VIDEORNG: RELIGHTABLE NEURAL GAUSSIANS FROM GENERATIVE VIDEO MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generating realistic 3D assets remains constrained by classical representations: surface meshes and compact physically based materials struggle to express objects whose appearance arises from fine geometry or complex light transport, while existing Gaussian-splatting generators typically bake fixed illumination into appearance. We introduce VideoRNG, a two-stage method that generates complete, relightable 3D assets from a text prompt or a single image without prescribing a mesh or analytic material model. A video generator fine-tuned from a pretrained video diffusion backbone first produces a structured RGBA turntable sequence under a known pose, camera, and illumination protocol. A feed-forward large reconstruction model initialized from a pretrained video diffusion backbone then maps the video-VAE latent frames of selected turntable frames directly to millions of anisotropic Gaussians with learned neural appearance features; all of its parameters are fine-tuned for reconstruction. A shared directional decoder and stochastic Gaussian ray tracer turn this representation, which we call ray-traced Relightable Neural Gaussians (rtRNG), into radiance under novel viewpoints and illumination. To preserve high-frequency appearance while learning geometry, we introduce depth-anchored appearance training, and we make million-Gaussian reconstruction practical through foreground token selection and cost-aware distributed sampling. VideoRNG generates complex assets such as fur, feathers, and spider webs that can be interactively rendered, relit, and integrated into scenes. Our anonymous results page: `https://videorng.github.io/`

1 INTRODUCTION

Creating complex, realistic 3D objects is essential to populating convincing virtual worlds, but shape modeling and material authoring remain labor-intensive and require substantial expertise, which limits the scale and diversity of virtual environments that can be built within a given budget. Generative models may offer a way forward: recent text- and image-conditioned models can rapidly synthesize high-quality 3D assets (Xiang et al., 2025a;b; Lin et al., 2025). Yet their output remains limited by the representations they produce.

Existing systems generally produce either surface meshes with physically based rendering (PBR) materials or Gaussian radiance fields that bake specific illumination into appearance. PBR materials typically combine diffuse and microfacet specular lobes controlled by textures such as base color, roughness, and metallicity (Walter et al., 2007; McAuley et al., 2012). This representation is compact, interpretable, and well supported by existing tools, but a poor fit for objects whose appearance arises from fine geometry or complex light transport, such as fur, feathers, moss, dense foliage, and spider webs. In contrast, image and video models need not explain their output through a prescribed geometric primitive or analytic reflectance model, and can readily depict such visual complexity.

Two lines of work suggest a route beyond this limitation. Gaussian representations can describe geometry too awkward for conventional surface meshes (Kerbl et al., 2023; Moenne-Loccoz et al., 2024), and can be predicted directly from sparse images (Zhang et al., 2024), but their stored appearance is commonly entangled with fixed illumination. Neural appearance models instead learn complex, relightable material response without committing to an analytic BRDF (Mullia et al., 2024; Wu et al., 2026). However, these methods assume that the underlying mesh geometry is known. Relightable Neural Gaussians (RNG) combine the two ideas by attaching

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



Figure 1: VideoRNG generates relightable 3D assets from a text prompt or an image. Here we show rtRNG assets rendered together in a single scene using a full path tracer: a fur coat, pyrite crystal cluster, moth, spider web, ostrich feather, robot figure, and honeycomb. Rendering takes approximately 50 ms per frame at 1280 × 720 resolution and one sample per pixel (1 spp), using per-asset decoders with hidden width 64 on an NVIDIA RTX A6000 GPU. All assets are generated from text prompts except the robot, which is reconstructed from synthetically rendered images. Right: close-ups of the feather edge, moth, spider web, and fur, corresponding to the red boxes.

neural directional appearance to Gaussian geometry for relighting (Fan et al., 2025). These methods reconstruct a supplied object rather than generate one from a text or image prompt; VideoNeuMat (Xue et al., 2026) demonstrates that a generative video prior can supply observations for neural-material extraction, but considers planar material samples rather than complete 3D objects.

This raises a natural question: can we extract complete, relightable 3D assets—including geometry and materials outside the practical reach of classical representations—from the rich outputs of a generative video model?

We introduce **VideoRNG**, a two-stage method for generating relightable neural 3D assets from a text prompt or a single image (Figure 2). The first is a video generator fine-tuned to produce an RGBA turntable sequence under a prescribed object-pose, camera, and illumination protocol. The second is a feed-forward large reconstruction model (LRM) initialized from a pretrained video backbone and fine-tuned for reconstruction; it maps the video-VAE latent frames of selected turntable frames directly to millions of anisotropic Gaussians in a single forward pass. Each Gaussian stores geometry and a learned appearance feature rather than a fixed color or PBR parameter set. A shared neural decoder conditions that feature on the incoming light and outgoing view directions to predict radiance. Forward rendering and differentiable training use stochastic Gaussian ray tracing to account for visibility and shadows (Sun et al., 2025; Xu et al., 2026). We call this combination of RNG's neural directional appearance and stochastic Gaussian ray tracing **rtRNG**: ray-traced Relightable Neural Gaussians. Its appearance feature space and decoder are learned jointly with the reconstructor from rendered images, without requiring a dataset of Gaussian assets as supervision. After training, the shared decoder can be distilled once into a smaller network used by every asset. The assets can be interactively ray-traced from novel cameras, relit, and inserted into a scene with full global illumination (Figure 1); compared with traditional Gaussian splatting, ray tracing has multiple benefits, including clean scene insertion and arbitrary shadow computation under any lighting.

Moving from VideoNeuMat's planar material samples to complete 3D objects introduces substantial reconstruction challenges. We emit one Gaussian along the camera ray of every foreground pixel, allowing the model to preserve fine and disconnected structures without first reducing them to a mesh. To make this high-resolution representation tractable, we remove background tokens before the transformer and group similarly sized objects during distributed training to avoid GPU idling due to load imbalance in a multi-GPU setting. We further observe that joint geometry and appearance learning produces soft appearance, motivating a fixed spatial reference for learning fine material detail. Our depth-anchored objective places Gaussian centers at ground truth depths during training while supervising the depth predictor for inference. Together, these choices let VideoRNG represent

objects such as furry coats, fuzzy dandelions, and delicate spider webs while retaining view- and light-dependent appearance. Our contributions are:

- We introduce VideoRNG, a two-stage pipeline that extracts complete relightable 3D assets from text- or image-conditioned generative video, without constraining their geometry to a mesh or their appearance to a prescribed PBR material model (Section 3).
- We initialize a foreground-sparse LRM from a pretrained generative video DiT and fine-tune all of its parameters to directly predict Gaussian geometry and neural directional appearance at 1024^2 resolution. Our efficient training scheme supports variable-length, million-Gaussian outputs, and learns the neural appearance basis from rendered images without requiring a dataset of Gaussian assets as supervision (Section 3.2; Appendix C.5).
- We propose depth-anchored appearance learning: during training, we place Gaussian centers at ground truth depths for rendering and supervise predicted depths with a separate depth loss. Rendering losses optimize the remaining Gaussian attributes and the appearance decoder; inference uses predicted depths (Section 3.3).

Section 4 evaluates generation, reconstruction from ground truth views, and relighting; the appendices compare depth-training strategies and provide detailed protocols and additional comparisons.

2 Related Work

Text- and image-conditioned 3D generation. DreamFusion (Poole et al., 2023) optimizes a 3D representation via score distillation from a 2D image prior, while MVDream (Shi et al., 2024) provides a multiview-consistent prior. Feed-forward systems pair generated sparse views with large reconstruction models such as LRM, Instant3D, InstantMesh, LGM, GRM, and GS-LRM (Hong et al., 2024; Li et al., 2024; Xu et al., 2024a; Tang et al., 2024; Xu et al., 2024b; Zhang et al., 2024). More recent generators, including TRELLIS, TRELLIS.2, and DiffSplat, directly produce structured 3D latents or Gaussian assets (Xiang et al., 2025b;a; Lin et al., 2025). LSRM (Li et al., 2026) reconstructs meshes from posed views, with an inverse-rendering extension that predicts BRDF materials. These methods improve speed and fidelity across several output representations, but none provides learned relightable neural appearance for objects with complex light transport.

Video and multiview diffusion as 3D priors. Multiview diffusion models generate consistent observations for optimization or feed-forward reconstruction (Shi et al., 2024; Li et al., 2024), and SV3D adapts latent video diffusion to orbital novel-view synthesis and 3D generation from one image (Voleti et al., 2024). Gen3R couples a pretrained video diffusion model with a VGGT-derived geometry latent space to jointly generate images, point clouds, depth, and cameras (Huang et al., 2026). Diff4Splat pairs a video diffusion model with a latent dynamic reconstruction model to predict explicit deformable Gaussian fields for 4D scene generation (Pan et al., 2026). VideoNeuMat extracts planar neural materials from generated video (Xue et al., 2026). VideoRNG uses video diffusion both to generate structured observations and to initialize a separate reconstruction backbone, which is fine-tuned with its denoising head replaced by a Gaussian prediction head to produce an explicit, renderable asset; this extends VideoNeuMat's planar materials to complete static objects with relightable neural appearance.

Neural appearance representations and generation. Neural materials replace restricted analytic reflectance models with learned directional appearance, supporting effects such as parallax, self-shadowing, and layered scattering (Kuznetsov et al., 2021; Xue et al., 2024; Zeltner et al., 2024); RNA and 8DNA extend them to complete relightable assets with global light transport but assume known surface geometry (Mullia et al., 2024; Wu et al., 2026). Generative Neural Materials synthesizes planar neural materials from image or text conditioning in a shared feature basis (Raghavan et al., 2025). VideoRNG learns its appearance basis jointly with a geometry-predicting LRM from rendered images, without a dataset of pre-existing rtRNG assets.

Relightable Gaussian assets and generators. RNG attaches neural directional appearance to Gaussian geometry, relighting objects with hard surfaces and soft boundaries without an analytic shading model, but fits each object independently (Fan et al., 2025). Large Material Gaussian Model and Luce generate relightable Gaussians by predicting explicit PBR quantities such as albedo, roughness, metallicity, and normals (Ye et al., 2025; Singh et al., 2026). RelitLRM generates relit Gaussian appearance for a target environment map rather than relightable assets (Zhang et al., 2025a).

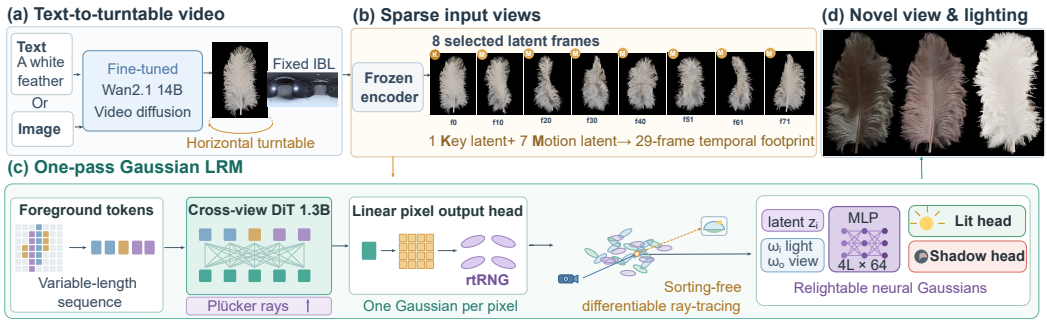

Figure 2: VideoRNG inference pipeline. (a) A fine-tuned video generator produces an 81-frame turntable. (b) 29 uniformly spaced frames are encoded into eight VAE latent frames. (c) The reconstruction DiT processes foreground tokens and predicts one Gaussian per foreground pixel, including a neural appearance feature. A shared directional MLP decodes these features during rendering. (d) The resulting asset supports novel views and lights through stochastic Gaussian ray tracing.

The concurrent work F-RNG has similar goals to ours but different methods: it reconstructs neural Gaussian assets with frozen relighting priors and shades blended features using rasterization (Fu et al., 2026). The above works do not currently ship code or models for comparison.

Gaussian rendering. 3D Gaussian Splatting rasterizes anisotropic Gaussians efficiently (Kerbl et al., 2023), 3D Gaussian Ray Tracing adds primary and secondary rays (Moenne-Loccoz et al., 2024), and sorting-free stochastic ray tracing makes transparent Gaussian clouds practical inside conventional path tracers (Sun et al., 2025), with a differentiable extension that supports reconstruction and relightable per-Gaussian shading with traced visibility (Xu et al., 2026). rtRNG builds on these renderers for interactive rendering, shadow rays, and integration with conventional geometry.

3 Method

VideoRNG reconstructs relightable assets from text or image (Figure 2). The video generator produces a turntable under the prescribed camera trajectory and studio illumination from a text prompt or a single image (a). We encode 29 frames from this standardized sequence into 8 video-VAE latent frames (b), then pass their foreground tokens to a reconstruction model that predicts Gaussian geometry and neural appearance in one forward pass (c). The resulting asset supports novel-view rendering and relighting (d). The inference pipeline is

$$\text{Text/Image} \xrightarrow{\text{Finetuned video generator}} \text{Turntable observations} \xrightarrow{\text{Gaussian LRM}} \text{rtRNG asset.} \quad (1)$$

Sections 3.1–3.4 describe observation generation, reconstruction, training, and interactive rendering under novel views and lights, respectively. Appendix C.5 details the sampling procedure and computational cost.

3.1 Stage 1: Generating Structured Observations

Generation protocol. We fine-tune a pretrained latent video diffusion model (Wan Team, 2025) with its flow-matching objective on ∼22K captioned turntable clips rendered from textured assets. Given a text prompt or a photograph from an arbitrary view that clearly reveals the front, it produces F=81 turntable frames at 1024×1024 resolution. The object completes one rotation about its vertical axis in front of a fixed camera at zero elevation, under studio environment lighting fixed in world coordinates. We selected this lighting for its neutral white balance and clear but not overly sharp specular highlights. Rotation changes both the viewing and illumination directions relative to the object, exposing view-dependent and light-dependent appearance. Joint sequence generation uses the video prior to model cross-frame consistency. The standard variant fine-tunes the pretrained DiT directly; the alpha-capable variant uses the Wan-Alpha VAE and merges its DoRA adapters into the DiT before full fine-tuning (Dong et al., 2026). Generated alpha separates foreground from background. Appendix C.1 gives the camera calibration and frame selection.

Latent observations. We select 29 uniformly spaced frames from the turntable, starting at the front view, and encode them with the causal video VAE (temporal compression four, spatial compression eight) into $V{=}8$ latent frames of $16 \times 128 \times 128$. One keyframe per latent defines the camera rays used to position the emitted Gaussians, and alpha-derived foreground masks select transformer tokens and emitted Gaussians (details in Appendix C.1).

We also support RGB video on a black background and six independently encoded axis-aligned orthographic views, with the same architecture and depth-anchored objective. RGBA video is most robust overall; results page `https://videorng.github.io/` shows examples of all variants.

3.2 Stage 2: Reconstructing Foreground-Sparse rtRNG Assets

Output representation. An asset comprises N *ray-traced Relightable Neural Gaussians* (rtRNG), $\{g_i\}$, each with a position $\boldsymbol{\mu}_i$, an anisotropic scale $\mathbf{s}_i \in \mathbb{R}^3$, a rotation $\mathbf{q}_i$, an opacity α_i, and a neural appearance latent $\mathbf{z}_i \in \mathbb{R}^{32}$. Section 3.4 describes their rendering.

Foreground tokens. Each latent frame (with resolution 128×128) is patchified into 64×64 tokens by the patch embedding of a pretrained video diffusion transformer (Wan Team, 2025), which we use as the reconstruction backbone. Let $m_{v,j} \in \{0, 1\}$ indicate whether patch j of latent observation v contains any foreground pixel; the transformer processes only

$$\mathcal{T}_{\text{fg}} \;=\; \{\, t_{v,j} \;\mid\; m_{v,j} = 1 \,\}, \tag{2}$$

and background tokens are removed *before* the first attention layer. Kept tokens retain the rotary embedding of their original latent-frame index and spatial location, so the attention pattern the backbone was pretrained with is preserved on the subset. Each token additionally receives its associated keyframe camera geometry through eight extra input channels of the patch embedding, initialized to zero at the start of training: the six Plücker coordinates of the pixel ray (Zhang et al., 2024) and the sine and cosine of the keyframe azimuth. No other view or pose information is injected; the full self-attention of the backbone over the kept tokens of all eight latent frames performs the cross-view fusion. On our data the foreground occupies about a fifth of the image area on average (21.6% of the pixels; 24% of the tokens survive), so this step removes the majority of the tokens and, downstream, the majority of the Gaussians (Appendix C.5).

Gaussian prediction. The reconstructor in Figure 2(c) fine-tunes all parameters of a pretrained 1.3B-parameter video DiT (30 blocks, width 1536) for 3D reconstruction. We replace the DiT's denoising head with a linear pixel head following GS-LRM (Zhang et al., 2024). After the final layer normalization and adaptive modulation, a single linear layer maps each retained 1536-dimensional token to a 16×16 block of per-pixel attributes. Each token covers a 2×2 latent patch, yielding outputs at the full 1024^2 resolution without a convolutional or VAE decoder. Each attribute vector has 41 used channels: one depth logit, three log-scales, four quaternion components, one opacity logit and the 32-dimensional appearance latent. Every foreground pixel (v, p) emits exactly one Gaussian along its own camera ray. Writing $\mathbf{o}_{v,p}$ and $\mathbf{r}_{v,p}$ for the ray origin and unit direction, the head predicts a depth logit $w_{v,p}$ and the position is

$$\hat{\boldsymbol{\mu}}_{v,p} \;=\; \mathbf{o}_{v,p} + \hat{d}_{v,p}\, \mathbf{r}_{v,p}, \qquad \hat{d}_{v,p} \;=\; d_{\min} + (d_{\max} - d_{\min})\ \text{sigmoid}(w_{v,p}), \tag{3}$$

here $[d_{\min}, d_{\max}]$ is the depth range of the object's bounding sphere along the ray, and the network predicts the logit $w_{v,p}$ rather than the depth itself. Hats denote quantities obtained from this prediction, while d^* denotes ground truth depth (Section 3.3). The same parameterization serves perspective and orthographic cameras, differing only in how $\mathbf{o}$ and $\mathbf{r}$ are formed. The remaining channels of the attribute vector give the log-scale (exponentiated and clamped), the quaternion (normalized), the opacity (sigmoid), and the 32-dimensional appearance latent. We prune Gaussians whose predicted opacity is below 0.01, which removes low-contribution primitives. The Gaussians from all eight latent observations are concatenated into one asset after opacity pruning, with no additional geometric merging step; at 1024^2 this is typically one to six million Gaussians per object. Constraining every Gaussian to its associated keyframe camera ray means the network never has to solve a free-space placement problem: geometry is a per-pixel depth, and appearance is a per-pixel latent, both directly aligned with the observation that produced them.

Rendering cost grows with Gaussian count. We group assets into four buckets by foreground token number and sample each distributed batch from one bucket, reducing cost variation across GPUs

while preserving each asset's marginal sampling probability. Appendix C.5 gives the procedure and training-cost measurements.

3.3 Training: Depth-Anchored Appearance Learning

During LRM training, Gaussians from all eight latent observations are rendered jointly under point lights at camera poses distinct from those of the input turntable. Section 3.2 places every Gaussian on its pixel ray at a depth predicted by the network (Eq. 3). The reconstructor learns from rendered images of synthetic assets under different cameras and point lights than the input frames, so the natural training strategy is to optimize predicted depth jointly with all other attributes through the rendering loss. When that gradient flows through predicted depth, however, reducing image error can move Gaussian centers as well as change appearance. We observed blurry appearance under such joint training, including when a direct depth loss is added (Appendix C.3). This motivates a fixed spatial reference for appearance learning: during training, we place Gaussian centers at ground truth depths and supervise the depth predictor separately.

Depth anchoring. Synthetic training assets provide ground truth depths d^* along the input camera rays and reference images I^* under supervision cameras and lights. Let $R(d, \eta)$ denote rendering with depths d and the remaining predicted Gaussian attributes η (scales, rotations, opacities, and appearance features). The shared appearance decoder, supervision camera, and lights are implicit in R. The network predicts $\hat{d}$ and η; the renderer's depth argument d can be supplied by either $\hat{d}$ or d^*. Let $\mathcal{L}_{\text{img}}$ be the image-reconstruction loss and $\mathcal{L}_d$ the supervised depth loss. The equations below show these two terms; Appendix C.2 gives the complete objective, including auxiliary losses. Joint training with depth supervision uses

$$\mathcal{L}_{\text{joint}} \;=\; \mathcal{L}_{\text{img}}\big(R(\hat{d}, \eta),\, I^*\big) \;+\; \lambda_d\, \mathcal{L}_d(\hat{d},\, d^*). \tag{4}$$

Three ways of breaking the coupling between geometry and appearance follow naturally. Stopping the gradient of $\mathcal{L}_{\text{img}}$ into $\hat{d}$ leaves the depth head to the depth loss alone, but appearance is still learned against a geometry that is both inaccurate and changing. *Freezing* the predicted depth of every object after a warm-up of K steps removes the motion, at the cost of a schedule and a per-object buffer of frozen depths. *Anchoring* fixes the depth used to place each Gaussian center at its ground truth value. The renderer uses d^*, while the predicted depths $\hat{d}$ continue to receive direct supervision:

$$\mathcal{L}_{\text{ours}} \;=\; \mathcal{L}_{\text{img}}\big(R(d^*, \eta),\, I^*\big) \;+\; \lambda_d\, \mathcal{L}_d(\hat{d},\, d^*). \tag{5}$$

Equations 4 and 5 differ in the depths supplied to the training renderer; their depth-supervision term is identical. Under Eq. 5, Gaussian centers are fixed for each training asset, while scales, rotations, opacities, appearance features, and the shared decoder remain trainable. The depth and attribute predictions share a backbone, so anchoring separates the gradient paths through the renderer, not the model parameters. We use this depth-anchoring principle for reconstruction; Figure 10 and Table 5 in Appendix C.3 compare the training strategies. The same appendix also compares training with the rendering loss only against depth-anchored training on the full training set (Table 6, Figure 11).

Alpha supervision. So far the opacities of the Gaussians are only implicitly learned through the $\mathcal{L}_{\text{img}}$ term, where the final color is α-composited based on the individual Gaussians' opacities along the ray. We observed that having a constant black background can confuse the model into reconstructing black silhouettes. To avoid this ambiguity between opacity and color in $\mathcal{L}_{\text{img}}$, we use a random gray background per sample in the ground truth RGB image, and α-composite our Gaussians with the same background. Moreover, we use an additional loss term that supervises the accumulated opacity of the Gaussians along the rays $\hat{\alpha}$ using the ground truth $\alpha^* \in [0, 1]$,

$$\mathcal{L}_{\text{total}} \;=\; \mathcal{L}_{\text{img}}\big(R(d^*, \eta),\, I^*\big) \;+\; \lambda_d\, \mathcal{L}_d(\hat{d},\, d^*) + \lambda_\alpha\, \mathcal{L}_\alpha(\hat{\alpha},\, \alpha^*) + \mathcal{L}_{\text{reg}}. \tag{6}$$

The $\mathcal{L}_{\text{img}}$ term combines ℓ_2, VGG LPIPS (Zhang et al., 2018), and $1 - \text{SSIM}$ (Wang et al., 2004), and together with the soft-alpha silhouette loss ($\mathcal{L}_\alpha$) and foreground depth supervision ($\mathcal{L}_d$) forms our loss function. Additionally, $\mathcal{L}_{\text{reg}}$ applies weak binarization regularization on rendered alpha and Gaussian opacity. Appendix C.2 gives the complete loss and regularizer definitions, weights, and augmentation settings.

Inference. ground truth depth is used only during training on our synthetic data. At inference, for a generated or supplied observation sequence, the reconstructor predicts $\hat{d}$ and η, places Gaussian

centers with Eq. 3, and renders $R(\hat{d}, \eta)$ under the requested camera and lighting; the change from $R(d^*, \eta)$ to $R(\hat{d}, \eta)$ is the depth-related train–test mismatch examined in Appendix C.3.

3.4 Rendering rtRNG Assets

A shared decoder $f_\psi(\mathbf{z}_i, \boldsymbol{\omega}_{\text{in}}, \boldsymbol{\omega}_{\text{out}})$ predicts directional appearance at render time. It is a nine-layer MLP with hidden width 512, conditioned on the appearance feature and the incoming light and outgoing view directions. (For final rendering, we use per-asset distillation into a four-layer MLP of width 64.) Both directions are defined in the Gaussian's local coordinate frame, before applying its rotation $\mathbf{q}_i$; world-space directions are mapped into this frame by $\mathbf{R}(\mathbf{q}_i)^\top$. The directional input comprises the two in-plane components of each direction, Fourier-encoded with six frequencies; the in-plane components of their half vector, encoded with one frequency; and the two components along the Gaussian's third local axis. The rotation that orients the Gaussian also fixes the frame in which its appearance is expressed, so the encoding depends only on the light and view directions relative to the Gaussian, not on any world axis, and anisotropic reflectance can be represented. We adopt the two-output shading formulation ("shadow hint") of RNA (Mullia et al., 2024): the decoder predicts two radiance responses for each incident-light and outgoing-view direction pair: f_ψ^{lit} for a visible incident direction and f_ψ^{shd} for a self-occluded one. We introduce an improvement: our renderer does not treat this as a binary decision but blends them with the transmittance $T_{i\ell} \in [0, 1]$ traced from the Gaussian along the world-space direction $\mathbf{R}(\mathbf{q}_i)\boldsymbol{\omega}_\ell$, where $\boldsymbol{\omega}_\ell$ is the sampled incident direction in the Gaussian's local frame:

$$\mathbf{c}_i(\boldsymbol{\omega}_{\text{out}}) \;=\; \sum_\ell E_\ell \Big[\, T_{i\ell}\, f_\psi^{\text{lit}}(\mathbf{z}_i, \boldsymbol{\omega}_\ell, \boldsymbol{\omega}_{\text{out}}) + (1 - T_{i\ell})\, f_\psi^{\text{shd}}(\mathbf{z}_i, \boldsymbol{\omega}_\ell, \boldsymbol{\omega}_{\text{out}}) \Big], \tag{7}$$

where E_ℓ is the weighted contribution of incident-light sample ℓ at the Gaussian. The summation accommodates point, area, and environment lighting. In a controlled fitting study the blend improves over the binary selection at every matched training checkpoint, by about 0.9 dB late in training (Appendix C.4). The computed radiances $\mathbf{c}_i$ are then alpha-composited along camera rays using the sorting-free stochastic estimator of Sun et al. (2025); shadow rays behind $T_{i\ell}$ are naturally sorting-free (particle kernel and asset conventions: Appendix C.6). During training, we use the differentiable stochastic formulation of Xu et al. (2026) to propagate image-space losses to the Gaussian parameters and neural appearance. Storing a neural latent per Gaussian follows Relightable Neural Gaussians (Fan et al., 2025) but our ray-traced formulation is much simpler to integrate into downstream renderers; we also learn the representation and its shared appearance basis end-to-end during LRM training, and predict a new rtRNG asset in a single forward pass, rather than fit it per object from hundreds of consistent observations.

We integrate rtRNG assets into a path tracer for scene rendering with inter-asset shadowing and global illumination. We support per-asset distillation of the shared decoder f_ψ for faster rendering. Per-asset distillation fits a width-64 decoder to the frozen teacher's directional responses on that asset's Gaussian features, without image supervision or updates to the Gaussian attributes. It takes approximately 2 minutes per object on one GPU. The rendering time in Figure 1 uses these per-asset decoders; Appendix C.7 details their architecture, sampling, and objective, and Appendix C.8 reports gallery rendering performance.

4 Experiments

4.1 Experimental Setup

Data and training. We train the video generator and the large reconstruction model (LRM) on approximately 22K textured assets curated from TexVerse (Zhang et al., 2025b). We first discard assets with overly simple PBR materials, then keep those whose captions describe materials, preferring complex materials and single objects rather than full scenes. For each asset, we render an 81-frame RGBA turntable at 1024^2 under a fixed environment light, following Section 3.1. This set is used to train our video generator and provides the input observations to the LRM. To supervise the LRM, we also render 128 views per asset under different camera/point lights than the 81-frame trajectory, with ground truth depth and masks. This set contains 64 frames with fixed and structured camera/light trajectories along with 64 per-asset random camera/light positions, with the aim of providing more

Table 1: Text-conditioned generation on 105 objects (four views each). Best scores are bold; evaluation details are in Appendix A.

Method	CLIP-T↑	R@1↑	R@5↑	PickScore↑	ImageReward↑
Ours	**28.1**	**0.81**	**1.00**	**20.59**	**-0.19**
TRELLIS text-xlarge	24.1	0.56	0.91	19.70	-1.00
LGM (MVDream views)	25.9	0.74	0.99	19.76	-0.73
DiffSplat (PixArt-Σ)	24.3	0.59	0.91	19.55	-0.90
Shap-E	17.6	0.12	0.35	18.32	-2.11
GaussianAnything	19.5	0.20	0.51	18.51	-1.82
Direct2.5	21.4	0.43	0.60	18.86	-1.53

robustness to the LRM supervision. Reconstruction benchmarks evaluate validation objects under held-out views and lights; text-generation benchmarks evaluate assets synthesized from prompts.

The two models are trained separately. We fine-tune the video generator on the rendered turntable clips with its flow-matching objective for three epochs. The LRM is initialized from the pretrained 1.3B video DiT and fine-tuned end to end with the depth-anchored objective (Section 3.3), following a coarse-to-fine schedule from 256^2 to 1024^2 with AdamW at a constant learning rate of 2×10^{-5}. Appendix C.2 reports the step counts, compute setup, and input augmentation for both models.

Baselines. We group methods by the conditioning they consume. In Figure 4, our reconstructor and our multi-image fine-tuning of TRELLIS.2 receive the same generated multi-view observations. Original TRELLIS.2 (Xiang et al., 2025a) and TRELLIS (Xiang et al., 2025b) receive only the generated front view. DiffSplat (Lin et al., 2025), TRELLIS text-xlarge, and LGM with MVDream (Tang et al., 2024; Shi et al., 2024) receive text prompts. Outputs in this comparison are rendered with the same orthographic cameras. The broader text-generation and ground truth reconstruction comparisons, including their input protocols, are detailed in Appendices A and B.

TRELLIS.2 generates meshes with PBR materials, which we use for relighting in Blender Cycles. TRELLIS supports mesh, Gaussian, and radiance-field outputs; LGM and DiffSplat generate Gaussians with baked appearance. In the ground truth benchmark, Hunyuan3D-2mv + Paint and LSRM produce textured meshes, relit with their textures as albedo. VideoRNG predicts Gaussians with neural appearance features: under a new environment map, its shared decoder evaluates responses to sampled incident-light directions, combined with ray-traced visibility as in Eq. 7.

4.2 Generation and Reconstruction Quality

Generation from prompts. Figure 1 shows generated assets integrated into one scene. We provide a quantitative analysis of our text-prompt generated objects based on several metrics in Table 1; CLIP-T (Radford et al., 2021) and R-Precision measure text–image similarity and prompt retrieval, while PickScore (Kirstain et al., 2023) and ImageReward (Xu et al., 2023) are learned human-preference models, detailed in Appendix A. VideoRNG obtains the best value on all five metrics, with LGM the strongest baseline on each of them. Text-conditioned methods receive only the prompt and may generate different instances of the described object.

Figure 5 in Appendix A provides a qualitative comparison of text-conditioned generation with the baselines. Figure 6 shows an image-conditioned qualitative comparison with TRELLIS.2-4B, TRELLIS image-large, and LGM with ImageDream, which receive the front view generated by our text-conditioned pipeline, whereas our reconstruction uses multiple generated observations. Appendix D contains the original prompt-construction instructions.

Human preference study. In a user study with 505 participants, each shown a target description and one rendered image per method, participants choose the result they prefer for visual quality and description alignment jointly. Figure 3 shows that VideoRNG receives a majority of selections (57.75%), followed by LGM (MVDream) at 18.85%. Appendix A provides the survey protocol.

Reconstruction of generated observations. Figure 4 compares generated objects under the same camera, grouping methods by their inputs as described in Section 4.1. Our reconstruction follows the generated observations; the fine-tuned TRELLIS.2 model keeps the shape but loses the fur (flat orange); the single-view generators change the web into a bowl and the coat into pale yellow. As

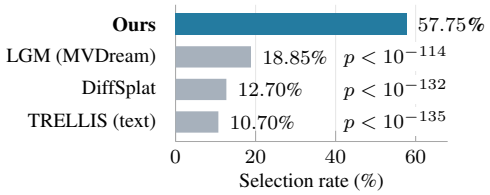

Figure 3: Human preference for text-to-3D renderings, judging visual quality and text alignment jointly. Annotations show p-value bounds for paired tests of per-participant selection rates against VideoRNG (Appendix A).

Table 2: GT-view reconstruction on 50 complex validation objects. Hunyuan3D-2mv receives four input views and LSRM sixteen posed views. The two lighting settings use different evaluation cameras (Appendix B).

	Input light		New light	
Method	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓
Ours	**26.25**	**0.138**	**25.99**	**0.154**
TRELLIS.2 multi-image	21.20	0.264	20.54	0.284
LGM	21.73	0.227	21.44	0.248
Hunyuan3D-2mv + Paint	19.82	0.269	19.80	0.288
TRELLIS.2-4B	19.42	0.268	20.04	0.280
LSRM	21.01	0.187	22.47	0.186

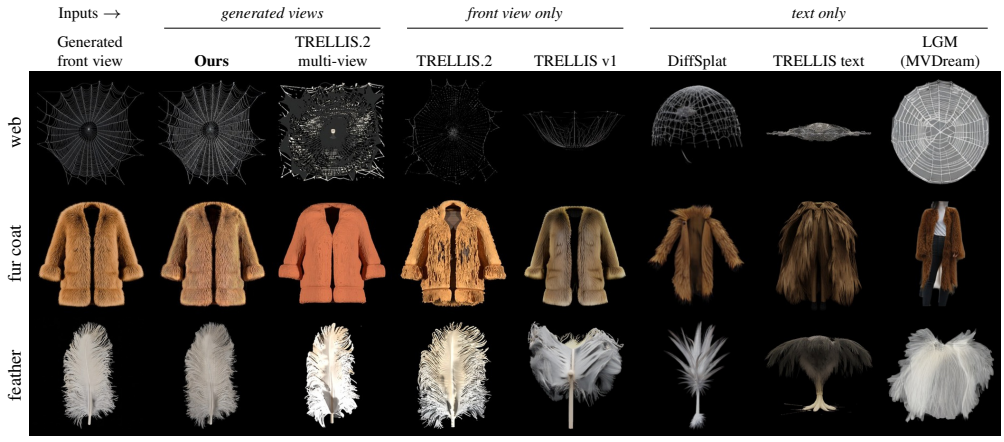

Figure 4: Comparison using generated observations. The first column shows a front frame from our video generator, not ground truth. VideoRNG and multi-view TRELLIS.2 receive the generated view set; single-image baselines receive this front view; text-only baselines receive prompts.

expected, the text-only generators produce different objects, but with less visual fidelity: a dome, a flat disc and a round mesh for the web.

Reconstruction accuracy comparison. To isolate reconstruction quality from errors in generated observations, we supply each method with rendered ground truth views of the object. We evaluate 50 complex validation objects on eight random perspective views, each under a different random HDRI. Table 2 reports PSNR and LPIPS; Appendix B provides the full protocol and SSIM results (Table 3). VideoRNG has the best reported PSNR, SSIM, and LPIPS in both settings. Under new illumination, each method is rendered as described in Section 4.1 and Appendix B; Figure 7 compares the reconstruction quality on new views. Figure 8 and Table 4 compare full single-image pipelines.

View and illumination generalization. We test whether reconstructed appearance remains consistent under novel views and rotation of the known studio environment map, comparing 20 synthetic validation assets with Blender ground truth (Figure 9). Mean PSNR is 29.26 dB on the input frames, 29.11 dB on the novel orbit, and 29.09 dB under new illumination. The moving highlights and darkening of surfaces follow the ground truth. Table 2 evaluates new HDRIs.

Direct per-object optimization. Per-object fitting improves input-view PSNR in the examples of Figure 12, while our feed-forward LRM better preserves fine structure under novel views and illumination. Appendix C.9 gives the setup and detailed comparison.

5 Conclusion

We presented **VideoRNG**, a two-stage pipeline turning text or image prompts into **rtRNG** assets capable of capturing fine structures, relighting, and integration into full scenes. A fine-tuned video generator supplies structured observations, and a feed-forward LRM converts them into Gaussians with neural directional appearance. More results: `https://videorng.github.io/`

AI USE STATEMENT

Coding agents were used in code development and experiment execution. An AI assistant was used to construct candidate object descriptions for text-conditioned generation and to suggest improvements to the manuscript. Appendix D includes the original prompt-construction instructions. The authors take full responsibility for the paper's claims and results.

REPRODUCIBILITY STATEMENT

Section 3 specifies the observation protocol, representation, reconstruction architecture, and depth-anchored objective. Appendix C provides camera calibration, training schedules, loss weights, augmentation, and sampling details. Appendices A and B describe the comparison sets, baseline inputs, rendering conditions, and evaluation metrics. The work is based on the open Wan video models and TexVerse data.

REFERENCES

Haotian Dong, Wenjing Wang, Chen Li, Jing Lyu, and Di Lin. Video Generation with Stable Transparency via Shiftable RGB-A Distribution Learner. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1885–1894, June 2026.

Jiahui Fan, Fujun Luan, Jian Yang, Milos Hasan, and Beibei Wang. RNG: Relightable neural gaussians. *Proceedings of CVPR 2025*, 2025.

Guangming Fu, Jiahui Fan, Jian Yang, Miloš Hašan, and Beibei Wang. F-RNG: Feed-Forward Relightable Neural Gaussians, 2026. URL https://arxiv.org/abs/2605.25975.

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 B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 50678–50702, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/dcad3425f5c8c36b5b3885c091bf1257-Paper-Conference.pdf.

Jiaxin Huang, Yuanbo Yang, Bangbang Yang, Lin Ma, Yuewen Ma, and Yiyi Liao. Gen3R: 3D Scene Generation Meets Feed-Forward Reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 25358–25369, June 2026.

Heewoo Jun and Alex Nichol. Shap-E: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4), 2023.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-Pic: an open dataset of user preferences for text-to-image generation. In *Proceedings of NeurIPS 2023*, volume 36, 2023.

Alexandr Kuznetsov, Krishna Mullia, Zexiang Xu, Miloš Hašan, and Ravi Ramamoorthi. NeuMIP: multi-resolution neural materials. *ACM Trans. Graph.*, 40(4), 2021.

Yushi Lan, Shangchen Zhou, Zhaoyang Lyu, Fangzhou Hong, Shuai Yang, Bo Dai, Xingang Pan, and Chen Change Loy. GaussianAnything: Interactive point cloud flow matching for 3d object generation. In *Proceedings of ICLR 2025*, 2025.

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. In *Proceedings of ICLR 2024*, 2024.

Zhengqin Li, Cheng Zhang, Jakob Engel, and Zhao Dong. Lsrm: High-fidelity object-centric reconstruction via scaled context windows. In Paolo Favaro, Zuzana Kukelova, Atsuto Maki, Anna Rohrbach, Konrad Schindler, and Federico Tombari (eds.), *Computer Vision – ECCV 2026*, pp. 266–286, Cham, 2026. Springer Nature Switzerland. ISBN 978-3-032-37447-9.

Chenguo Lin, Panwang Pan, Bangbang Yang, Zeming Li, and Yadong MU. DiffSplat: Repurposing Image Diffusion Models for Scalable Gaussian Splat Generation. In *Proceedings of ICLR 2025*, 2025.

Yuanxun Lu, Jingyang Zhang, Shiwei Li, Tian Fang, David McKinnon, Yanghai Tsin, Long Quan, Xun Cao, and Yao Yao. Direct2.5: Diverse text-to-3d generation via multi-view 2.5d diffusion. In *Proceedings of CVPR 2024*, pp. 8744–8753, 2024.

Stephen McAuley, Stephen Hill, Naty Hoffman, Yoshiharu Gotanda, Brian Smits, Brent Burley, and Adam Martinez. Practical physically-based shading in film and game production. In *SIGGRAPH 2012 Courses*, 2012.

Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3D gaussian ray tracing: Fast tracing of particle scenes. *ACM Trans. Graph.*, 43(6), 2024.

Krishna Mullia, Fujun Luan, Xin Sun, and Miloš Hašan. RNA: Relightable neural assets. *ACM Trans. Graph.*, 44(1), 2024.

OpenAI. GPT-6 Astra: A new generation of intelligence, 2026. URL `https://openai.com/index/gpt-6-astra/`.

Panwang Pan, Chenguo Lin, Chenxin Li, Jingjing Zhao, Yuchen Lin, Haopeng Li, Yunlong Lin, Kairun Wen, Yixuan Yuan, and Yadong MU. Diff4Splat: Repurposing Video Diffusion Models for Dynamic Scene Generation. In *Proceedings of CVPR 2026*, 2026.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3D using 2D diffusion. In *Proceedings of ICLR 2023*, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of ICML 2021*, pp. 8748–8763, 2021.

Nithin Raghavan, Krishna Mullia, Alexander Trevithick, Fujun Luan, Miloš Hašan, and Ravi Ramamoorthi. Generative neural materials. In *Proceedings of SIGGRAPH 2025*, 2025.

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

Mayank Singh, Michele Stoppa, Alvise Memo, Rui Yu, Harsha Kalli, Srimanth Gunturi, Muhammad Ahmed Riaz, Behrooz Shahsavari, Waleed Abdulla, and David E. Jacobs. Luce: Relightable gaussians for 3D asset generation, 2026.

Xin Sun, Iliyan Georgiev, Yun Fei, and Miloš Hašan. Stochastic ray tracing of 3D transparent gaussians. *EGSR conference proceedings*, 2025.

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. In *Proceedings of ECCV 2024*, pp. 1–18, 2024.

Vikram Voleti, Chun-Han Yao, Mark Boss, Adam Letts, David Pankratz, Dmitry Tochilkin, Christian Laforte, Robin Rombach, and Varun Jampani. Sv3d: Novel multi-view synthesis and 3d generation from a single image using latent video diffusion. In *Proceedings of ECCV 2024*, pp. 439–457, 2024.

Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of EGSR 2007*, pp. 195–206, 2007.

Wan Team. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

Liwen Wu, Haolin Lu, Bing Xu, Miloš Hašan, and Ravi Ramamoorthi. 8DNA: 8D neural asset light transport by distribution learning. In *Proceedings of SIGGRAPH 2026*, 2026.

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, and Jiaolong Yang. Native and compact structured latents for 3D generation. *Tech report*, 2025a.

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3D latents for scalable and versatile 3d generation. In *Proceedings of CVPR 2025*, pp. 21469–21480, 2025b.

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, 2024a.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. ImageReward: Learning and evaluating human preferences for text-to-image generation. In *Proceedings of NeurIPS 2023*, volume 36, 2023.

Peiyu Xu, Shuang Zhao, Xin Sun, Krishna Mullia, Raymond Fei, and Iliyan Georgiev. Stochastic Ray Tracing for the Reconstruction of 3D Gaussian Splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19001–19010, June 2026.

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. In *Proceedings of ECCV 2024*, pp. 1–20, 2024b.

Bowen Xue, Shuang Zhao, Henrik Wann Jensen, and Zahra Montazeri. A hierarchical architecture for neural materials. *Computer Graphics Forum*, 43(6):e15116, 2024.

Bowen Xue, Saeed Hadadan, Zheng Zeng, Fabrice Rousselle, Zahra Montazeri, and Miloš Hašan. Videoneumat: Neural material extraction from generative video models. In *Proceedings of SIGGRAPH 2026*, 2026.

Jingrui Ye, Lingting Zhu, Runze Zhang, Zeyu Hu, Yingda Yin, Lanjiong Li, Lequan Yu, and Qingmin Liao. Large material gaussian model for relightable 3D generation, 2025.

Tizian Zeltner, Fabrice Rousselle, Andrea Weidlich, Petrik Clarberg, Jan Novák, Benedikt Bitterli, Alex Evans, Tomáš Davidovič, Simon Kallweit, and Aaron Lefohn. Real-time neural appearance models. *ACM Trans. Graph.*, 43(3), 2024.

Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. GS-LRM: Large reconstruction model for 3D gaussian splatting. In *Proceedings of ECCV 2024*, pp. 1–19, 2024.

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

Tianyuan Zhang, Zhengfei Kuang, Haian Jin, Zexiang Xu, Sai Bi, Hao Tan, He Zhang, Yiwei Hu, Milos Hasan, William T. Freeman, Kai Zhang, and Fujun Luan. RelitLRM: Generative relightable radiance for large reconstruction models. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL `https://openreview.net/forum?id=3Oli4u6q3p`.

Yibo Zhang, Li Zhang, Rui Ma, and Nan Cao. Texverse: A universe of 3d objects with high-resolution textures. *arXiv preprint arXiv:2508.10868*, 2025b.

Zibo Zhao et al. Hunyuan3D 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv preprint arXiv:2501.12202*, 2025.

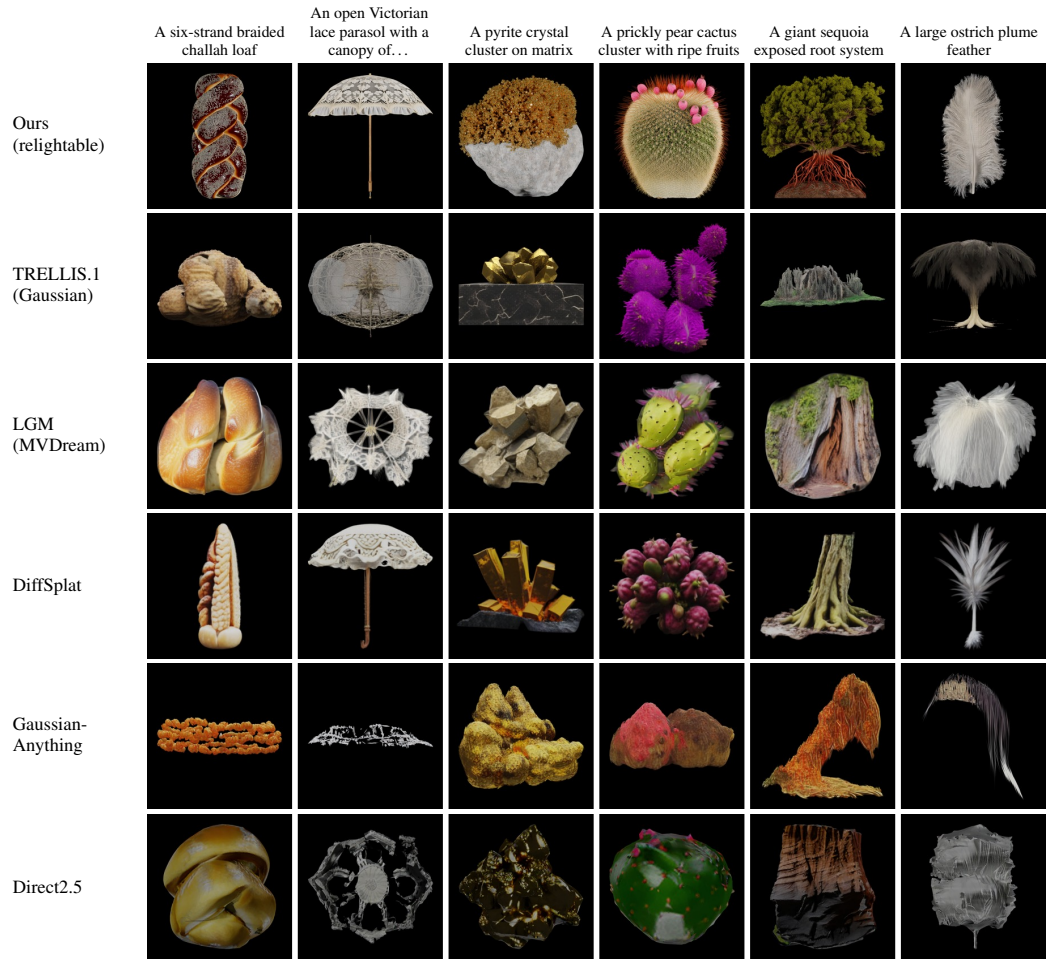

Figure 5: Comparison with text-conditioned generation methods on the six objects from Fig. 6 (columns; headers summarize the prompt subjects), front views. Our row is the full text pipeline (prompt → generated views → LRM). Note that only our method produces relightable assets.

A Comparison with text-conditioned 3D generators

We compare our full text-to-3D pipeline with public text-conditioned generators on 105 selected objects. Following the prompt-construction protocol in Appendix D, this set emphasizes intricate geometry and appearance visible from lateral viewpoints. The baselines are TRELLIS text-xlarge, LGM with MVDream, DiffSplat (PixArt-Σ), Shap-E (Jun & Nichol, 2023), GaussianAnything (Lan et al., 2025), and Direct2.5 (Lu et al., 2024), each run with its official code and default settings. All outputs are rendered with the same four-view rig and studio lighting, using Blender for meshes and a 3DGS rasterizer for Gaussians. Text-conditioned methods receive only the prompt and may generate different instances of the described object, so Table 1 reports reference-free scores. CLIP-T is ViT-L/14 text–image similarity multiplied by 100 (Radford et al., 2021). R@k is the fraction of objects whose own prompt ranks among the top-k of the 105 prompt entries by CLIP similarity. Scores are averaged over four rendered side views per object. PickScore (Kirstain et al., 2023) and ImageReward (Xu et al., 2023) are learned human-preference models.

Figure 4 compares the web and fur-coat examples under matched cameras. Figure 5 shows six further example objects. Figure 6 compares reconstruction from generated observations with image-conditioned generators given our generated front view (TRELLIS.2-4B, TRELLIS image-large, and LGM with ImageDream). For the qualitative GPT-6 Astra comparison (OpenAI, 2026), we used Ultra mode and supplied the complete 81-frame generated video with the prompt "Create a Blender 3D asset that matches this 81-frame video." Appendix B separately evaluates reconstruction against ground truth on 50 complex validation objects.

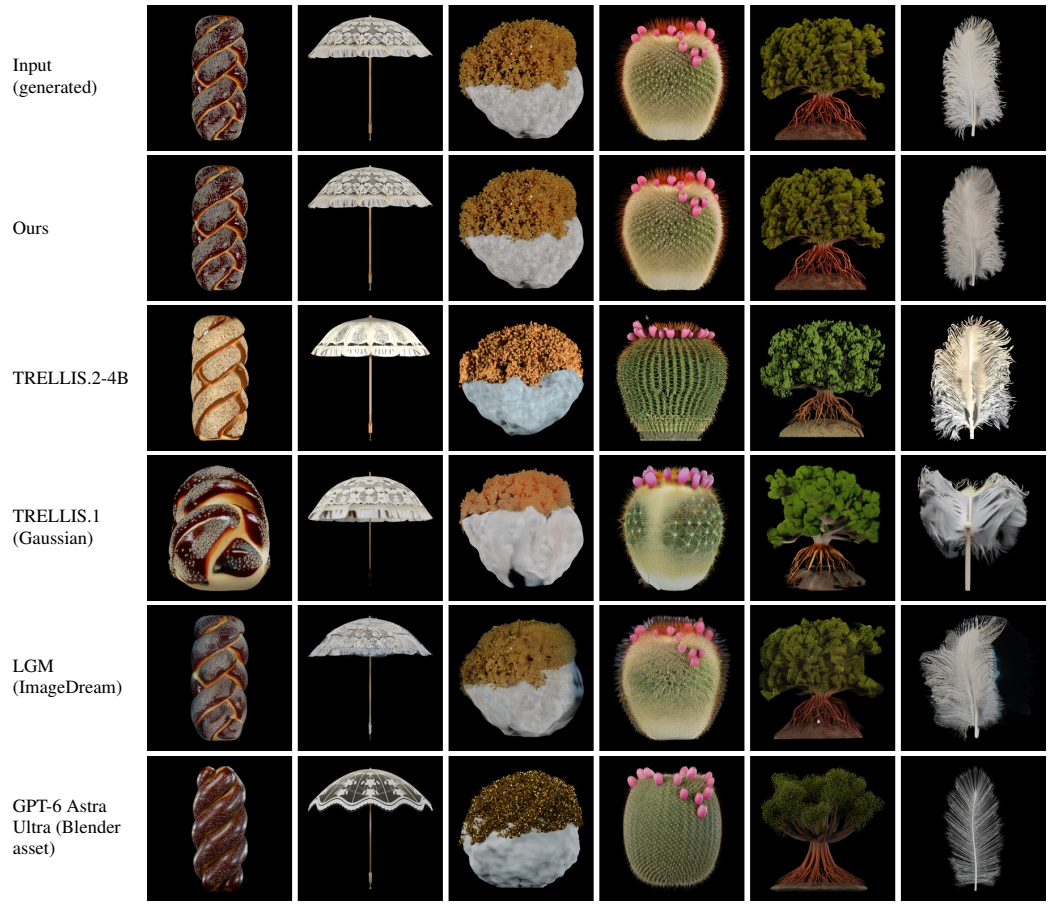

Figure 6: Our generated front view used as input to image-conditioned generators. Top row: the view our text pipeline generated for the prompt (same objects and column order as Fig. 5), which TRELLIS.2-4B, TRELLIS image-large and LGM (ImageDream) receive as their input; second row: our reconstruction of the generated views. Front views; each baseline cell is the one of its four side views that best matches the input orientation. Last row: GPT-6 Astra Ultra Blender results, generated from the complete 81-frame video.

Human preference study. We evaluate renderings of text-generated assets using 30 single-choice questions. Each question presents a target description prepared for the survey and one rendered image from each of VideoRNG, LGM (MVDream), DiffSplat, and TRELLIS (text). The results are labeled A–D, with method order varying between questions. Respondents make one choice based jointly on overall visual quality and agreement with the target description.

The 505 participants each answered all 30 questions. Figure 3 reports the fraction of choices assigned to each method across all questions and participants. For statistical analysis, we compute one selection rate per method for each participant and compare VideoRNG with each baseline using a two-sided paired t-test ($\mathrm{df} = 504$). We apply Holm correction to control the probability of making any false-positive significance claim across the three comparisons; all adjusted p-values are below 0.001.

Each question asked: "Considering overall visual quality and how well each result matches the target description, which result do you prefer?" The accompanying instructions were:

> For each object, read the target description and compare results A, B, C, and D. Make one choice considering both overall visual quality (shape, detail, and material appearance) and how well the result matches the described object and its attributes. These descriptions were prepared for this survey. Letters identify positions within each question; the order changes between questions. There are 30 single-choice questions; no name or contact details are requested.

Table 3: Reconstruction from ground truth views on 50 complex validation objects (protocol in Appendix B). Best values are bold.

	Input light			New light		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Ours	**26.25**	**0.829**	**0.138**	**25.99**	**0.797**	**0.154**
TRELLIS.2 multi-image	21.20	0.672	0.264	20.54	0.637	0.284
LGM	21.73	0.698	0.227	21.44	0.675	0.248
Hunyuan3D-2mv + Paint	19.82	0.665	0.269	19.80	0.634	0.288
TRELLIS.2-4B	19.42	0.650	0.268	20.04	0.626	0.280
LSRM	21.01	0.738	0.187	22.47	0.728	0.186

Table 4: Metrics for the objects of Figure 8 : its 81-view ground truth orbit under the input illumination (left) and the eight random views under novel environment maps (right). Best values are bold.

	Input illumination			New illumination		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Ours	**22.15**	**0.785**	**0.168**	**22.72**	**0.784**	**0.173**
TRELLIS.2-4B	19.21	0.744	0.196	19.66	0.746	0.204
TRELLIS image-large	18.38	0.752	0.195	17.45	0.740	0.202
DiffSplat (PixArt-Σ, image-cond.)	19.10	0.759	0.201	19.87	0.757	0.209
LGM (ImageDream)	19.86	0.725	0.232	19.51	0.731	0.231

B Comparison against ground truth on complex synthetic objects

This section compares our reconstructions with public 3D generators on objects for which exact ground truth exists. We take the 50 most complex objects of our validation set (TexVerse assets ranked by depth-discontinuity density, silhouette complexity and texture detail of their rendered views, with foreground coverage between 10% and 55%). Two ground truth sets are used: (i) held-out views under the studio illumination of the training data, and (ii) eight random perspective views of each object, each rendered under a different random HDRI with random rotation, intensity and color gain. Set (i) tests novel views under the illumination a method has seen in its input, set (ii) novel views under novel illumination. The two sets use different evaluation views as well as illumination, so their score difference is not a paired comparison of lighting alone. Baseline outputs are brought into the dataset frame before rendering (orientation and scale chosen by silhouette agreement with the input views). Mesh baselines are rendered in Blender. Under the novel HDRIs of set (ii), the TRELLIS.2 variants are relit in Cycles using their PBR materials. LGM is rendered with a 3DGS rasterizer and retains its baked appearance under new illumination. Our Gaussians are rendered with our own renderer, whose decoder relights them under the per-view HDRI. Public baselines use their official code and default settings, including their own background removal; Hunyuan3D-2mv receives four input views and uses the Hunyuan3D 2.0 shape model followed by Hunyuan3D-Paint (Zhao et al., 2025); LSRM (Li et al., 2026) receives sixteen posed perspective views (two rings at 20° and 45° elevation, rendered under the same illumination) and returns a textured mesh, run with its released checkpoint and default settings; and the TRELLIS.2 multi-image baseline is our fine-tuning of the official model to accept multiple input views.

Reconstruction from rendered views. Every method receives rendered views of the synthetic object. This isolates the reconstruction networks from any generative front end (Table 3, Figure 7).

View and illumination generalization. Figure 9 shows the test described in Section 4.2: a 15°-elevated novel orbit under the training illumination and a fixed novel view under rotations of the training environment map, compared with Blender ground truth on 20 validation assets. Please note that we train our LRM on a disjoint set of frames from the input frames: the supervision set consists of 64 renderings at fixed and structured trajectories of light and camera across the dataset, as well as 64 per-object random camera and light locations; this increases robustness of the LRM outputs under novel view and illuminations.

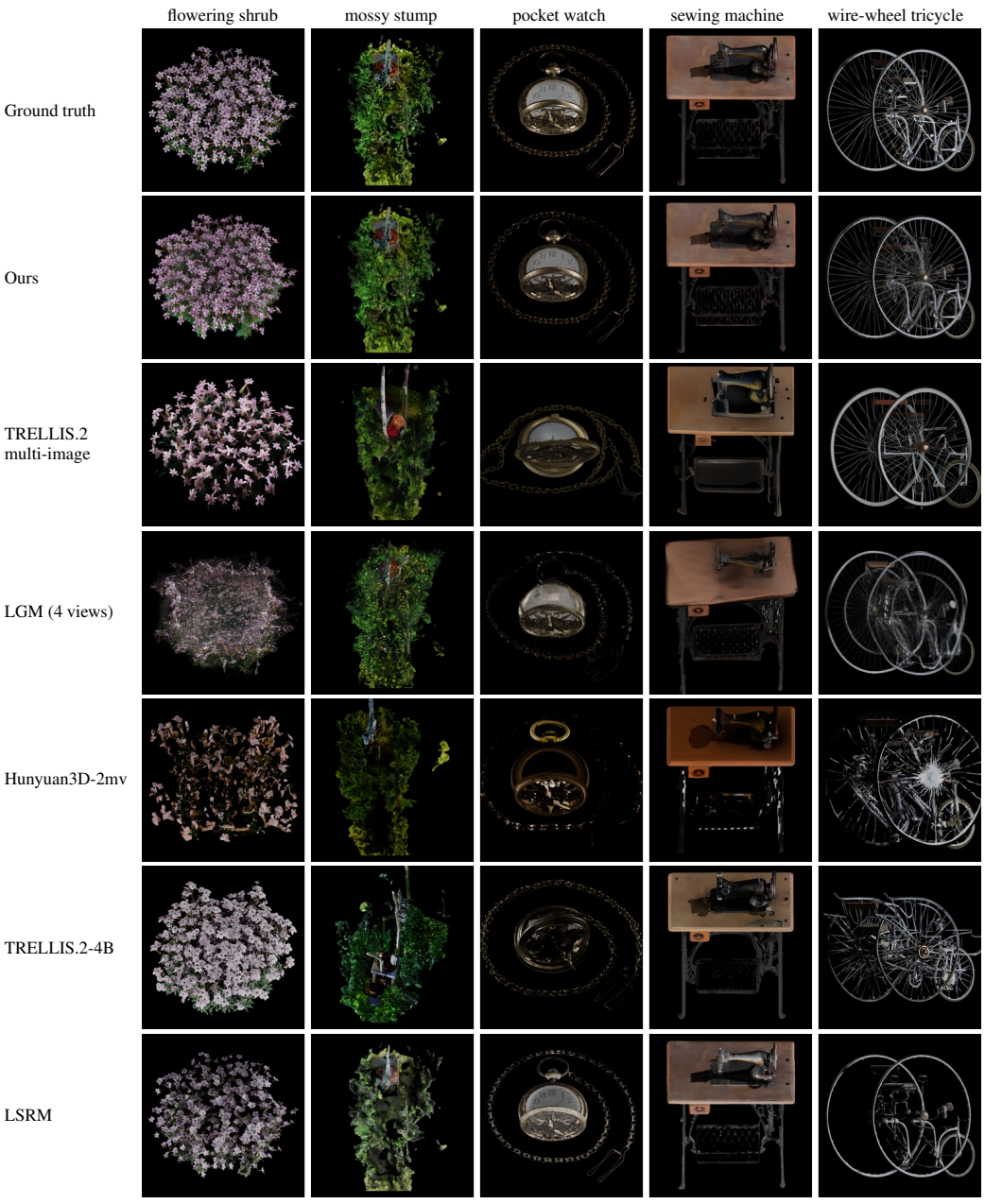

Figure 7: Table 3, reconstruction from ground truth views: new views under the new illumination. Columns are objects; rows are ground truth and the methods; within a column every cell shows the same crop of the same view.

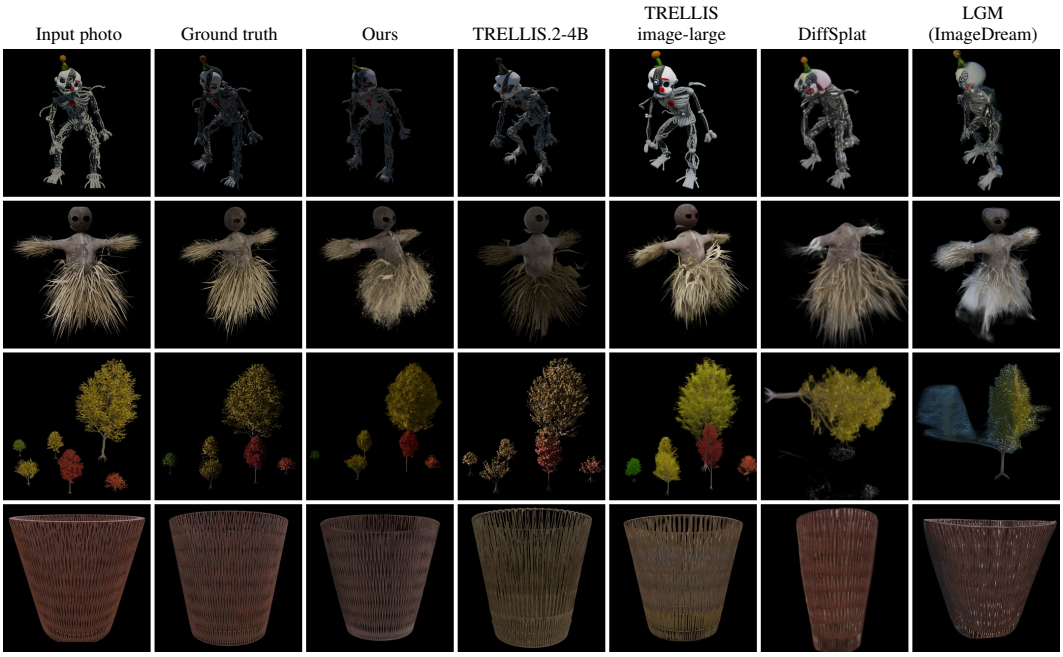

Figure 8: Full single-image pipelines on validation objects under *novel* illumination (one object per row; Table 4 scores the this table). Left: the input photo, a perspective render of the object under a random environment map. The other cells show a held-out random perspective views, rendered under a different random environment map (random rotation, intensity) that no method has seen: ground truth, ours and four public single-image generators given the same photo, rendered with the same camera.

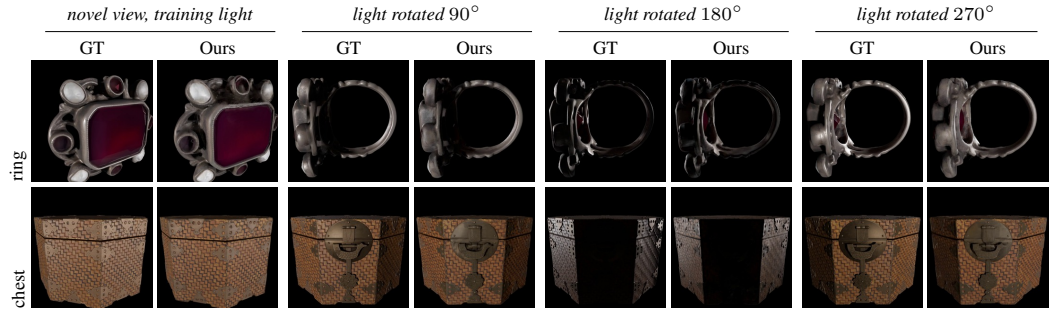

Figure 9: View and illumination generalization against Blender ground truth. The reconstructor receives the input turntable. Evaluation uses a $15°$-elevated orbit under the training illumination and a fixed novel view at the same elevation while the training environment map rotates through $360°$. PSNR is averaged over 81 frames per asset, then over 20 assets. All frames shown are novel views: the leftmost pair uses the training environment map, and the remaining pairs rotate that map by $90°$, $180°$, and $270°$. Highlights and shadowed regions follow the ground truth.

C Implementation and Additional Experiments

C.1 Observation Protocol Details

The camera is a perspective camera with a 35 mm-equivalent focal length (field of view 49.1°) at distance 1.5 from the origin of the unit-normalized object, at zero elevation and looking at the origin. Frame k shows the object rotated by $\theta_k = \theta_0 - 360^\circ\, k/F$, where θ_0 is a fixed calibration offset. We select 29 uniformly spaced frames, starting at the front view (frame 0), and encode them into eight latent frames; the first latent holds frame 0 and each further latent holds four consecutive selected frames, spanning roughly 40° of object rotation. One keyframe per latent provides the camera rays that position the emitted Gaussians (Section 3.2).

C.2 Training Schedule and Reconstruction Objective

Training. We train the video generator on approximately 22K assets for three epochs using 128 GPUs. Reconstruction follows a coarse-to-fine schedule: we first fine-tune all DiT parameters for 422.3K steps at 256^2, followed by an additional 443K steps at 1024^2. The 1024^2 stage uses 256 A100 GPUs, with one asset per GPU and one supervision view per asset per step. The reconstructor uses AdamW with a constant learning rate of 2×10^{-5}. We apply input noise augmentation with scattered points throughout the region outside the foreground mask.

Losses. We composite the prediction (using rendered alpha $\hat{\alpha}$) and reference image (using the dataset's soft alpha $\alpha^* \in [0, 1]$) over the same randomly sampled gray background. All image losses are evaluated on these full-frame composites without foreground masking:

$$\mathcal{L}_{\text{img}} = 0.8\,\mathcal{L}_{\ell_2}^{\text{sRGB}} + 0.2\,\mathcal{L}_{\text{LPIPS}}^{\text{VGG}} + 0.2\,(1 - \text{SSIM}). \tag{8}$$

The silhouette loss is $\mathcal{L}_\alpha = \text{mean}[(\hat{\alpha} - \alpha^*)^2]$, averaged over all image pixels. Its reference α^* retains soft boundaries, unlike the binary mask used for token selection. The depth loss $\mathcal{L}_d$ is the ℓ_1 distance between predicted and ground truth ray depths on foreground pixels. Two weak binarization penalties, $\mathcal{L}_{\text{bin}}^{\alpha} = \text{mean}[\hat{\alpha}(1-\hat{\alpha})]$ over rendered alpha and $\mathcal{L}_{\text{bin}}^{o} = \text{mean}_i[\alpha_i(1-\alpha_i)]$ over Gaussian opacities, express a weak preference for values near zero or one; and alpha and opacity remain continuous during optimization. Together they form $\mathcal{L}_{\text{reg}}$ in Eq. 6. The total loss is

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{img}}\big(R(d^*, \eta),\, I^*\big) + 1.0\,\mathcal{L}_\alpha + 3.0\,\mathcal{L}_d(\hat{d},\, d^*) + 0.2\,\mathcal{L}_{\text{bin}}^{\alpha} + 0.01\,\mathcal{L}_{\text{bin}}^{o}. \tag{9}$$

Full-frame compositing and soft-alpha supervision provide training signals for the background and soft boundaries, with the aim of suppressing diffuse Gaussian haze and dark edge halos.

C.3 Depth-Anchored Appearance Fitting Study

We compare depth-training strategies in a controlled fitting study limited to 50 assets. Each model is fitted to these same 50 assets and evaluated across all available views of those assets, so this experiment measures the ability to fit appearance under the different geometric training states rather than generalization to unseen assets. Table 5 reports PSNR with every model rendered using its predicted depth at evaluation. Joint optimization without a depth loss reaches 23.0 dB; adding direct depth supervision gives 26.32 dB. Preventing the rendering-loss gradient from flowing into the depth head, while retaining depth supervision, gives 26.0 dB. Freezing predicted depth after 13K steps reaches 29.00 dB, and ground truth depth anchoring reaches 30.61 dB. The comparison thus distinguishes joint optimization, gradient stopping, freezing predicted geometry, and learning appearance at ground truth Gaussian centers rather than treating depth supervision as a single on/off choice.

Figure 10 shows the joint, frozen-depth, and anchored variants on two of the fitted assets. For the anchored model, rendering once with ground truth centers and once with predicted centers isolates the deployment gap caused by predicted geometry. ground truth-depth rendering is used only for this qualitative diagnostic.

Rendering loss only on the full training set. The study above fits 50 assets. Table 6 and Figure 11 repeat the comparison at the scale of the main model: both models are trained at 256^2 on the full

Table 5: Controlled fitting study on 50 assets. Models are fitted to the same assets and evaluated across all available views of those assets using predicted depth.

Strategy	PSNR↑
Joint, no depth loss	23.00
Stop rendering-loss gradient to depth	26.00
Joint + depth loss (Eq. 4)	26.32
Predicted depth frozen after 13K steps	29.00
GT-depth anchoring (Eq. 5)	**30.61**

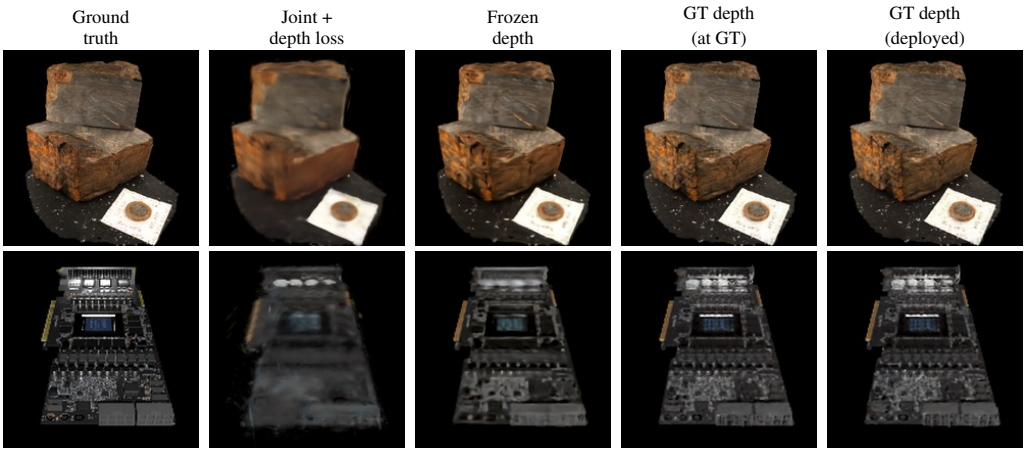

Figure 10: Qualitative comparison on two assets from the 50-asset fitting study. Each row shows the same view: ground truth; joint training with depth loss; frozen predicted depth; a ground truth-depth-trained model rendered at ground truth depth; and the same model rendered with its predicted depth. The last two columns show the deployment effect of replacing ground truth centers with predicted centers; the frozen-depth result corresponds to its row in Table 5.

training set, one with the rendering loss only (joint optimization of the predicted depth, no depth term) and one with depth anchoring (Eq. 5). We evaluate both models on unseen validation assets, each rendered from a novel viewpoint under a novel lighting condition.

C.4 Shading Rule: Transmittance Blend versus Binary Selection

Equation 7 blends the two decoder outputs with the traced transmittance $T_{i\ell}$. The shadow-hint formulation of RNA (Mullia et al., 2024) instead selects one output per light sample by thresholding the visibility at 0.5, so only the selected output receives gradient. We compare the two rules under the controlled fitting protocol of Appendix C.3 (the same 50 assets; full-frame PSNR on validation views of four of them), with everything else fixed: the same data order, initialization, optimizer, and step budget, on 32 GPUs with one sample per GPU. The comparison uses a reduced decoder of four layers and width 64, the size of our distilled per-asset decoders, trained from scratch so that decoder capacity is the bottleneck.

Table 7 lists PSNR at matched checkpoints of the same runs. Neighbouring checkpoints of one run differ by up to 0.7 dB, so a single checkpoint cannot rank the rules, but the paired differences at matched steps can. The blend is above the binary selection at every one of the seven checkpoints, and its margin grows as training proceeds, from 0.04 dB at 9K steps to 0.96 dB at 15K steps. Two properties explain the direction: the blend is the exact expectation of the direct term under fractional occlusion, and both outputs receive gradient wherever $0 < T_{i\ell} < 1$, which on a representative asset carries about 21% of the opacity mass, whereas the selection leaves one output without gradient on each side of the threshold.

C.5 Training: Efficient Foreground Sampling

Foreground sparsity makes the number of tokens and Gaussians a property of the object: across our training set the emitted Gaussian count N per object ranges from about 0.45 M (10th percentile) to

Table 6: Rendering loss only versus depth-anchored training on the full training set at 256^2: full-frame PSNR (dB), SSIM and LPIPS (VGG) evaluated on unseen objects from novel viewpoints under novel lighting conditions.

Method	PSNR↑	SSIM↑	LPIPS↓
Rendering loss only	24.49	0.759	0.162
Ours, depth-anchored	**29.22**	**0.863**	**0.064**

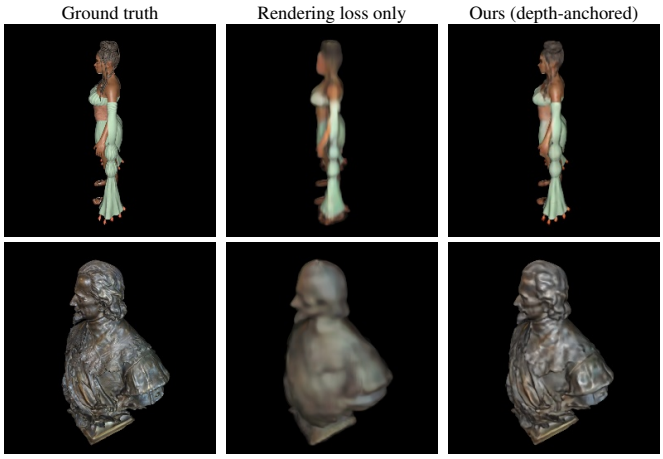

Figure 11: Rendering loss only versus depth-anchored training on the full training set at 256^2, evaluated on unseen objects from novel viewpoints under novel lighting conditions.

$8.4\,\mathrm{M}$ (maximum) at 1024^2, with a mean of $1.8\,\mathrm{M}$, and the cost of a training step—dominated by ray tracing and shading—grows roughly linearly with N. With data-parallel training on G GPUs and one object per GPU, every step waits for the largest of G independently drawn objects, so the expected step time is governed by the expected maximum of N rather than its mean; for our training set the expected maximum over 64 draws is $3.3\times$ the mean (Table 8).

We reduce this waiting without changing each object's sampling probability. Objects are sorted by N (available before training from the foreground masks) and split into K equal-size buckets. At each step, all GPUs use a common bucket sampled with probability proportional to its size, and each GPU independently samples an object uniformly within that bucket. The marginal probability of every object is unchanged, so this is still uniform sampling of the training set; only the joint distribution across ranks is altered, so that the G objects processed together have similar cost.

Measured throughput. We use $K{=}4$. Table 8 quantifies both ingredients. Foreground selection reduces the work per object from $8.4\,\mathrm{M}$ Gaussians (one per pixel of the eight 1024^2 views) to $1.8\,\mathrm{M}$ on average, but with one object per GPU every step waits for the largest of 64 independently drawn objects, whose expectation is $6.1\,\mathrm{M}$ Gaussians for our size distribution. Bucketed sampling ($K{=}4$) lowers this expected maximum to $3.1\,\mathrm{M}$ without changing the marginal sampling distribution. On 64 GPUs, it achieves $1.7\times$ the measured training throughput of foreground selection alone. Together with the foreground-only output head, this is what makes training the reconstructor at 1024^2 with millions of Gaussians per object practical.

C.6 Particle Kernel and Asset Conventions

The ray tracer evaluates every Gaussian with the generalized Gaussian kernel of 3DGRT (Moenne-Loccoz et al., 2024), which Stoch3DGS (Xu et al., 2026) keeps as its default. Along a ray, a particle's response is

$$\rho(r) \;=\; \exp\!\big(-4.5\,(r/3)^4\big), \tag{10}$$

where r is the Mahalanobis distance from the Gaussian center to the ray's closest point, and the particle is ignored where $\rho < e^{-4.5} \approx 0.011$, i.e. beyond 3σ. 3DGRT writes the exponent on the squared distance and calls this a degree-2 generalized Gaussian; its implementation refers to the same kernel as degree 4. Compared with the standard Gaussian $\exp(-r^2/2)$, which agrees with

Table 7: Shading rule ablation in the 50-asset controlled fitting protocol: full-frame PSNR (dB) on validation views. MLP decoder (4 layers, width 64) trained from scratch.

Step	9000	9250	9500	9750	10000	12500	15000
Binary selection (RNA)	28.34	28.74	28.94	28.31	28.91	29.46	29.66
Transmittance blend (Eq. 7)	**28.38**	**29.00**	**29.20**	**28.53**	**29.25**	**29.92**	**30.62**
Δ blend $-$ selection	$+0.04$	$+0.26$	$+0.26$	$+0.23$	$+0.34$	$+0.46$	$+0.96$

Table 8: Training cost at 1024^2 on 64 GPUs, one asset per GPU. Counts use the 21,985 training assets: tokens and per-asset Gaussians are means; per-step Gaussians are the expected maximum over 64 assets. A token covers 16×16 pixels. Speedup is relative to foreground selection without bucketing; full-token throughput is unavailable.

Setting	Tokens per asset	Gaussians per asset	Max Gaussians per step	Speedup
Full tokens	32,768	8.4 M	8.4 M	–
Foreground tokens	7,937 (24%)	1.8 M	6.1 M	$1.0\times$
Foreground + buckets (K=4)	7,937 (24%)	1.8 M	3.1 M	$1.7\times$

Eq. 10 only at the 3σ cutoff, the kernel is nearly flat within one standard deviation (0.95 at r=1 versus 0.61) and falls off steeply beyond it. 3DGRT adopted it because denser particles require fewer ray–particle intersections, reporting roughly $2\times$ faster tracing for a loss of about 0.3 dB.

C.7 Appearance Decoder Distillation

Teacher and student. The full appearance decoder is a nine-layer MLP with leaky-ReLU of negative slope 0.01 after every layer, including the six-channel output: lit RGB and shadowed RGB. Global distillation produces a width-128 6 layer decoder shared across assets after 2K steps during training. Separately, per-asset distillation compresses the full decoder into a four-layer student with the same activations for one reconstructed object. The teacher is the decoder associated with that object's reconstruction checkpoint; it is frozen and queried only on the object's own Gaussian features. Gaussian geometry, opacities, and appearance features remain fixed, and only the student MLP is optimized. This compression uses online teacher evaluations without image supervision.

Inputs and sampling. All decoders in this paper take the same 92-dimensional input. Directions are expressed in each Gaussian's own frame $(\mathbf{t}, \mathbf{b}, \mathbf{n})$, the columns of $\mathbf{R}(\mathbf{q}_i)$ (Section 3.4). The input concatenates a Fourier encoding with L=6 frequencies of the tangent-plane components of $\boldsymbol{\omega}_{\text{in}}$ and of $\boldsymbol{\omega}_{\text{out}}$ (26 values each), a Fourier encoding with L=1 of the tangent-plane components of their half vector $\text{normalize}(\boldsymbol{\omega}_{\text{in}} + \boldsymbol{\omega}_{\text{out}})$ (6 values), the raw normal components $\boldsymbol{\omega}_{\text{in}} \cdot \mathbf{n}$ and $\boldsymbol{\omega}_{\text{out}} \cdot \mathbf{n}$ (2 values), and the 32-dimensional appearance latent, where $\text{Fourier}_L(x) = [\, x,\ \sin(2^k \pi x),\ \cos(2^k \pi x)\,]_{k=0}^{L-1}$ is applied per component. Each distillation step draws $B = 65{,}536$ fresh samples: $\boldsymbol{\omega}_{\text{in}}$ and $\boldsymbol{\omega}_{\text{out}}$ independently uniform on the sphere, and latents drawn with replacement from the object's Gaussians with probability proportional to opacity.

Distillation objective. Let g_ϕ denote the student and f_ψ the frozen teacher, each producing a six-channel directional response. We minimize their mean absolute difference:

$$\mathcal{L}_{\text{distill}} = \frac{1}{6B} \sum_{b=1}^{B} \left\| g_\phi(\mathbf{z}_{i_b}, \boldsymbol{\omega}_{\text{in},b}, \boldsymbol{\omega}_{\text{out},b}) - f_\psi(\mathbf{z}_{i_b}, \boldsymbol{\omega}_{\text{in},b}, \boldsymbol{\omega}_{\text{out},b}) \right\|_1 . \tag{11}$$

We use Adam for 6K steps with a cosine learning-rate schedule from 10^{-3} to 10^{-5}, about 0.39G teacher evaluations per object. A student is accepted when the full object, shaded once with the teacher and once with the student and composited by the same tracer from two cameras, matches the teacher's rendering within at least 40 dB PSNR; typical values are 48–49 dB. At rendering time, the student replaces the teacher in Eq. 7, retaining both lit and shadowed responses and the same visibility blending.

C.8 Rendering Performance

We benchmark the complete gallery of 62 assets from four views each (248 images), rendered at 1024×1024 with 64 samples per pixel on an NVIDIA RTX A6000 GPU. The benchmark uses the OptiX FP32 rendering path, the fourth-order generalized Gaussian kernel with cutoff 3.0 (Appendix C.6), point-light appearance decoding, and no cooperative vectors. Table 9 summarizes rendering, asset loading, and geometry acceleration structure (GAS) setup times. Rendering takes 0.872 s per view on average, with a 95th-percentile time of 1.436 s.

Table 9: Gallery rendering performance at 1024×1024 and 64 spp on an RTX A6000. All times are in seconds. Rendering statistics are over 248 views; loading and GAS setup statistics are over 62 assets. P95 denotes the 95th percentile.

Stage	Total	Mean	Median	P95
Rendering (per view)	216.26	0.872	0.798	1.436
Asset loading (per asset)	112.27	1.811	1.614	3.291
OptiX/GAS setup (per asset)	90.04	1.452	1.422	1.860

C.9 Direct Per-Object Optimization

We compare the feed-forward reconstruction with direct per-object optimization against the same generated turntable (81 RGBA frames). Three representations are fitted with the same schedule against all 81 frames at 1024^2: 3000 Adam iterations, each on a batch of eight frames drawn from a shuffled pass over the sequence (about 300 passes over every frame), and an L1 color term with 0.2 D-SSIM plus an L1 alpha loss: (i) 3D Gaussians with baked SH color as in 3DGS (Kerbl et al., 2023) (same L1 + D-SSIM objective and SH degree 3; rasterized with gsplat, densified with MCMC instead of adaptive density control); (ii) 3D Gaussians with an rtRNG relightable decoder of our architecture, trained from scratch together with the Gaussians and rendered with our ray tracer; and (iii) test-time optimization (TTO) of the LRM's Gaussians and latent features with the shared decoder frozen.

Figure 12 compares a knit beanie and a dreamcatcher. The per-object fits obtain higher input-view PSNR, but this does not ensure coherent appearance away from the fitted observations. At the novel viewpoint, optimized Gaussians show fragmented geometry inside the hat and broken web strands. Under new illumination, the from-scratch decoder and TTO produce smeared knit and feather detail, while the LRM preserves finer structure. SH color bakes in the input illumination and cannot be relit.

C.10 Limitations

The video model sometimes produces inconsistent content across frames, which can lead to ghosting artifacts in reconstructed assets. Accurately reproducing sharp specular effects also remains challenging, as reported in prior neural appearance models (Kuznetsov et al., 2021; Mullia et al., 2024).

D LLM-Driven Prompt Generation Pipelinen

We used an AI assistant to generate candidate object descriptions for text-conditioned generation. The task prompt requested independently written descriptions of approximately 40–80 words, each describing a single, plausible real-world object with distinctive overall geometry and inherent geometric or material detail. It encouraged diversity across natural and manufactured subjects, including fibrous, thin, perforated, layered, and optically complex structures, while discouraging template-based expansion and near-duplicate subjects. The instructions excluded camera, lighting, animation, background, and rendering-style directions, since the observation protocol is prescribed by the fine-tuned generator. They also favored structures visible from lateral viewpoints, consistent with the horizontal turntable protocol.

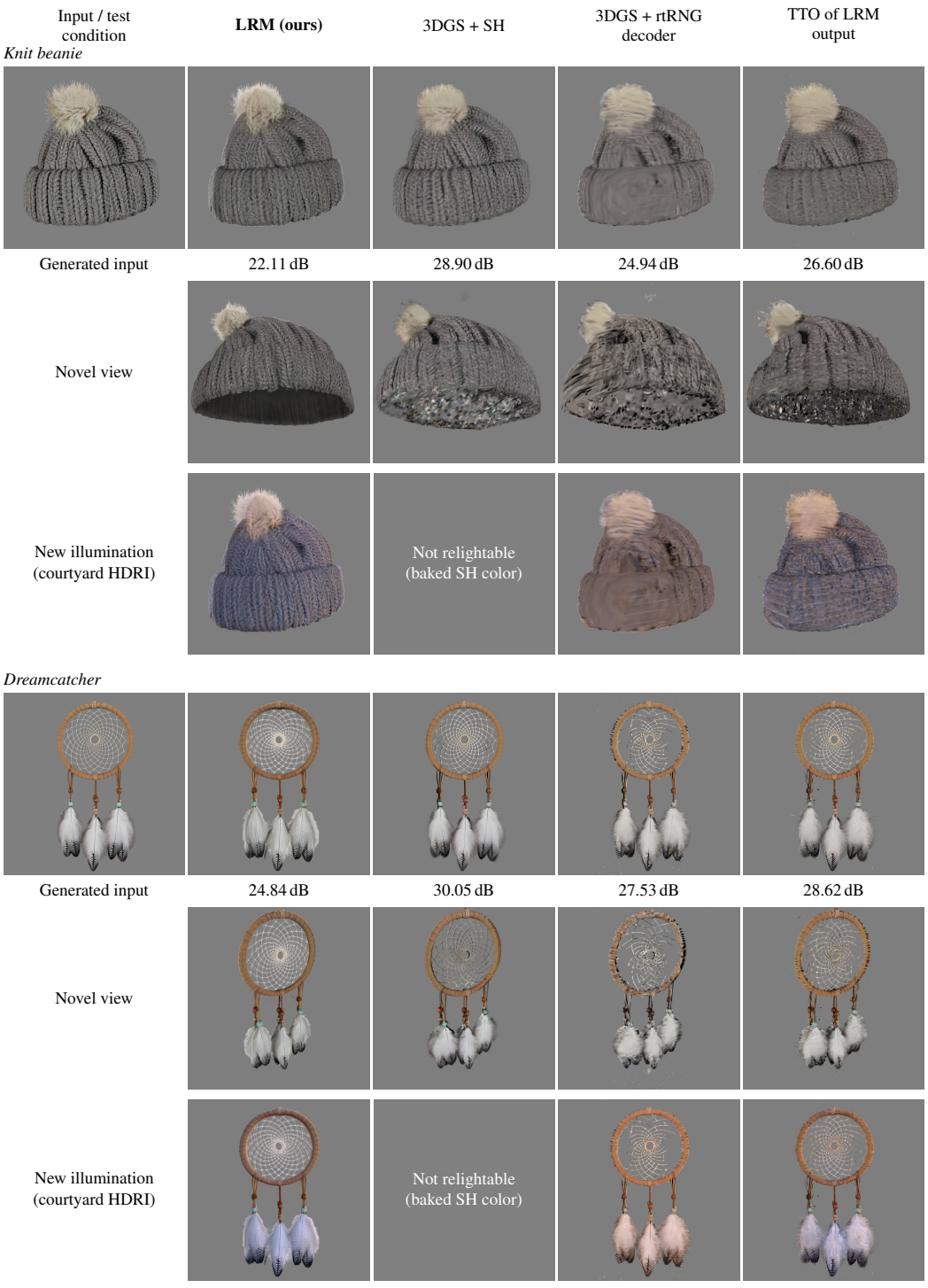

Figure 12: Direct per-object optimization on two generated objects. For each object, rows show an input view, a novel view, and new illumination, with matched cameras and lighting across methods. Numbers report full-frame PSNR over all 81 input frames. The first column shows generated observations, not ground truth. SH appearance cannot be relit.

Original prompt supplied to the AI assistant. The following instructions were used to request candidate descriptions; the requested subject properties do not themselves establish reconstruction capabilities.

```
Create 100 prompts for 3D object generation. Each prompt should describe
one interesting, realistic object with distinctive overall geometry and
rich, inherent medium-scale and fine-scale geometric or material detail.
Prefer subjects with intrinsic optical complexity, such as translucency,
subsurface scattering, layered fibers, sheen, iridescence, glossy or wet
coatings, or mixed opaque and transmissive regions, as well as other
features that are difficult to represent practically with traditional 3D
meshes and compact analytic material models.

1. Write each prompt individually.
Independently conceive and write every prompt. Do not use code, scripts,
template filling, synonym substitution, or combinatorial expansion to
generate or multiply prompt content. Do not pad the collection by
repeating the same description with a different name, species, color, or
material.

2. Describe only the object itself.
This is because the video model consuming the prompt is finetuned to
produce a turntable of only one object. Describe its identity, overall
form, proportions, component arrangement, intrinsic coloration,
materials, and visible geometric details. Each subject must be one
coherent object. A functional assembly with multiple connected parts
counts as one object.
Do not describe scenery, backgrounds, ground planes, surrounding props,
or added display stands. Integral components, including an object's base,
frame, or supports, are allowed if appropriate.
Avoid written text, logos, labels, and legible symbols unless they are
essential to the identity or function of the object, since they introduce
a separate text-rendering challenge.

3. Exclude all camera, lighting, and animation instructions. This is
because the video model consuming the prompt is finetuned to produce a
turntable of a static object under pre-defined pose, camera and light
conditions, orthogonal to the prompt.
Do not mention cameras, viewpoints, framing, lenses, camera movement,
trajectories, turntables, or shot sequences.
Do not describe lighting setups, light sources, cast shadows, moving
lights, or changes in lighting and reflections. Intrinsic material
properties, such as roughness, metallicity, or translucency, are allowed.
Do not describe actions, behaviors, movement, animation, deformation, or
changes over time. Use neutral descriptions of static anatomy, shape, and
component arrangement. Distinguish fixed geometry from an action: a
helical shell or downward-curving leaf is a shape, not a movement.
A static posture or fixed articulation may be specified when needed to
define an animal, plant, vehicle, tool, or mechanical object clearly, but
do not describe an action or motion.
When describing optical complexity, state the intrinsic material property
itself rather than a lighting setup or how the object changes under
moving illumination.
These restrictions are instructions for writing the prompts. Do not
repeat them inside the generated prompts, even as negative phrases such
as "no camera movement" or "no lighting changes."

4. Choose believable real-world subjects.
Select recognizable natural objects or human-made artifacts. Their
anatomy, proportions, materials, thicknesses, connections, and
construction must be plausible and internally consistent. Avoid
fantastical hybrids, impossible structures, arbitrarily attached parts,
and invented embellishments intended solely to increase complexity.

5. Make geometric richness inherent to the subject.
```

```
Choose objects whose actual anatomy, material structure, or functional
construction already provides interesting geometry. Describe specific
features and where they occur. Phrases such as "highly detailed,"
"intricate texture," are fine but should not be relied upon to create
complexity.
Prioritize distinctive medium-scale structures and finer surface relief
that can plausibly remain visible at the scale of the complete object. Do
not rely on hidden internal mechanisms or invisible microscopic features
to justify complexity.
Using more detail than typical for the given object is fine, but do not
artificially cover every surface with detail. Preserve naturally smooth
regions. Reject objects whose overall geometry is simple and whose
apparent complexity comes only from printed patterns, color variation, or
added decoration. Simple geometric components are acceptable when their
arrangement forms a genuinely rich overall structure, especially when
combined with material complexity.
Prefer subjects whose defining structures are visible across lateral
viewpoints. Avoid relying on geometry that is visible only from directly
above, directly below, or inside the object, since the primary
observation protocol is a horizontal turntable.

6. Maximize meaningful diversity across the collection.
Cover a broad range of natural and manufactured subjects, such as
terrestrial and marine animals, plants, minerals, foods, textiles, tools,
vehicles, instruments, toys, and mechanical devices.
Vary silhouettes, proportions, materials, component arrangements, and
types of geometric detail. Related subjects are acceptable when they
contribute substantially different geometry. Avoid near-duplicates and
excessive concentration in a few categories.
Balance the collection across different representation challenges,
including:
- fibrous, hairy, fuzzy, or feathery structures;
- thin, perforated, filamentary, or disconnected geometry;
- branching, layered, folded, frilled, or foliated forms;
- porous, cellular, woven, knitted, braided, or interlocking structures;
- translucent, subsurface-scattering, or mixed opaque and transmissive
materials;
- anisotropic, iridescent, pearlescent, glossy, wet, or otherwise
strongly view-dependent surfaces; and
- complex but plausible assemblies of multiple materials and connected
components.
Individual prompts may exercise several of these challenges. Do not
satisfy a category through superficial wording: the relevant structure or
material must be inherent to the chosen subject and described concretely.

7. Keep the output concrete and self-contained.
Write each prompt as a concise standalone paragraph of approximately
40-80 words, beginning with a clear object identity. Focus on its
defining structures instead of an exhaustive inventory of minor details.
Do not add photographic, cinematic, rendering-style, or quality tags.
Before responding, check that each prompt describes only the object,
contains no camera or lighting instructions or temporal events, and
contributes meaningful structural variety.
Output exactly 100 prompts, one unnumbered prompt per line, with no
bullets, category headings, explanations, blank lines, or additional
text.
```