flat texture

https://doi.org/10.52202/079017-0303

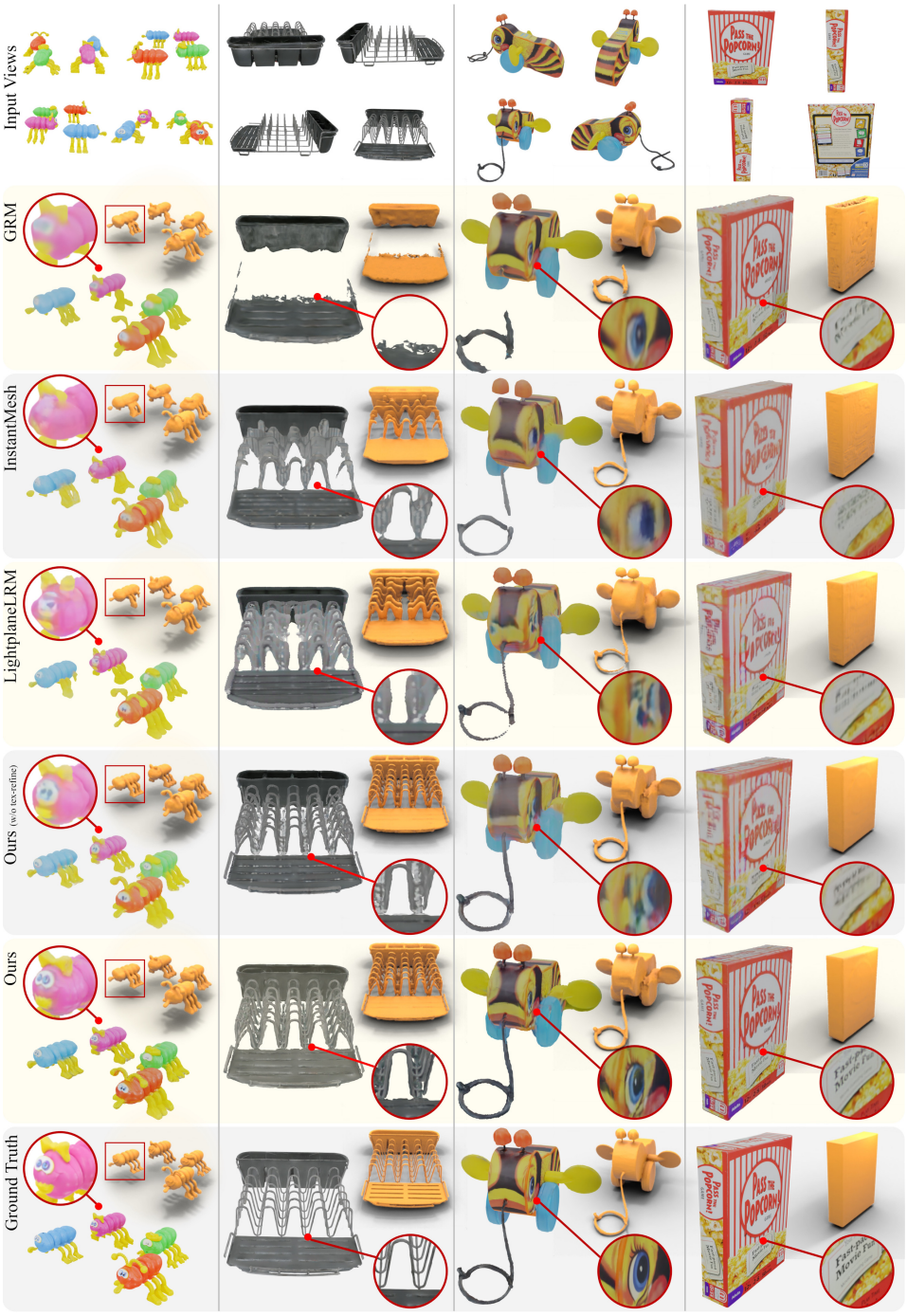


Figure 4: **Qualitative comparison for sparse-view reconstruction.** AssetGen gives better geometry (shown in orange) and higher fidelity texture (inset) compared to state of the art. SDF representation along with the direct SDF loss gives a better geometry compared to the base LightplaneLRM model which uses occupancy (row 4 and 5). Furthermore, our texture refiner greatly enhances texture fidelity (row 5 and 6).

https://doi.org/10.52202/079017-0303

Figure 5: **Qualitative comparison for text-to-3D.** We compare 3D meshes generated by Meta 3D AssetGen and state-of-the-art baselines. We include material decomposition for methods producing PBR materials (Luma Genie and our Meta 3D AssetGen). Our approach produces higher quality materials with better-defined metalness and roughness, and a more accurate decoupling of lighting effects in the albedo.

shading to our outputs. Additionally, we compare with the preview stage of Meshy v3 [85] and LumaAI Genie 1.0 [84], proprietary text-to-3D methods with PBR workflow capable of creating assets within 30 and 15 s respectively. A comparison with the significantly longer refinement stages for Luma and Meshy is provided in the appendix. Fig. 5 shows that AssetGen meshes are visually more appealing and have meaningful materials Figs. 6 and 12 provide more examples and comparisons and showcase fine-grained material control.

For quantitative evaluation, we conducted an extensive user study in Tab. 2 using the 404 deduplicated text prompts from DreamFusion [68]. Users were shown 360° videos of the generated and baseline meshes and were asked to rate them based on 3D shape quality and alignment with the text prompt. A total of 11,080 responses were collected, with significant preference for AssetGen's meshes.

Finally, we ablate the effect of generating dual-channel albedo+shaded grids compared to albedo-only input in Fig. 3 revealing significant PBR decomposition superiority of the former. Additionally, Fig. 13 illustrates the effect of our deferred shading loss.

5 Conclusions

We have introduced Meta 3D AssetGen, a significant advancement in sparse-view reconstruction and text-to-3D. Meta 3D AssetGen can generate 3D meshes with high-quality textures and PBR materials faithful to the input text. This uses several key innovations: generating multi-view grids with both shaded and albedo channels, introducing a new reconstruction network that predicts PBR materials from this information, using deferred shading to train this network, improving geometry via a new scalable SDF-based renderer and SDF loss, and introducing a new texture refinement network. Comprehensive evaluations and ablations demonstrate the effectiveness of these design choices and state-of-the-art performance.

https://doi.org/10.52202/079017-0303

Figure 6: **Text-to-3D** meshes generated by Meta 3D AssetGen along with their PBR decomposition. Note that Meta 3D AssetGen provides detailed albedo and material properties, as highlighted by the metalness of the platter (top right) and the golden objects (last row).

https://doi.org/10.52202/079017-0303

References

[1] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6290–6301, June 2022.

[2] P. Beckmann and A. Spizzichino. *The Scattering of Electromagnetic Waves from Rough Surfaces*. Pergamon Press, 1963.

[3] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P.A. Lensch. NeRD: Neural Reflectance Decomposition from Image Collections. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[4] Mark Boss, Varun Jampani, Raphael Braun, Ce Liu, Jonathan T. Barron, and Hendrik P. A. Lensch. Neural-PIL: Neural Pre-Integrated Lighting for Reflectance Decomposition. *arXiv preprint*, 2021.

[5] Brent Burley. Physically-based shading at disney. Technical report, Disney, 2012.

[6] Ang Cao, Justin Johnson, Andrea Vedaldi, and David Novotny. Lightplane: Highly-scalable components for neural 3d fields. *arXiv*, 2024.

[7] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J. Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *Proc. CVPR*, 2022.

[8] Eric R. Chan, Koki Nagano, Matthew A. Chan, Alexander W. Bergman, Jeong Joon Park, Axel Levy, Miika Aittala, Shalini De Mello, Tero Karras, and Gordon Wetzstein. Generative novel view synthesis with 3D-aware diffusion models. In *Proc. ICCV*, 2023.

[9] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial radiance fields. In *arXiv*, 2022.

[10] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3D: Disentangling geometry and appearance for high-quality text-to-3D content creation: Disentangling geometry and appearance for high-quality text-to-3d content creation. *arXiv.cs*, abs/2303.13873, 2023.

[11] Yabo Chen, Jiemin Fang, Yuyang Huang, Taoran Yi, Xiaopeng Zhang, Lingxi Xie, Xinggang Wang, Wenrui Dai, Hongkai Xiong, and Qi Tian. Cascade-Zero123: One image to highly consistent 3D with self-prompted nearby views. *arXiv.cs*, abs/2312.04424, 2023.

[12] Zilong Chen, Feng Wang, and Huaping Liu. Text-to-3D using Gaussian splatting. *arXiv*, 2309.16585, 2023.

[13] Zilong Chen, Yikai Wang, Feng Wang, Zhengyi Wang, and Huaping Liu. V3D: Video diffusion models are effective 3D generators. *arXiv*, 2403.06738, 2024.

[14] Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In *Proc. ECCV*, 2016.

[15] Robert L. Cook and Kenneth E. Torrance. A reflectance model for computer graphics. In Doug Green, Tony Lucido, and Henry Fuchs, editors, *Proc. SIGGRAPH*, 1981.

[16] Xiaoliang Dai, Ji Hou, Chih-Yao Ma, Sam S. Tsai, Jialiang Wang, Rui Wang, Peizhao Zhang, Simon Vandenhende, Xiaofang Wang, Abhimanyu Dubey, Matthew Yu, Abhishek Kadian, Filip Radenovic, Dhruv Mahajan, Kunpeng Li, Yue Zhao, Vladan Petrovic, Mitesh Kumar Singh, Simran Motwani, Yi Wen, Yiwen Song, Roshan Sumbaly, Vignesh Ramanathan, Zijian He, Peter Vajda, and Devi Parikh. Emu: Enhancing image generation models using photogenic needles in a haystack. *CoRR*, abs/2309.15807, 2023.

[17] François Darmon, Bénédicte Bascle, Jean-Clément Devaux, Pascal Monasse, and Mathieu Aubry. Improving neural implicit surfaces geometry with patch warping. In *Proc. CVPR*, 2022.

 https://doi.org/10.52202/079017-0303

[18] Akio Doi and Akio Koide. An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. *IEICE TRANSACTIONS on Information and Systems*, 74(1):214–224, 1991.

[19] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B. McHugh, and Vincent Vanhoucke. Google Scanned Objects: A high-quality dataset of 3D scanned household items. In *Proc. ICRA*, 2022.

[20] James D. Foley, Andries van Dam, Steven Feiner, and John F. Hughes. *Computer graphics - principles and practice, 3nd Edition*. Addison-Wesley, 2013.

[21] Qiancheng Fu, Qingshan Xu, Yew-Soon Ong, and Wenbing Tao. Geo-Neus: Geometry-Consistent Neural Implicit Surfaces Learning for Multi-view Reconstruction. In *NeurIPS*, 2022.

[22] Jun Gao, Wenzheng Chen, Tommy Xiang, Clement Fuji Tsang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Learning deformable tetrahedral meshes for 3D reconstruction. In *Proc. NeurIPS*, 2020.

[23] Ruiqi Gao, Aleksander Holynski, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul Srinivasan, Jonathan T. Barron, and Ben Poole. CAT3D: Create Anything in 3D with Multi-View Diffusion Models. *arXiv.cs*, 2024.

[24] Shubham Goel, Georgia Gkioxari, and Jitendra Malik. Differentiable Stereopsis: Meshes from multiple views using differentiable rendering. In *CVPR*, 2022.

[25] Antoine Guédon and Vincent Lepetit. SuGaR: Surface-aligned Gaussian splatting for efficient 3D mesh reconstruction and high-quality mesh rendering. *arXiv.cs*, abs/2311.12775, 2023.

[26] Anchit Gupta, Wenhan Xiong, Yixin Nie, Ian Jones, and Barlas Oguz. 3DGen: Triplane latent diffusion for textured mesh generation. *corr*, abs/2303.05371, 2023.

[27] Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. Shape, Light, and Material Decomposition from Images using Monte Carlo Rendering and Denoising. *arXiv preprint*, 2022.

[28] Lukas Höllein, Aljaž Božič, Norman Müller, David Novotny, Hung-Yu Tseng, Christian Richardt, Michael Zollhöfer, and Matthias Nießner. ViewDiff: 3D-Consistent Image Generation with Text-to-Image Models. *arXiv preprint*, 2024.

[29] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. LRM: Large reconstruction model for single image to 3D. In *Proc. ICLR*, 2024.

[30] Yukun Huang, Jianan Wang, Yukai Shi, Xianbiao Qi, Zheng-Jun Zha, and Lei Zhang. Dreamtime: An improved optimization strategy for text-to-3D content creation. *CoRR*, abs/2306.12422, 2023.

[31] Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. *CVPR*, 2022.

[32] Yifan Jiang, Hao Tang, Jen-Hao Rick Chang, Liangchen Song, Zhangyang Wang, and Liangliang Cao. Efficient-3Dim: Learning a generalizable single-image novel-view synthesizer in one day. *arXiv*, 2023.

[33] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. GaussianShader: 3D Gaussian splatting with shading functions for reflective surfaces. *arXiv.cs*, abs/2311.17977, 2023.

[34] Heewoo Jun and Alex Nichol. Shape-E: Generating conditional 3D implicit functions. *arXiv*, 2023.

[35] Philip Torr Junlin Han, Filippos Kokkinos. Vfusion3d: Learning scalable 3d generative models from video diffusion models. *arXiv preprint*, 2024.

[36] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *Proc. ECCV*, 2018.

[37] Animesh Karnewar, Andrea Vedaldi, David Novotny, and Niloy Mitra. HoloDiffusion: training a 3D diffusion model using 2D images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[38] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for real-time radiance field rendering. *Proc. SIGGRAPH*, 42(4), 2023.

[39] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Popa Tiberiu. Clip-mesh: Generating textured meshes from text using pretrained image-text models. *SIGGRAPH Asia 2022 Conference Papers*, December 2022.

[40] Seungwook Kim, Yichun Shi, Kejie Li, Minsu Cho, and Peng Wang. Multi-view image prompted multi-view diffusion for improved 3D generation. *arXiv*, 2404.17419, 2024.

[41] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Proc. ICLR*, 2015.

[42] Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. Instant3D: Fast text-to-3D with sparse-view generation and large reconstruction model. *Proc. ICLR*, 2024.

[43] Zhihao Liang, Qi Zhang, Ying Feng, Ying Shan, and Kui Jia. GS-IR: 3D Gaussian splatting for inverse rendering. *arXiv.cs*, abs/2311.16473, 2023.

[44] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3D: High-resolution text-to-3D content creation. *arXiv.cs*, abs/2211.10440, 2022.

[45] Minghua Liu, Ruoxi Shi, Linghao Chen, Zhuoyang Zhang, Chao Xu, Xinyue Wei, Hansheng Chen, Chong Zeng, Jiayuan Gu, and Hao Su. One-2-3-45++: Fast single image to 3D objects with consistent multi-view generation and 3D diffusion. *arXiv.cs*, abs/2311.07885, 2023.

[46] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su. One-2-3-45: Any single image to 3D mesh in 45 seconds without per-shape optimization. In *Proc. NeurIPS*, 2023.

[47] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3D object. In *Proc. ICCV*, 2023.

[48] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3D reasoning. *arXiv.cs*, abs/1904.01786, 2019.

[49] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. SyncDreamer: Generating multiview-consistent images from a single-view image. *arXiv*, 2309.03453, 2023.

[50] Zexiang Liu, Yangguang Li, Youtian Lin, Xin Yu, Sida Peng, Yan-Pei Cao, Xiaojuan Qi, Xiaoshui Huang, Ding Liang, and Wanli Ouyang. UniDream: Unifying Diffusion Priors for Relightable Text-to-3D Generation. *arXiv preprint*, 2023.

[51] Xiaoxiao Long, Yuanchen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, and Wenping Wang. Wonder3D: Single image to 3D using cross-domain diffusion. *arXiv.cs*, abs/2310.15008, 2023.

[52] Jonathan Lorraine, Kevin Xie, Xiaohui Zeng, Chen-Hsuan Lin, Towaki Takikawa, Nicholas Sharp, Tsung-Yi Lin, Ming-Yu Liu, Sanja Fidler, and James Lucas. ATT3D: amortized text-to-3D object synthesis. In *Proc. ICCV*, 2023.

[53] Tiange Luo, Chris Rockwell, Honglak Lee, and Justin Johnson. Scalable 3d captioning with pretrained models. *arXiv preprint*, 2023.

 https://doi.org/10.52202/079017-0303

[54] Luke Melas-Kyriazi, Iro Laina, Christian Rupprecht, Natalia Neverova, Andrea Vedaldi, Oran Gafni, and Filippos Kokkinos. IM-3D: Iterative multiview diffusion and reconstruction for high-quality 3D generation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

[55] Luke Melas-Kyriazi, Christian Rupprecht, Iro Laina, and Andrea Vedaldi. RealFusion: 360 reconstruction of any object from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[56] Antoine Mercier, Ramin Nakhli, Mahesh Reddy, and Rajeev Yasarla. HexaGen3D: Stablediffusion is just one step away from fast and diverse text-to-3D generation. *arXiv*, 2024.

[57] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *CVPR*, 2019.

[58] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proc. ECCV*, 2020.

[59] Tom Monnier, Jake Austin, Angjoo Kanazawa, Alexei A. Efros, and Mathieu Aubry. Differentiable blocks world: Qualitative 3d decomposition by rendering primitives. *arXiv*, abs/2307.05473, 2023.

[60] Tom Monnier, Matthew Fisher, Alexei A. Efros, and Mathieu Aubry. Share With Thy Neighbors: Single-View Reconstruction by Cross-Instance Consistency. In *ECCV*, 2022.

[61] Norman Müller, Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulò, Peter Kontschieder, and Matthias Nießner. DiffRF: Rendering-guided 3D radiance field diffusion. In *Proc. CVPR*, 2023.

[62] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. In *Proc. SIGGRAPH*, 2022.

[63] Jacob Munkberg, Wenzheng Chen, Jon Hasselgren, Alex Evans, Tianchang Shen, Thomas Muller, Jun Gao, and Sanja Fidler. Extracting Triangular 3D Models, Materials, and Lighting From Images. In *CVPR*, 2022.

[64] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-E: A system for generating 3D point clouds from complex prompts. *arXiv.cs*, abs/2212.08751, 2022.

[65] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision. In *CVPR*, 2020.

[66] Michael Oechsle, Songyou Peng, and Andreas Geiger. UNISURF: unifying neural implicit surfaces and radiance fields for multi-view reconstruction. *arXiv.cs*, abs/2104.10078, 2021.

[67] OpenAI. Triton: Open-source gpu programming for neural networks. `https://github.com/triton-lang/triton`.

[68] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D diffusion. In *Proc. ICLR*, 2023.

[69] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, and Bernard Ghanem. Magic123: One image to high-quality 3D object generation using both 2D and 3D diffusion priors. *arXiv.cs*, abs/2306.17843, 2023.

[70] Lingteng Qiu, Guanying Chen, Xiaodong Gu, Qi Zuo, Mutian Xu, Yushuang Wu, Weihao Yuan, Zilong Dong, Liefeng Bo, and Xiaoguang Han. Richdreamer: A generalizable normal-depth diffusion model for detail richness in text-to-3D. *arXiv.cs*, abs/2311.16918, 2023.

[71] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv*, 2020.

[72] Christophe Schlick. An inexpensive BRDF model for physically-based rendering. *Comput. Graph. Forum*, 13(3), 1994.

[73] Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. Zero123++: a single image to consistent multi-view diffusion base model. *arXiv.cs*, abs/2310.15110, 2023.

[74] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. MVDream: Multi-view diffusion for 3D generation. In *Proc. ICLR*, 2024.

[75] J. Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3D neural field generation using triplane diffusion. *arXiv.cs*, abs/2211.16677, 2022.

[76] Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. MeshGPT: Generating triangle meshes with decoder-only transformers. *arXiv.cs*, abs/2311.15475, 2023.

[77] B. Smith. Geometrical shadowing of a random rough surface. *IEEE Trans. on Antennas and Propagation*, 15(5), 1967.

[78] Jingxiang Sun, Bo Zhang, Ruizhi Shao, Lizhen Wang, Wen Liu, Zhenda Xie, and Yebin Liu. DreamCraft3D: Hierarchical 3D generation with bootstrapped diffusion prior. *arXiv.cs*, abs/2310.16818, 2023.

[79] Stanislaw Szymanowicz, Christian Rupprecht, and Andrea Vedaldi. Viewset diffusion: (0-)image-conditioned 3D generative models from 2D data. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023.

[80] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. LGM: Large multi-view Gaussian model for high-resolution 3D content creation. *arXiv*, 2402.05054, 2024.

[81] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. DreamGaussian: Generative gaussian splatting for efficient 3D content creation. *arXiv*, 2309.16653, 2023.

[82] Junshu Tang, Tengfei Wang, Bo Zhang, Ting Zhang, Ran Yi, Lizhuang Ma, and Dong Chen. Make-It-3D: High-fidelity 3d creation from A single image with diffusion prior. *arXiv.cs*, abs/2303.14184, 2023.

[83] Shitao Tang, Jiacheng Chen, Dilin Wang, Chengzhou Tang, Fuyang Zhang, Yuchen Fan, Vikas Chandra, Yasutaka Furukawa, and Rakesh Ranjan. MVDiffusion++: A dense high-resolution multi-view diffusion model for single or sparse-view 3d object reconstruction. *arXiv*, 2402.12712, 2024.

[84] Luma Team. Luma genie 1.0. `https://www.luma-ai.com/luma-genie-1-0/`.

[85] Meshy Team. Meshy - AI 3D Model Generator with pbr materials— meshy.ai. `https://www.meshy.ai/`.

[86] Dmitry Tochilkin, David Pankratz, Zexiang Liu, Zixuan Huang, Adam Letts, Yangguang Li, Ding Liang, Christian Laforte, Varun Jampani, and Yan-Pei Cao. TripoSR: fast 3D object reconstruction from a single image. *arXiv*, 2403.02151, 2024.

[87] K. E. Torrance and E. M. Sparrow. Theory for off-specular reflection from roughened surfaces. *J. Opt. Soc. Am.*, 57(9), 1967.

[88] Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, D. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, A. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez,

 https://doi.org/10.52202/079017-0303

Madian Khabsa, Isabel M. Kloumann, A. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288, 7 2023.

[89] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. NeurIPS*, 2017.

[90] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proc. Eurographics*, 2007.

[91] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Greg Shakhnarovich. Score Jacobian Chaining: Lifting Pretrained 2D Diffusion Models for 3D Generation. In *CVPR*, 2023.

[92] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv.cs*, abs/2106.10689, 2021.

[93] Peng Wang and Yichun Shi. ImageDream: Image-prompt multi-view diffusion for 3D generation. In *Proc. ICLR*, 2024.

[94] Tengfei Wang, Bo Zhang, Ting Zhang, Shuyang Gu, Jianmin Bao, Tadas Baltrusaitis, Jingjing Shen, Dong Chen, Fang Wen, Qifeng Chen, and Baining Guo. Rodin: A generative model for sculpting 3D digital avatars using diffusion. In *Proc. CVPR*, 2023.

[95] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. ProlificDreamer: High-fidelity and diverse text-to-3D generation with variational score distillation. *arXiv.cs*, abs/2305.16213, 2023.

[96] Zhengyi Wang, Yikai Wang, Yifei Chen, Chendong Xiang, Shuo Chen, Dajiang Yu, Chongxuan Li, Hang Su, and Jun Zhu. CRM: Single image to 3D textured mesh with convolutional reconstruction model. *arXiv*, 2403.05034, 2024.

[97] Xinyue Wei, Kai Zhang, Sai Bi, Hao Tan, Fujun Luan, Valentin Deschaintre, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. MeshLRM: large reconstruction model for high-quality mesh. *arXiv*, 2404.12385, 2024.

[98] Haohan Weng, Tianyu Yang, Jianan Wang, Yu Li, Tong Zhang, C. L. Philip Chen, and Lei Zhang. Consistent123: Improve consistency for one image to 3D object synthesis. *arXiv*, 2023.

[99] Rundi Wu, Ben Mildenhall, Philipp Henzler, Keunhong Park, Ruiqi Gao, Daniel Watson, Pratul P. Srinivasan, Dor Verbin, Jonathan T. Barron, Ben Poole, and Aleksander Holynski. ReconFusion: 3D Reconstruction with Diffusion Priors. *arXiv preprint*, 2023.

[100] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Unsupervised learning of probably symmetric deformable 3D objects from images in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[101] Kevin Xie, Jonathan Lorraine, Tianshi Cao, Jun Gao, James Lucas, Antonio Torralba, Sanja Fidler, and Xiaohui Zeng. LATTE3D: Large-scale amortized text-to-enhanced3D synthesis. In *arXiv*, 2024.

[102] Zhang Xiuming, Srinivasan Pratul P., Deng Boyang, Debevec Paul, Freeman William T., and Barron Jonathan T. NeRFactor: neural factorization of shape and reflectance under an unknown illumination. In *Proc. SIGGRAPH*, 2021.

[103] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. InstantMesh: efficient 3D mesh generation from a single image with sparse-view large reconstruction models. *arXiv*, 2404.07191, 2024.

[104] Xudong Xu, Zhaoyang Lyu, Xingang Pan, and Bo Dai. MATLABER: Material-Aware Text-to-3D via LAtent BRDF auto-EncodeR. *arXiv preprint*, 2023.

[105] Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. GRM: Large gaussian reconstruction model for efficient 3D reconstruction and generation. *arXiv*, 2403.14621, 2024.

[106] Jiayu Yang, Ziang Cheng, Yunfei Duan, Pan Ji, and Hongdong Li. ConsistNet: Enforcing 3D consistency for multi-view images diffusion. *arXiv.cs*, abs/2310.10343, 2023.

[107] Yunhan Yang, Yukun Huang, Xiaoyang Wu, Yuan-Chen Guo, Song-Hai Zhang, Hengshuang Zhao, Tong He, and Xihui Liu. DreamComposer: Controllable 3D object generation via multi-view conditions. *arXiv.cs*, abs/2312.03611, 2023.

[108] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *arXiv.cs*, abs/2106.12052, 2021.

[109] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *Proc. NeurIPS*, 2020.

[110] Lior Yariv, Omri Puny, Natalia Neverova, Oran Gafni, and Yaron Lipman. Mosaic-SDF for 3D generative models. *arXiv.cs*, abs/2312.09222, 2023.

[111] Taoran Yi, Jiemin Fang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. GaussianDreamer: Fast generation from text to 3D gaussian splatting with point cloud priors. *arXiv.cs*, abs/2310.08529, 2023.

[112] Jonathan Young. Xatlas: Mesh parameterization / uv unwrapping library, 2022. GitHub repository.

[113] Wangbo Yu, Li Yuan, Yan-Pei Cao, Xiangjun Gao, Xiaoyu Li, Long Quan, Ying Shan, and Yonghong Tian. HiFi-123: Towards high-fidelity one image to 3D content generation. *arXiv.cs*, abs/2310.06744, 2023.

[114] Jason Y. Zhang, Gengshan Yang, Shubham Tulsiani, and Deva Ramanan. NeRS: Neural Reflectance Surfaces for Sparse-view 3D Reconstruction in the Wild. In *NeurIPS*, 2021.

[115] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. GS-LRM: large reconstruction model for 3D Gaussian splatting. *arXiv*, 2404.19702, 2024.

[116] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. PhySG: Inverse Rendering with Spherical Gaussians for Physics-based Material Editing and Relighting. *arXiv preprint*, 2021.

[117] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. CVPR*, pages 586–595, 2018.

[118] Xiaoyu Zhou, Xingjian Ran, Yajiao Xiong, Jinlin He, Zhiwei Lin, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. GALA3D: Towards text-to-3D complex scene generation via layout-guided generative gaussian splatting. *arXiv.cs*, abs/2402.07207, 2024.

[119] Junzhe Zhu and Peiye Zhuang. HiFA: High-fidelity text-to-3D with advanced diffusion guidance. *CoRR*, abs/2305.18766, 2023.

[120] Zi-Xin Zou, Zhipeng Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Yan-Pei Cao, and Song-Hai Zhang. Triplane meets Gaussian splatting: Fast and generalizable single-view 3D reconstruction with transformers. *arXiv.cs*, abs/2312.09147, 2023.

 https://doi.org/10.52202/079017-0303

A Appendix

A.1 Societal Impact

Safeguards should be implemented to prevent abuse, such as filtering input text prompts and detecting unsafe content in generated 3D models. Additionally, our generation process may be vulnerable to biases present in the data and 3D models it relies on, potentially perpetuating these biases in the generated content. Despite these risks, our method can augment the work of artists and creative professionals by serving as a complementary tool to boost productivity. It also holds the potential to democratize 3D content creation, making it accessible to those without specialized knowledge or expensive proprietary software.

A.2 Limitations

Meta 3D AssetGen significantly advances shape generation but faces several limitations. Despite the fine-tuning of the multiview image grid generator for view consistency, it is not guaranteed, potentially impacting 3D reconstruction quality. Since we use an SDF as an underlying representation, the reconstructor may incorrectly model translucent objects or thin structures like hair or fur. Additionally, while our scalable Triton [67] implementation supports a triplane representation at a resolution of 128×128, this representation is inefficient, as much of its capacity is used for empty regions. Future work could explore scalable representations such as octrees, sparse voxel grids, and hash-based methods, which may remove the need for a separate texture enhancement model. We also only predict albedo, metalness and roughness, and not emissivity or ambient occlusions. Finally, our method has only been tested on object-level reconstructions, leaving scene-scale 3D generation for future research.

A.3 Additional qualitative comparisons

This section describes additional qualitative comparisons that, due to limited space, could not be included in the main paper. Firstly, please refer to the video attached in the supplementary material which provides a holistic presentation of Meta 3D AssetGen's qualitative results. In Fig. 7, we highlight the contributions of MetaILRM in geometry, texture and material reconstruction. In Fig. 12, we visualize the control of materials provided by Meta 3D AssetGen, i.e., metalness and roughness, by changing the text prompt for the same concept. Fig. 8 visualizes the renders of the material maps extracted with MetaILRM given four input test views. In Fig. 9, we provide a more extensive qualitative comparison to MeshLRM, the strongest few-view reconstruction baseline. Finally, Fig. 6 provides a gallery of text-conditioned generations depicting Blender-shaded renders together with the rendered PBR maps.

A.4 User-study details

As described in Sec. 4.2, we conducted an user study on 404 meshes generated using the DreamFusion [68] prompt-set on a standard crowdsourcing marketplace. In the study, users were shown 360°

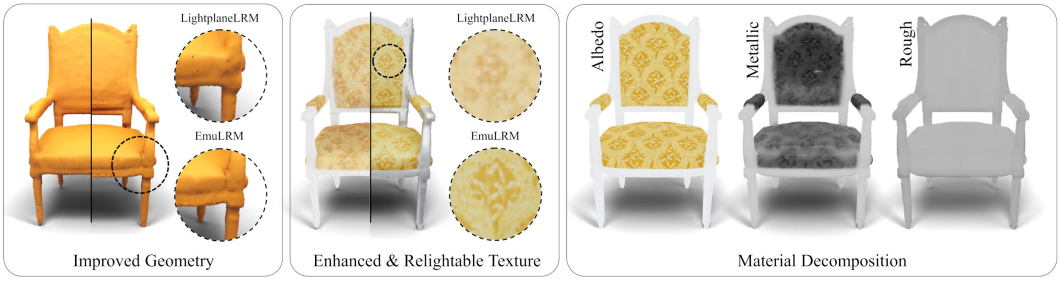


Figure 7: MetaILRM builds upon LightplaneLRM [6], providing improved geometry by employing SDF as a representation, along with direct scalable losses in 3D, improved texture using a UV space texture refiner, and material decomposition by predicting material properties regularized through a novel deferred shading loss.

https://doi.org/10.52202/079017-0303

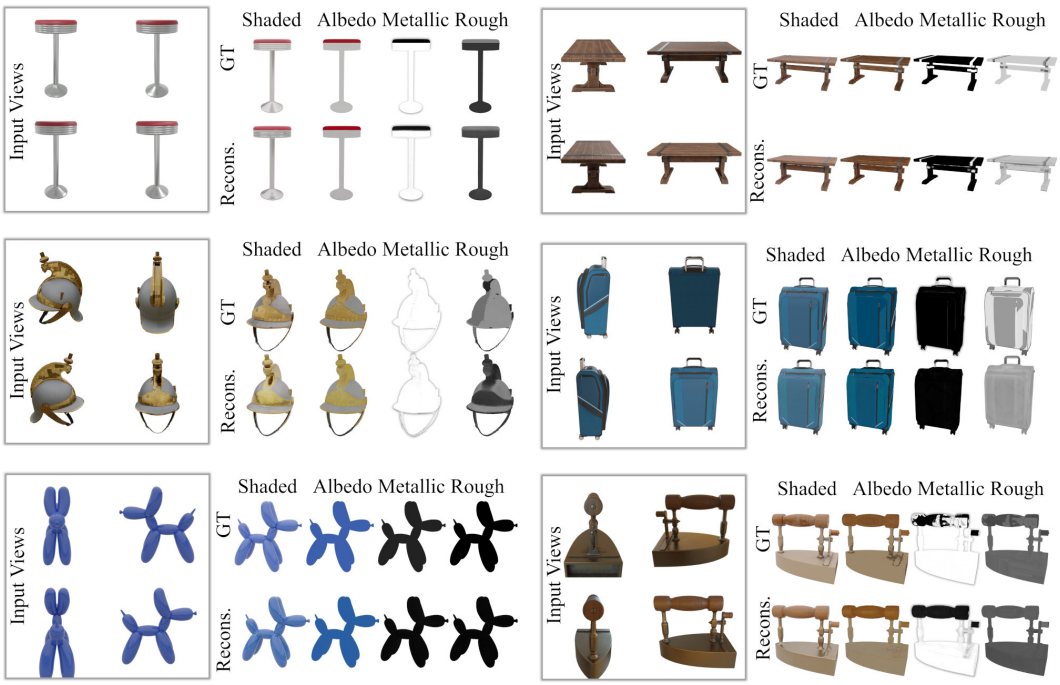


Figure 8: Sparse view reconstruction with intrinsic decomposition. Here the MetaILRM takes 4 shaded views as input and reconstructs the 3D object along with it's albedo, metallic and roughness properties.

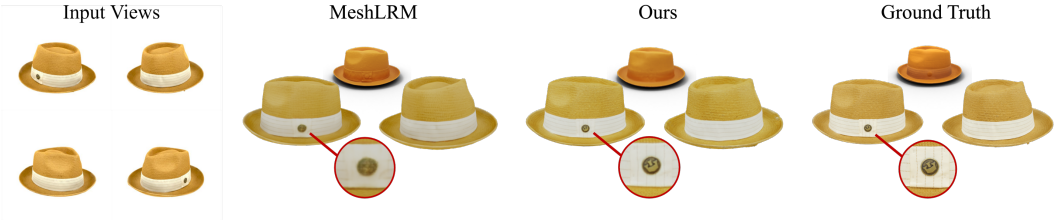


Figure 9: Qualitative comparison on the task of sparse view reconstruction against MeshLRM [97]. Note the higher quality texture detail in our results. Since an open-source implementation of MeshLRM has not yet been released, we compare against the meshes provided on their webpage.

videos of the generated and baseline meshes and were asked to rate them based on 3D shape quality and alignment with the text prompt as shown in Fig. 11. They were asked to consider various factors like identity (whether the object matches what is described in the prompt), texture, existence of Janus problems, and bad geometry (like floaters, disconnected components, etc). 11,080 responses were collected in total, with 5 responses per pair of videos, to eliminate variance in user preference.

A.5 Additional text-to-3D comparisons

While Tab. 2 compared Meta 3D AssetGen's text-to-3D generations to several fast baselines, for completeness, this section includes additional comparisons to significantly slower methods. More specifically, we conduct the same user study as in Tab. 2 but we compare to the "refinement" stages of the industry baselines Meshy v3 and Luma Genie whose asset generation time is 5 and 10 min re-

Table 4: **Win-rate of Meta 3D AssetGen in text-to-3D user study** evaluating visual quality and text alignment of the generated meshes. In addition to Tab. 2, here we compare to slower baselines. Our Meta 3D AssetGen generates a 3D asset within 30 sec.

Method	Visual quality	Text fidelity	Runtime
Meshy v3, refined [85]	77.5 %	80.9 %	300 sec
Luma Genie 1.0, refined (hi-res) [84]	51.2 %	46.3 %	600 sec

https://doi.org/10.52202/079017-0303

Figure 10: Comparison against RichDreamer, Unique3D, the very recently released Stable Fast 3D Mesh, and Luma Stage 2 (Refinement). RichDreamer takes around an hour per mesh, whereas we generate a mesh in less than 30 seconds with significantly better PBR materials. RichDreamer struggles to separate lighting effects from albedo and generates suboptimal geometry. Unique3D also produces inferior geometry compared to ours and cannot generate PBR materials. Stable Fast 3D Mesh predicts only a single value for metallicity and roughness instead of generating a map. It tends to produce suboptimal geometry and flat objects, as seen with the car and the pug respectively. Luma Stage 2 takes around 10 minutes, generates much better textures than Luma Stage 1, but still struggles to separate illumination from albedo, as evidenced by the car hood.

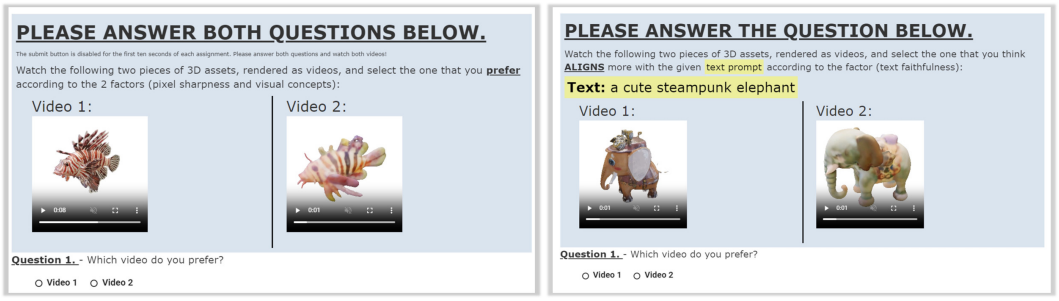


Figure 11: User study interface submitted to a standard crowdsourcing marketplace. Participants are shown videos corresponding to Meta 3D AssetGen and a baseline in a random order, and asked their preference in terms of either quality (left) or faithfulness to the text prompt (right).

https://doi.org/10.52202/079017-0303

spectively. Tab. 4 contains the results of our user-study. Meta 3D AssetGen significantly outperforms Meshy in both text fidelity and visual quality while being 10× faster. Surprisingly, Meta 3D AssetGen is on par with Luma Genie in text fidelity and wins in 40% of cases in visual quality. This is a remarkable result considering Meta 3D AssetGen's 20× better generation time. We show further qualitative comparisons in Fig. 10.

A.6 Additional technical details

A.6.1 Grid Generator

We employ a text-to-image diffusion model pre-trained on billions of images annotated with text [16] and expand its input and output channels by a factor of 2 to support simultaneous generation of shaded appearance and albedo. We finetune the model to predict a grid of four images I_i, $i = 1, \ldots, 4$, in similar fashion to [74, 42] via minimization of the standard diffusion loss. Training spans a total of 2 days, employing 32 A100 GPUs with a total batch size of 128 and a learning rate of 10^{-5}.

A.6.2 MetaILRM

As mentioned in the main paper, MetaILRM is optimized using the direct SDF loss $\mathcal{L}_{\text{sdf}}$, PBR loss $\mathcal{L}_{\text{pbr}}$, deferred shading loss $\mathcal{L}_{\text{def}}$, the binary cross-entropy mask loss $\mathcal{L}_{\text{mask}}$, and the depth-MSE loss $\mathcal{L}_{\text{depth}}$ so the global objective is:

$$\mathcal{L} = 0.5\mathcal{L}_{\text{sdf}} + \mathcal{L}_{\text{pbr}} + 0.5\mathcal{L}_{\text{def}} + 0.1\mathcal{L}_{\text{mask}} + 0.1\mathcal{L}_{\text{depth}}.$$

The texture refiner uses only PBR loss and the deferred shading loss: $\mathcal{L}_{\text{pbr}} + 0.5\mathcal{L}_{\text{def}}$.

In each training batch, we randomly sample 4 views per scene as source input views I_i, and another 4 target views I^{tgt}, into which we render the sdf-field predicted by MetaILRM, or the mesh predicted by the texture refiner. We then evaluate the aforementioned losses in the target views. 3 scenes per GPU are sampled randomly, and we train on 64 GPUS NVIDIA A100 gpus, yielding an effective batch size of $3 \times 4 \times 64 = 768$ images. The total loss has been optimized using Adam [41] with learning rate 10^{-4} for 13K steps.

A.6.3 Deferred shading loss ablation

Figure 12: Generated assets for the prompt: "A cat made of <MATERIAL>". Meta 3D AssetGen predicts various plausible PBR material maps leading to realistic interaction with the environment light (sphere-mapped in the center)

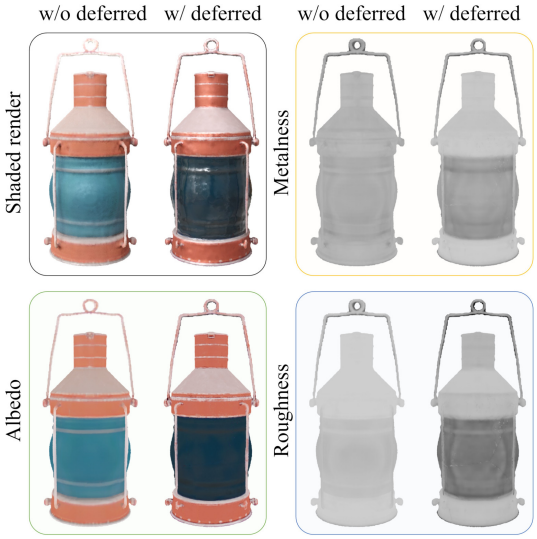


Figure 13: Using a deferred shading loss on rendered channels enhances PBR quality, resulting in more defined metalness and roughness, such as increased metalness in the lantern's metal parts and decreased roughness in its glass parts.

https://doi.org/10.52202/079017-0303

Besides verifying quantitatively the benefits of the deferred shading loss $\mathcal{L}_{\text{def}}$ in Tab. 1, we also provide a qualitative proof in Fig. 13. Specifically, the PBR materials predicted from albedo&shaded channels exhibit better metalness map on the actual metallic parts of the 3D lantern asset.

A.6.4 Direct SDF loss $\mathcal{L}_{\text{sdf}}$

We follow Azinovic *et al.*'s [1] direct SDF supervision for the SDF field. Given a pixel p in an image and the sampled points S_p on the ray corresponding to the pixel, the direct SDF loss is computed as

$$\mathcal{L}_{\text{sdf}}(p) = \mathcal{L}^{\text{tr}}_{\text{sdf}}(p) + 0.01\mathcal{L}^{\text{fs}}_{\text{sdf}}(p). \tag{7}$$

$\mathcal{L}^{\text{fr}}_{\text{sdf}}$ is a 'free-space' objective, which forces the MLP to predict a value of 1 for samples $s \in S^{\text{fs}}_p$ which lie between the camera origin and the truncation region of a surface:

$$\mathcal{L}^{\text{fr}}_{\text{sdf}}(p) = \frac{1}{|S^{\text{fr}}_p|} \sum_{s \in S^{\text{fs}}_p} (D_s - 1)^2 \tag{8}$$

where D_s is the predicted SDF from the MLP. For samples within the truncation region ($s \in S^{\text{tr}}_p$), we apply $\mathcal{L}^{\text{tr}}_{\text{sdf}}$, the signed distance objective of samples close to the surface.

$$\mathcal{L}^{\text{tr}}_{\text{sdf}}(p) = \frac{1}{|S^{\text{tr}}_p|} \sum_{s \in S^{\text{tr}}_p} (D_s - \hat{D}_s)^2 \tag{9}$$

A naïve PyTorch implementation of this is memory intensive, because of the evaluation of $B \times H \times W \times N_{\text{ray}}$ points, where B, H, W, N_{ray} are the number of target images in a batch, the height, width, and the number of points per ray respectively. Therefore, to support large batch sizes, image resolution, and denser point sampling on rays, we implement the direct SDF loss using custom Triton [67] kernels.

A.6.5 Depth loss $\mathcal{L}_{\text{depth}}$

The depth loss $\mathcal{L}_{\text{depth}}$ minimizes the mean-squared error between the rendered depth prediction $\mathcal{R}_{\text{depth}}(\cdot \mid \hat{s}, \pi)$ the ground-truth depth $\mathcal{R}_{\text{depth}}(\cdot \mid M, \pi)$

$$\mathcal{L}_{\text{depth}} = \left\| \mathcal{R}_{\text{depth}}(\cdot \mid \hat{s}, \pi) - \mathcal{R}_{\text{depth}}(\cdot \mid M, \pi) \right\|^2,$$

where $\mathcal{R}_{\text{depth}}(s, \pi)$ is an operator rendering the depth-map of the shape representation s (mesh or an SDF) from the viewpoint π.

A.7 Texture refiner

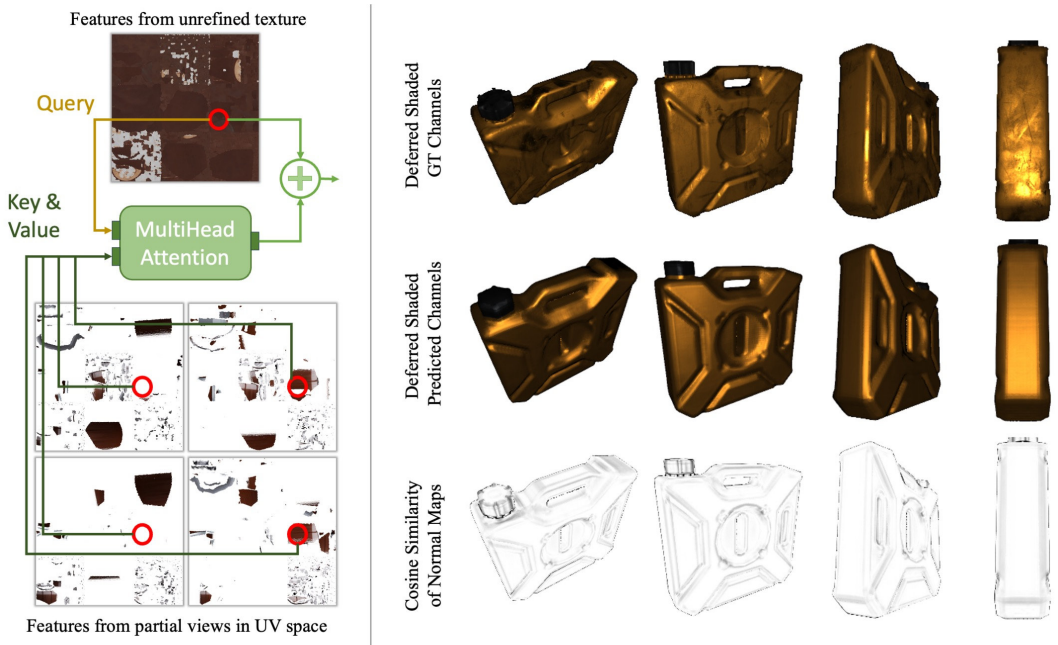
Having described a high-level overview of our texture refiner in Sec. 3.3, here we provide more details.

As mentioned, the texture refiner network Φ accepts $N + 1$ texture images K_i in total. The first input to the network is the augmented texture image $K_0 \in \mathbb{R}^{V \times V \times 11}$ given by:

$$\forall v \in [0, V]^2 : \quad K_0(v) = \begin{cases} k(\boldsymbol{x}_v) \oplus \boldsymbol{n}(\boldsymbol{x}_v) \oplus \boldsymbol{x}_v, & \text{if } v \in \text{Im}(\phi), \\ \mathbf{0}, & \text{otherwise,} \end{cases} \quad \text{where } \boldsymbol{x}_v = \phi(v).$$

The condition $v \in \text{Im}(\phi)$ selects 'valid' UV points that correspond to mesh points; and $\oplus$ denotes channel-wise concatenation, so that $K_0(v)$ is the stack of the 5 PBR parameters $k(\boldsymbol{x}_v)$ from MetaI LRM, normal $\boldsymbol{n}(\boldsymbol{x}_v)$, and the 3D point $\boldsymbol{x}_v$.

In addition to K_0, we input to the network Φ texture images K_i, each extracted by looking up information from the corresponding input view I_i directly (thus sidestepping MetaILRM). As noted above, each valid texture point v corresponds to a unique 3D point $\boldsymbol{x}_v = \phi(v) \in M$ on the mesh, which in turn projects to a pixel $u = \pi_i(\boldsymbol{x}_v)$ in the image I_i. Let $\chi_i(v) \in \{0, 1\}$ be the flag that tells if point $\boldsymbol{x}_v$ is *visible* in image I_i or not. When point $\boldsymbol{x}_v$ is visible in several views, it is best measured in the most frontal one, which is captured by the cosine $\boldsymbol{\omega}_{\text{o}} \cdot \boldsymbol{n}(\boldsymbol{x}_v)$ between the normal $\boldsymbol{n}$ at $\boldsymbol{x}_v$



(a) Cross View Attention Layer

(b) Deferred Shading Loss Example

Figure 14: **(a)** Illustration of Cross-View Attention. Cross-view attention facilitates communication between the UNet branches processing the predicted texture features and the UV space projected input views. This layer blends the predicted texture features with the UV projected input view features based on their match using a multiheaded attention mechanism. **(b)** Example of Deferred Shading Loss Calculation. Deferred shading computes pixel shading using albedo, metalness, roughness, normals, object position, and light source position. We apply it to both the ground truth channels (top) and the predicted channels (middle). The error is calculated as the difference between the two, weighted by the similarity between ground truth normals and predicted normals, to avoid penalizing shading errors due to incorrect normals.

and the ray direction $\boldsymbol{\omega}_v \propto \boldsymbol{x}_0 - \boldsymbol{x}_v$. All this information is packed into additional texture images $K_i \in \mathbb{R}^{V \times V \times (D+1)}$ by setting:

$$\forall v \in [0, V]^2 : \quad K_i(v) = \begin{cases} I_i(\pi_i(\boldsymbol{x}_v)) \oplus (\boldsymbol{\omega}_v \cdot \boldsymbol{n}(\boldsymbol{x}_v)), & \text{if } v \in \mathrm{Im}(\phi) \text{ and } \chi_i(v) = 1, \\ \mathbf{0}, & \text{otherwise.} \end{cases}$$

The texture network Φ is a U-Net that takes as input the texture augmented texture image K_0 and outputs the final enhanced texture $K \in \mathbb{R}^{V \times V \times 5}$. This network also fuses information from the view-specific texture images K_i. The goal is to select, for each UV point v, which of the N input views provides the best information. This is achieved via cross-attention. Specifically, each K_i is processed in parallel by another U-Net, and the first queries information across all the others via multi-head cross attention. In Fig. 14 (a), we provide an illustration of the latter cross-view attention layer.

A.8 Physically-Based Rendering: Radiance, BRDFs, and models

We briefly summarise key notion of radiometry and standard BRDF models, and then provide a precise expression of the BRDF model used in Meta 3D AssetGen.

A.8.1 Radiance

The *radiant flux* Φ is the electromagnetic power flowing through a particular surface $A \subset \mathbb{R}^3$ oriented by the unit normal $\boldsymbol{n}$. The *radiance* $L(p, \boldsymbol{\omega})$ is the radiant flux density at p towards a particular direction $\boldsymbol{\omega}$ per unit *orthogonal* area $dA_\perp$ and per unit solid angle $d\Omega$. The unit vector $\boldsymbol{\omega}$ points at the direction of propagation of the flux.

https://doi.org/10.52202/079017-0303

The flux density is measured with respect to an area which is orthogonal to the direction of propagation $\boldsymbol{\omega}$. In fact, the energy flow is the same through all areas that cut the same 'tube of flux'; the specific area is irrelevant and not a property of the radiation. This dependency is removed by considering the normalized area $dA_\perp$, which is orthogonal to the direction of the flux.

Because the radiance is expressed in units of orthogonal area $dA_\perp$, in order to compute the flux through the surface patch dA, which may not be orthogonal, we must account for the *foreshortening factor*, which relates the areas dA and $dA_\perp$:

$$dA_\perp = |\langle \boldsymbol{n}, \boldsymbol{\omega} \rangle| \, dA.$$

With this, the flux that passes through dA towards direction $\boldsymbol{\omega}$ in the solid angle $d\Omega$ is

$$d\Phi = L(p, \boldsymbol{\omega}) \, |\langle \boldsymbol{n}, \boldsymbol{\omega} \rangle| \, dA \, d\Omega.$$

Note that $d\Phi$ depends on both $\boldsymbol{\omega}$ and $\boldsymbol{n}$ whereas L only depends on $\boldsymbol{\omega}$. This reinforces the notion that $\boldsymbol{n}$ is a property of the surface, not of the radiation.

A.9 Reflectance models

Let p be a point on a surface A that separates two media and let dA be a surface patch sitting at p. Let $\boldsymbol{n}$ be the normal at this point. We now consider the case where A separates air or empty space from an opaque object, with $\boldsymbol{n}$ pointing towards the outside of this object.

Let $\boldsymbol{\omega}$ be an orientation on the same side of the surface as $\boldsymbol{n}$, i.e., such that $\langle \boldsymbol{n}, \boldsymbol{\omega} \rangle \geq 0$. From the viewpoint of the object, we interpret $L(p, \boldsymbol{\omega})$ as *outgoing radiant flux* and $L(p, -\boldsymbol{\omega})$ as *incoming radiant flux*. These two quantities are related by the *Bidirectional Reflectance Distribution Function* (BRDF) f, defined such that:

$$\frac{dL(p, \boldsymbol{\omega}_\mathrm{o})}{d\Omega_\mathrm{i}} = f(p, \boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}) \, \langle \boldsymbol{n}, \boldsymbol{\omega}_\mathrm{i} \rangle L(p, -\boldsymbol{\omega}_\mathrm{i}). \tag{10}$$

In this definition, for convenience both $\boldsymbol{\omega}_\mathrm{i}$ and $\boldsymbol{\omega}_\mathrm{o}$ are taken on the same side as $\boldsymbol{n}$ (hence the negative sign in front of $\boldsymbol{\omega}_\mathrm{i}$). A useful consequence is that we do not need to take the absolute value of the inner product $\langle \boldsymbol{n}, \boldsymbol{\omega}_\mathrm{i} \rangle$ as this is positive by definition.

The BRDF thus takes the radiation receives from direction $-\boldsymbol{\omega}_\mathrm{i}$ and distributes it along various outgoing directions $\boldsymbol{\omega}_\mathrm{o}$. Integrating over all incoming directions, gives use the overall radiation reflected towards $\boldsymbol{\omega}_\mathrm{o}$:

$$L(p, \boldsymbol{\omega}_\mathrm{o}) = \int_{H(\boldsymbol{n})} \frac{dL(p, \boldsymbol{\omega}_\mathrm{o})}{d\Omega_\mathrm{i}} \, d\boldsymbol{\omega}_\mathrm{i} = \int_{H(\boldsymbol{n})} f(p, \boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}) \, \langle \boldsymbol{n}, \boldsymbol{\omega}_\mathrm{i} \rangle L(p, -\boldsymbol{\omega}_\mathrm{i}) \, d\Omega_\mathrm{i}. \tag{11}$$

where $H(\boldsymbol{n}) = \{\boldsymbol{\omega} : \langle \boldsymbol{n}, \boldsymbol{\omega} \rangle \geq 0\}$ is the hemisphere. Next, we provide common basic models for the BRDF function in PBR.

A.9.1 Diffuse reflectance

In diffuse reflectance, the radiation is absorbed by the material, internally scattered in random directions, and output again to give rise to a uniform distribution. Namely, the *diffuse BRDF* is:

$$f(p, \boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}) = \frac{R}{\pi}$$

where $0 \leq R \leq 1$ is the fraction of power reflected by the diffusion process. The $1/\pi$ factor ensures that the total energy is conserved when $R = 1$.

A.10 Specular reflectance

The reflection for a perfectly flat interface between two media at p is *specular*: the incoming light radiation $-\boldsymbol{\omega}_\mathrm{i}$ is partially reflected in the specular direction $\boldsymbol{\omega}_\mathrm{o} = r(\boldsymbol{n}, \boldsymbol{\omega}_\mathrm{i}) = 2\boldsymbol{n}\langle \boldsymbol{n}, \boldsymbol{\omega}_\mathrm{i} \rangle - \boldsymbol{\omega}_\mathrm{i}$, and partially transmitted. This phenomena is characterised by *Fresnel's equations*, which are derived from Maxwell's equations, utilising continuity conditions for the electromagnetic field at the interface between the two media. Fresnel's equations describe the planar radiation in full, including its polarisation (in the most general case, using phasors, and thus complex numbers). For graphics, we

assume that light is unpolarized, so we only calculate the power. The fraction of power reflected is given by Fresnel's coefficient (using Schlick's approximation [72]):

$$F(\langle \boldsymbol{n}, \boldsymbol{\omega}_\mathrm{i} \rangle) = F_0 + (1 - F_0)\, |\langle \boldsymbol{n}, \boldsymbol{\omega}_\mathrm{i} \rangle|^5, \quad F_0 = \left(\frac{\hat{n}_1 - \hat{n}_2}{\hat{n}_1 + \hat{n}_2} \right)^2.$$

Here $\hat{n}_1$ and $\hat{n}_2$ are the indices of reflectivity of the two media, respectively. This equation is valid for dielectrics (non-metallic objects), but also used as an approximation for metals by tweaking F_0.

In order to write this relation as a BRDF, we write:

$$L(p, \boldsymbol{\omega}_\mathrm{o}) = \int_{H(\boldsymbol{n})} f(p, \boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})\, \langle \boldsymbol{n}, \boldsymbol{\omega}_\mathrm{i} \rangle L(p, -\boldsymbol{\omega}_\mathrm{i})\, d\Omega_\mathrm{i} = R(\langle \boldsymbol{n}, r(\boldsymbol{n}, \boldsymbol{\omega}_\mathrm{o}) \rangle) L(p, -r(\boldsymbol{n}, \boldsymbol{\omega}_\mathrm{o})).$$

Hence, the BRDF must be a delta function centered at $\boldsymbol{\omega}_\mathrm{i}^* = r(\boldsymbol{n}, \boldsymbol{\omega}_\mathrm{o})$:

$$f(p, \boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}) = \frac{F(\langle \boldsymbol{n}, \boldsymbol{\omega}_\mathrm{o} \rangle)}{\langle \boldsymbol{n}, \boldsymbol{\omega}_\mathrm{o} \rangle} \delta_{r(\boldsymbol{n}, \boldsymbol{\omega}_\mathrm{o})}(\boldsymbol{\omega}_\mathrm{i}) \tag{12}$$

where we used the fact that $\langle \boldsymbol{n}, r(\boldsymbol{n}, \boldsymbol{\omega}_\mathrm{o}) \rangle = \langle \boldsymbol{n}, \boldsymbol{\omega}_\mathrm{o} \rangle$ and where $\delta_{r(\boldsymbol{n}, \boldsymbol{\omega}_\mathrm{o})}$ is the delta distribution centered at $r(\boldsymbol{n}, \boldsymbol{\omega}_\mathrm{o})$.

A.11 Microfacet models

Rough surfaces can be though of as a collection of randomly-oriented flat microfacets, each reflecting light in a specular fashion. Consider a point p on a surface and incoming and outgoing radiation directions $\boldsymbol{\omega}_\mathrm{i}$ and $\boldsymbol{\omega}_\mathrm{o}$. If the point contains a microfacet that enables light to reflect from $\boldsymbol{\omega}_\mathrm{i}$ to $\boldsymbol{\omega}_\mathrm{o}$, then the normal of the microfacet must be $\boldsymbol{m} \propto \boldsymbol{\omega}_\mathrm{i} + \boldsymbol{\omega}_\mathrm{o}$. If the microfacet is oriented elsewhere, then no light flows in the direction $\boldsymbol{\omega}_\mathrm{o}$. Hence, we can write

$$\boldsymbol{m} = h(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}) = \frac{\boldsymbol{\omega}_\mathrm{i} + \boldsymbol{\omega}_\mathrm{o}}{|\boldsymbol{\omega}_\mathrm{i} + \boldsymbol{\omega}_\mathrm{o}|}$$

as a function of the incoming and outgoing radiation. This is called *half vector* as it sits in between the two vectors $\boldsymbol{\omega}_\mathrm{i}$ and $\boldsymbol{\omega}_\mathrm{o}$.

Now we wish to derive the macro BRDF $f(p, \boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})$ from the micro BRDF $F(p, \boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o} | \boldsymbol{m})$, where we have emphasized the fact that the BRDF is oriented relative to the microfacet normal $\boldsymbol{m}$. A short calculation [90] shows that:

$$f(p, \boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}) = f_m(p, \boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o} | \boldsymbol{m}) \frac{\langle \boldsymbol{m}, \boldsymbol{\omega}_\mathrm{i} \rangle}{\langle \boldsymbol{n}, \boldsymbol{\omega}_\mathrm{i} \rangle} \frac{\langle \boldsymbol{m}, \boldsymbol{\omega}_\mathrm{o} \rangle}{\langle \boldsymbol{n}, \boldsymbol{\omega}_\mathrm{o} \rangle} \frac{1}{\langle \boldsymbol{m}, \boldsymbol{n} \rangle}.$$

In practice, there is a distribution over possible surface normals $\boldsymbol{m}$, characterised by the *microfacet distribution function* $D(\boldsymbol{m} | \boldsymbol{n})$. The latter is defined such that $D(\boldsymbol{m} | \boldsymbol{n})\, dA\, d\Omega_m$ is the total area of the microfacets within patch dA of the macrosurface with orientation in $d\Omega_m$. In practice, only part of the microfacet is visible and illuminated, depending on the interaction with other facets. This is accounted for by the *shadowing-masking* function $G(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}, \boldsymbol{m}, \boldsymbol{n}) \in [0, 1]$. The expected reflectance is thus:

$$f(p, \boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}) = \int_{H(\boldsymbol{n})} \frac{\langle \boldsymbol{m}, \boldsymbol{\omega}_\mathrm{i} \rangle}{\langle \boldsymbol{n}, \boldsymbol{\omega}_\mathrm{i} \rangle} \frac{\langle \boldsymbol{m}, \boldsymbol{\omega}_\mathrm{o} \rangle}{\langle \boldsymbol{n}, \boldsymbol{\omega}_\mathrm{o} \rangle} f_m(p, \boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o} | \boldsymbol{m})\, D(\boldsymbol{m} | \boldsymbol{n})\, G(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}, \boldsymbol{m}, \boldsymbol{n})\, d\Omega_m.$$

Plugging Eq. (12) in the value for the mirror-like reflectance f_m for each microfacet, we get [90]:

$$f(p, \boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}) = \frac{F(\langle \boldsymbol{h}, \boldsymbol{\omega}_\mathrm{o} \rangle) D(\boldsymbol{h} | \boldsymbol{n})\, G(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}, \boldsymbol{h}, \boldsymbol{n})}{4 \langle \boldsymbol{n}, \boldsymbol{\omega}_\mathrm{i} \rangle \langle \boldsymbol{n}, \boldsymbol{\omega}_\mathrm{o} \rangle} \quad \text{where } \boldsymbol{h} = h(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}).$$

A.12 Standard microfacet models

Here, we discuss common choices for the functions D and G in standard PBR models.

https://doi.org/10.52202/079017-0303

The Torrance-Sparrow model. The oldest such model is due to Torrance and Sparrow [87]. They simply assume a Gaussian model for the microfacet distribution function:

$$D(\boldsymbol{m}|\boldsymbol{n}) = b \exp\left(-\alpha^2\theta\right), \quad \text{where } \cos\theta = \langle \boldsymbol{m}, \boldsymbol{n}\rangle,$$

and b is a suitable normalization constant. For the shadowing-masking function they pick:

$$G(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}, \boldsymbol{m}, \boldsymbol{n}) = \min\left\{1, \frac{2\langle \boldsymbol{m}, \boldsymbol{n}\rangle\langle \boldsymbol{n}, \boldsymbol{\omega}_\mathrm{i}\rangle}{\langle \boldsymbol{m}, \boldsymbol{\omega}_\mathrm{i}\rangle}, \frac{2\langle \boldsymbol{m}, \boldsymbol{n}\rangle\langle \boldsymbol{n}, \boldsymbol{\omega}_\mathrm{o}\rangle}{\langle \boldsymbol{m}, \boldsymbol{\omega}_\mathrm{o}\rangle}\right\}.$$

This function has a simple geometric derivation under a basic geometric model of the microfacets.

The Beckmann-Spizzichino-Smith model. Beckmann and Spizzichino [2] suggested the model:

$$D(\boldsymbol{m}|\boldsymbol{n}) = \frac{1}{\pi\alpha^2\cos^4\theta} e^{-\frac{\tan^2\theta}{\alpha^2}} \quad \text{where } \cos\theta = \langle \boldsymbol{m}, \boldsymbol{n}\rangle.$$

Smith [77] noted that the shadowing-masking function should be derived from the microfacet distribution function, which describes the micro-geometry of the surface. They also proposed a factorized model $G(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}, \boldsymbol{m}, \boldsymbol{n}) = G_1(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{n})G_1(\boldsymbol{\omega}_\mathrm{o}, \boldsymbol{n})$. For Beckmann's distribution, the Smith shadowing-masking function is given by:

$$G_1(\boldsymbol{\omega}, \boldsymbol{n}) = \frac{2}{1 + \mathrm{erf}(a) + \frac{1}{a\sqrt{\pi}}e^{-a^2}} \quad \text{where } a = \frac{1}{\alpha\tan\theta_{\boldsymbol{\omega}}} \text{ and } \cos\theta_{\boldsymbol{\omega}} = \langle \boldsymbol{n}, \boldsymbol{\omega}\rangle.$$

The GGX model. The GGX model by [90] is a variant of the Beckmann model, with slightly different microfacet distribution and shadowing-masking function:

$$D(\boldsymbol{m}|\boldsymbol{n}) = \frac{\alpha^2}{\pi\cos^4\theta(\alpha^2 + \tan^2\theta)^2}, \qquad G_1(\boldsymbol{\omega}, \boldsymbol{n}) = \frac{2}{1 + \sqrt{1 + \alpha^2\tan^2\theta_{\boldsymbol{\omega}}}}, \tag{13}$$

where $\cos\theta = \langle \boldsymbol{m}, \boldsymbol{n}\rangle$ and $\cos\theta_{\boldsymbol{\omega}} = \langle \boldsymbol{n}, \boldsymbol{\omega}\rangle$.

A.12.1 The BRDF model used in Meta 3D AssetGen

The BRDF model used in our paper combines diffuse and GGX BRDFs:

$$f(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}|k(\boldsymbol{x}), \boldsymbol{n}) = \frac{R}{\pi} + \frac{F(\boldsymbol{h}|\boldsymbol{n})D(\boldsymbol{h}|\boldsymbol{n})G_1(\boldsymbol{n}, \boldsymbol{\omega}_\mathrm{i})G_1(\boldsymbol{n}, \boldsymbol{\omega}_\mathrm{o})}{4(\boldsymbol{n}\cdot\boldsymbol{\omega}_\mathrm{i})(\boldsymbol{n}\cdot\boldsymbol{\omega}_\mathrm{o})}.$$

The first term is the *diffuse component* (Lambertian reflection), where $R \in [0, 1]$ is the fraction of light power reflected by diffusion. The second term in the *specular component*, where F is Schlick's approximation [72] $F(\boldsymbol{\omega}_\mathrm{i}|\boldsymbol{n}) = F_0 + (1 - F_0)(1 - \boldsymbol{n}\cdot\boldsymbol{\omega}_\mathrm{i})^5$ of Fresnel's reflectance and where $F_0 \in [0, 1]$ is the Fresnel coefficient at normal incidence. The unit vector $\boldsymbol{h} \propto \boldsymbol{\omega}_\mathrm{i} + \boldsymbol{\omega}_\mathrm{o}$ is the *half vector*, which is the orientation needed to reflect $\boldsymbol{\omega}_\mathrm{i}$ into $\boldsymbol{\omega}_\mathrm{o}$ by the rough material's microfacets (generally different from but averaging to $\boldsymbol{n}$). The function D and G_1 are the microfacet distribution function and the shadowing-masking function given by:

$$D(\boldsymbol{m}|\boldsymbol{n}) = \frac{\alpha^2}{\pi((\boldsymbol{m}\cdot\boldsymbol{n})^2(\alpha^2 - 1) + 1)^2}, \quad G_1(\boldsymbol{n}, \boldsymbol{\omega}) = \frac{2(\boldsymbol{n}\cdot\boldsymbol{\omega})}{\boldsymbol{n}\cdot\boldsymbol{\omega} + \sqrt{(\boldsymbol{n}\cdot\boldsymbol{\omega})^2(\alpha^2 - 1) + \alpha^2}}.$$

These are the same as Eq. (13) with the trigonometric functions expanded in terms of dot products. In this model, the reflectance R and reflectance at normal incidence F_0 are RGB triplets. The specular highlight color F_0 is approximately white (equal to $\mathbf{1}$) for dielectrics, and colored for metals; furthermore, metals have no diffuse component ($R = \mathbf{0}$). Thus, we introduce the parameter *metalness* $\gamma \in [0, 1]$ and the *base color* $\rho_0 \in [0, 1]^3$ and define:

$$R = \rho_0(1 - \gamma), \qquad F_0 = \mathbf{1}(1 - \gamma) + \rho_0\gamma.$$

In this manner, the albedo is used as diffuse color or specular color depending on whether the material is a dielectric or a metal. In the paper, we call the parameter ρ_0 *albedo* as it is a better known concept; however, in this model ρ_0 is the albedo only when $\gamma = 0$, i.e., when the object is a perfect dielectric.

The BRDF is thus fully described by the albedo ρ_0, the roughness α, and the metalness γ, for a total of five scalar parameters. Hence, the LRM predicts the triplet $k(\boldsymbol{x}) = (\rho_0, \gamma, \alpha)$ at each 3D point $\boldsymbol{x}$.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See Sec. 1 and the abstract.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See App. A.2

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

https://doi.org/10.52202/079017-0303

Justification: No theoretical results present.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Please see Secs. 3 and 4 and Apps. A.6.2 and A.7 for all technical details required for reproducing the paper results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We do not currently plan to open-source the method or training data.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Please see Secs. 3 and 4 and Apps. A.6.2 and A.7 for all technical details required for reproducing the paper results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Reported metrics have negligible or zero variance.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

https://doi.org/10.52202/079017-0303

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Sec. 4 and App. A.6.2

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The following is our statement addressing concerns about training data raised during the review process. The data used in this paper was not obtained by scraping the internet. The data was purchased from a well-respected and widely-known vendor of 3D graphic assets. We acquired a data license that explicitly allows use of the models in machine-learning applications, including all applications in this paper. We follow the terms and conditions of this license scrupulously, also based on internal legal advice.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes] ,

    Justification: See App. A.1

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Data or models are not released

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use an internal dataset of artist-created meshes whose use has been approved by a professional legal team.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

https://doi.org/10.52202/079017-0303

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: See App. A.4.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [Yes]

    Justification: The internal legal and privacy audit has approved our user study.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-0303