Structured 3D Latents for Scalable and Versatile 3D Generation
(Supplementary Material)

Table I. Network configurations used in this paper. *SW* stands for "Shifted Window", *MSA* and *MCA* for "Multihead Self-Attention" and "Multihead Cross-Attention", and *Sp. Conv.* for "Sparse Convolution".

Network	#Layer	#Dim.	#Head	Block Arch.	Special Modules	#Param.
$\mathcal{E}_\mathrm{S}$	–	–	–	–	3D Conv. U-Net	59.3M
$\mathcal{D}_\mathrm{S}$	–	–	–	–	3D Conv. U-Net	73.7M
$\mathcal{E}$	12	768	12	3D-SW-MSA + FFN	3D Swin Attn.	85.8M
$\mathcal{D}_\mathrm{GS}$	12	768	12	3D-SW-MSA + FFN	3D Swin Attn.	85.4M
$\mathcal{D}_\mathrm{RF}$	12	768	12	3D-SW-MSA + FFN	3D Swin Attn.	85.4M
$\mathcal{D}_\mathrm{M}$	12	768	12	3D-SW-MSA + FFN	3D Swin Attn. + Sp. Conv. Upsampler	90.9M
$\mathcal{G}_\mathrm{S}$-B (text ver.)	12	768	12	MSA + MCA + FFN	QK Norm.	157M
$\mathcal{G}_\mathrm{S}$-L (text ver.)	24	1024	16	MSA + MCA + FFN	QK Norm.	543M
$\mathcal{G}_\mathrm{S}$-XL (text ver.)	28	1280	16	MSA + MCA + FFN	QK Norm.	975M
$\mathcal{G}_\mathrm{S}$-L (image ver.)	24	1024	16	MSA + MCA + FFN	QK Norm.	556M
$\mathcal{G}_\mathrm{L}$-B (text ver.)	12	768	12	MSA + MCA + FFN	QK Norm. + Sp. Conv. Downsampler / Upsampler + Skip Conn.	185M
$\mathcal{G}_\mathrm{L}$-L (text ver.)	24	1024	16	MSA + MCA + FFN	QK Norm. + Sp. Conv. Downsampler / Upsampler + Skip Conn.	588M
$\mathcal{G}_\mathrm{L}$-XL (text ver.)	28	1280	16	MSA + MCA + FFN	QK Norm. + Sp. Conv. Downsampler / Upsampler + Skip Conn.	1073M
$\mathcal{G}_\mathrm{L}$-L (image ver.)	24	1024	16	MSA + MCA + FFN	QK Norm. + Sp. Conv. Downsampler / Upsampler + Skip Conn.	600M

A. More Implementation Details

A.1. Network Architectures

The networks used in our method primarily consist of transformers [30], augmented by a few specialized modules. The configurations and statistics for each network are listed in Tab. I. In particular, $\mathcal{E}_\mathrm{S}$ and $\mathcal{D}_\mathrm{S}$ compose the VAE designed for sparse structures, as discussed in Sec. 3.3 of the main paper. The remaining networks are also defined in the main paper. Below, we provide detailed descriptions of the architectures of the specialized modules introduced.

3D convolutional U-net. The VAE for sparse structures ($\mathcal{E}_\mathrm{S}$ and $\mathcal{D}_\mathrm{S}$) is introduced to enhance the efficiency of the structure generator $\mathcal{G}_\mathrm{S}$ and to convert the binary grids of active voxels into continuous latents for flow training. Its architecture is similar to the VAEs in LDM [25], but it employs 3D convolutions and omits self-attention metchanisms. $\mathcal{E}_\mathrm{S}$ ($\mathcal{D}_\mathrm{S}$) consists of a series of residual blocks and downsampling (upsampling) blocks, reducing the spatial size from 64^3 to 16^3. The feature channels are set to 32, 128, 512 for spatial sizes of 64^3, 32^3, 16^3, respectively. The latent channel dimension is set to 8. We utilize pixel shuffle [27] in the upsampling block and replace group normalizations with layer normalizations.

3D shifted window attention. In the VAE for structured latents (SLAT), we employ 3D shifted window attention to facilitate local information interaction and improve efficiency. Specifically, we partition the 64^3 space into 8^3 windows, with tokens inside each window performing self-attention independently. Despite the potential variation in the number of tokens per window, this challenge can be efficiently addressed using modern attention implementations (*e.g.*, FlashAttention [6] and xformers [17]). The transformer blocks alternate between non-shifted window attention and window attention shifted by $(4, 4, 4)$, ensuring that the windows in adjacent layers overlap uniformly.

QK normalization. Similar to the challenges reported in SD3 [9], we encounter training instability caused by the exploding norms of queries and keys within the multi-head attention blocks. To mitigate this issue, we follow [9] to apply root mean square normalizations [33] (RMSNorm) to the queries and keys before sending them into the attention operators.

Sparse convolutional downsampler/upsampler. In $\mathcal{D}_\mathrm{M}$ and $\mathcal{G}_\mathrm{L}$, it is necessary to alter the spatial size of sparse tensors to increase the resolution of the SDF grid for meshes and to improve the efficiency of the SLAT generator, respectively. To achieve this, we employ downsampling and upsampling blocks equipped with sparse convolutions [31]. These blocks are composed of residual networks with two sparse convolutional layers, skip connections with optional linear mappings, and pooling or unpooling operators. We use average pooling and nearest-neighbor unpooling. For $\mathcal{G}_\mathrm{L}$, given that the structures of 64^3 are pre-determined, we only average the features from active voxels within each 2^3 pooling window and recover the 64^3 structures during unpooling. This is done by assigning values to active voxels from their nearest neighbors in the 32^3 space. For $\mathcal{D}_\mathrm{M}$, we simply subdivide each voxel into 2^3, resulting in a new sparse tensor with doubled spatial dimensions in each upsampling block.

A.2. Training Details

We provide more details about the training process for each model, including hyperparameter tuning, algorithm details, and loss function designs.

Sparse structure VAE. We frame the training of the sparse structure VAE as a binary classification problem, given the binary nature of the active voxels. Each decoded voxel is classified as either positive (active) or negative (inactive). Due to the imbalance between positive and negative labels, where active voxels are sparser than inactive ones, we adopt the Dice loss [21] to effectively manage this disparity.

Structured latent VAE. For the versatile decoding of SLAT, we implement decoders for various 3D representations, namely $\mathcal{D}_{\text{GS}}$ for 3D Gaussians [13], $\mathcal{D}_{\text{RF}}$ for Radiance Fields [20], and $\mathcal{D}_{\text{M}}$ for meshes. We provide detailed information on their respective training processes.

(a) 3D Gaussians. Following Mip-Splatting [32], we address aliasing by setting the minimal scale for Gaussians to $9e-4$ and the variance of the screen space Gaussian filter to 0.1. The value $9e-4$ is derived from the assumption of a 512^3 sampling rate within the $(-0.5, 0.5)^3$ cube. For each active voxel, 32 Gaussians are predicted (*i.e.*, $K = 32$ in the main paper). Since original density control schemes are not applicable when Gaussians are predicted by neural networks, we employ regularizations for volume [18] and opacity of the Gaussians to prevent their degeneration, specifically to avoid them becoming excessively large or transparent. The full training objective is:

$$\mathcal{L}_{\text{GS}} = \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{vol}} + \mathcal{L}_{\alpha}, \tag{I}$$

where $\mathcal{L}_{\text{recon}}$, $\mathcal{L}_{\text{vol}}$ and $\mathcal{L}_{\alpha}$ are defined below:

$$\begin{aligned} \mathcal{L}_{\text{recon}} &= \mathcal{L}_1 + 0.2(1-\text{SSIM}) + 0.2\text{LPIPS}, \\ \mathcal{L}_{\text{vol}} &= \frac{1}{LK}\sum_{i=1}^{L}\sum_{k=1}^{K}\prod \boldsymbol{s}_i^k, \\ \mathcal{L}_{\alpha} &= \frac{1}{LK}\sum_{i=1}^{L}\sum_{k=1}^{K}(1-\alpha_i^k)^2. \end{aligned} \tag{II}$$

(b) Radiance Fields. We predict 4 orthogonal vectors $\boldsymbol{v}_i^{\text{x}}, \boldsymbol{v}_i^{\text{y}}, \boldsymbol{v}_i^{\text{z}}, \boldsymbol{v}_i^{\text{c}}$ for each active voxel. These vectors represent the CP-decomposition [3] of a local 8^3 radiance volume $\boldsymbol{V} \in \mathbb{R}^{8\times8\times8\times4}$:

$$\boldsymbol{V}_{i,xyzc} = \sum_{r=1}^{R} \boldsymbol{v}_{i,rx}^{\text{x}} \boldsymbol{v}_{i,ry}^{\text{y}} \boldsymbol{v}_{i,rz}^{\text{z}} \boldsymbol{v}_{i,rc}^{\text{c}}. \tag{III}$$

The last dimension of $\boldsymbol{V}$, which has a size of 4, contains the color and density information. We set the rank $R = 16$. The recovered local volumes are then assembled according to the position of their respective active voxels, forming a 512^3 radiance field. Additionally, we implement an efficient differentiable renderer using CUDA, which enables real-time rendering by integrating sorting, ray marching, radiance integration, and the CP reconstruction into a single kernel. The training objective of $\mathcal{D}_{\text{RF}}$ is $\mathcal{L}_{\text{recon}}$ as defined in Eq. (II).

(c) Meshes. ***Erratum:*** We apologize for the typo in Eq. (4) in the main paper regarding the output attributes of $\mathcal{D}_{\text{M}}$. The correct dimensions for $\boldsymbol{w}_i^j$ and $\boldsymbol{d}_i^j$ should be 45 and 8, respectively. Below, we provide detailed information about the mesh decoding process.

We increase the spatial size of sparse structures from 64^3 to 256^3, by appending two aforementioned sparse convolutional upsamplers after the transformer backbone. For $\mathcal{D}_{\text{M}}$, although our primary focus is on shape (geometry), we also predict colors and normal maps for the meshes. As a result, the final output for each high-resolution active voxel is:

$$(\boldsymbol{w}_i^j, \boldsymbol{d}_i^j, \boldsymbol{c}_i^j, \boldsymbol{n}_i^j). \tag{IV}$$

Here, $\boldsymbol{w}_i^j = (\boldsymbol{\alpha}_i^j, \boldsymbol{\beta}_i^j, \gamma_i^j, \boldsymbol{\delta}_i^j)$ are the flexible parameters defined in FlexiCubes [26], where $\boldsymbol{\alpha}_i^j \in \mathbb{R}^8$ and $\boldsymbol{\beta}_i^j \in \mathbb{R}^{12}$ are interpolation weights per voxel, $\gamma_i^j \in \mathbb{R}$ is the splitting weights per voxel, and $\boldsymbol{\delta}_i^j \in \mathbb{R}^{8\times3}$ is per vertex deformation vectors of the voxel. In addition, $\boldsymbol{d}_i^j \in \mathbb{R}^8$ is the signed distance values for the eight vertices of the voxel, $\boldsymbol{c}_i^j \in \mathbb{R}^{8\times3}$ denotes vertex colors, and $\boldsymbol{n}_i^j \in \mathbb{R}^{8\times3}$ represents vertex normals. Since each vertex is connected to multiple voxels, we derive the final vertex attributes (*i.e.*, $\boldsymbol{\delta}$, $\boldsymbol{d}$, $\boldsymbol{c}$, and $\boldsymbol{n}$) by averaging the predictions from all associated voxels.

To simplify implementation, we attach the sparse structure to a dense grid for differentiable surface extraction using FlexiCubes. For all inactive voxels in the dense grid, we set their signed distance values to 1.0 and all other associated attributes to zero. We then extract meshes from the 0-level iso-surfaces of the dense grid. For each mesh vertex, its associated attributes (*i.e.*, $\boldsymbol{c}$ and $\boldsymbol{n}$) are interpolated from those of the corresponding grid vertices. We utilize Nvdiffrast [16] to render the extracted mesh along with its attributes, producing a foreground mask $\boldsymbol{M}$, a depth map $\boldsymbol{D}$, a normal map $\boldsymbol{N}_m$ directly derived from the mesh, an RGB image $\boldsymbol{C}$, and a normal map $\boldsymbol{N}$ from the predicted normals. The training objective is then defined as follows:

$$\mathcal{L}_{\text{M}} = \mathcal{L}_{\text{geo}} + 0.1\mathcal{L}_{\text{color}} + \mathcal{L}_{\text{reg}}, \tag{V}$$

where $\mathcal{L}_{\text{geo}}$ and $\mathcal{L}_{\text{color}}$ are written as:

$$\begin{aligned} \mathcal{L}_{\text{geo}} &= \mathcal{L}_1(\boldsymbol{M}) + 10\mathcal{L}_{\text{Huber}}(\boldsymbol{D}) + \mathcal{L}_{\text{recon}}(\boldsymbol{N}_m), \\ \mathcal{L}_{\text{color}} &= \mathcal{L}_{\text{recon}}(\boldsymbol{C}) + \mathcal{L}_{\text{recon}}(\boldsymbol{N}). \end{aligned} \tag{VI}$$

Table II. Ablation study on timestep sampling distributions.

	Distribution	CLIP↑	$\text{FD}_{\text{dinov2}}$↓
Stage 1	$\text{logitNorm}(0,1)$	26.03	287.33
	$\text{logitNorm}(1,1)$	**26.37**	**269.56**
Stage 2	$\text{logitNorm}(0,1)$	**26.61**	242.36
	$\text{logitNorm}(1,1)$	**26.61**	**240.20**

Here, $\mathcal{L}_{\text{recon}}$ is defined identically to Eq. (II). Finally, $\mathcal{L}_{\text{reg}}$ consists of three terms:

$$\mathcal{L}_{\text{reg}} = \mathcal{L}_{\text{consist}} + \mathcal{L}_{\text{dev}} + 0.01\mathcal{L}_{\text{tsdf}}, \tag{VII}$$

where $\mathcal{L}_{\text{consist}}$ penalizes the variance of attributes associated with the same voxel vertex, $\mathcal{L}_{\text{dev}}$ is a regularization term defined in FlexiCubes to ensure plausible mesh extraction, and $\mathcal{L}_{\text{tsdf}}$ enforces the predicted signed distance values $\boldsymbol{d}$ to closely match the distances between grid vertices and the extracted mesh surface, helping to stablize the training process in its early stages.

Rectified flow models. We employ rectified flow models $\mathcal{G}_{\text{S}}$ and $\mathcal{G}_{\text{L}}$ for sparse structure generation and structured latent generation, respectively. During training, we alter the timestep sampling distribution, replacing the $\text{logitNorm}(0,1)$ distribution used in SD3 with $\text{logitNorm}(1,1)$. We evaluate their performance at each stage of our generation pipeline using the Toys4k dataset. As shown in Tab. II, the latter provides a better fit for our task and we set it as the default setting.

B. Data Preparation Details

Recognizing the critical importance of both the quantity and quality of training data for scaling up the generative models, we carefully curate our training data from currently available open-source 3D datasets to construct a high-quality, large-scale 3D dataset. Moreover, we employed state-of-the-art multimodal model, GPT4o [1], to caption each 3D asset, ensuring precise and detailed text descriptions. This facilitates accurate and controllable generation of 3D assets from text prompts. In the following sections, we will first briefly introduce each 3D dataset utilized, and then provide details about our data curation pipeline. In addition, we provide a comprehensive explanation of both the captioning process and our rendering settings.

B.1. 3D Datasets

Objaverse-XL [8]. Objaverse-XL is the largest open-source 3D dataset, comprising over 10 million 3D objects sourced from diverse platforms such as GitHub, Thingiverse, Sketchfab, Polycam, and the Smithsonian Institution. This extensive collection includes manually designed objects, photogrammetry scans of landmarks and everyday items, as well as professional scans of historic and antique artifacts. Despite its large scale, Objaverse-XL is quite noisy, containing a significant number of low-quality objects, such as those with missing parts, low-resolution textures, and simplified geometries. Therefore, we include only the objects from Sketchfab (also known as ObjaverseV1 [7]) and GitHub in our training dataset and perform a thorough filtering process to clean the dataset.

ABO [5]. ABO includes about 8K high-quality 3D models provided by Amazon.com. These models are designed by artists and feature complex geometries and high-resolution materials. The dataset encompasses 63 categories, primarily focusing on furniture and interior decoration.

3D-FUTURE [10]. 3D-FUTURE contains around 16.5K 3D models created by experienced designers for industrial production, offering rich geometric details and informative textures. This dataset specifically focuses on 3D furniture shapes designed for household scenarios.

HSSD [14]. HSSD is a high-quality, human-authored synthetic 3D scene dataset designed to test navigation agent generalization to realistic 3D environments. It includes a total of 14K 3D models, primarily assets of indoor scenes such as furniture and decorations.

Toys4k [28]. Toys4k contains approximately 4K high-quality 3D objects from 105 object categories, featuring a diverse set of object instances within each category. Since previous works have not utilized this dataset for training, we leverage it as our testing dataset to evaluate the generalization of our model.

B.2. Data Curation Pipeline

To ensure high-quality training data, we implement a systematic curation process. First, we render 4 images from uniformly distributed viewpoints around each 3D object. We then employ a pretrained aesthetic assessment model [1] to evaluate the quality of each 3D asset. More specifically, we assess the average aesthetic score across 4 rendered view for each 3D object. We empirically find this scoring mechanism can effectively identify objects with poor visual quality – those that receive low aesthetic scores typically exhibit undesirable characteristics such as minimal texturing or overly simplistic geometry. We visualize the distribution of aesthetic scores in each dataset in Fig. I, and further provide some examples in Fig. II to illustrate the correspondance between the quality of 3D assets and their aesthetic scores. By filtering out objects with average aesthetic score below a certain aesthetic score threshold (*i.e.*, 5.5 for Objaverse-XL and 4.5 for the other datasets), we maintain a high standard of geometric and textural complexity in our dataset. After filtering, there are about 500K high-quality

[1] https://github.com/christophschuhmann/improved-aesthetic-predictor

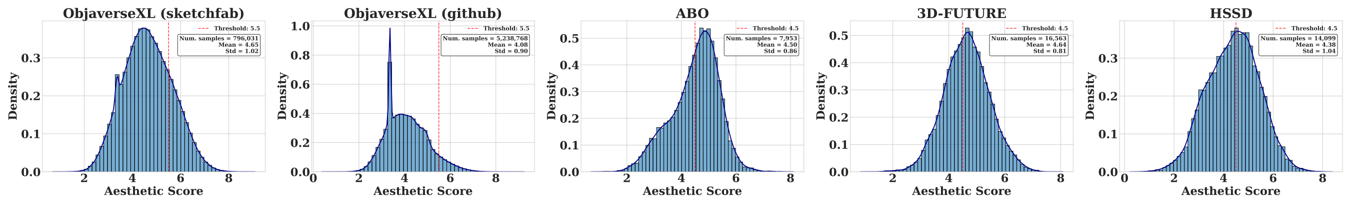


Figure I. Distribution of aesthetic scores in each dataset.

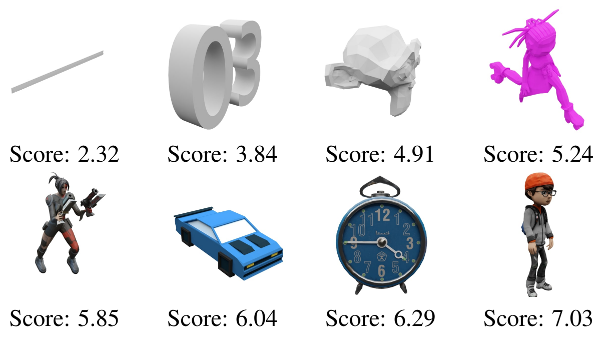


Figure II. 3D asset examples from Objaverse-XL with their corresponding aesthetic scores.

Table III. Composition of the training set and evaluation set.

Source	Aesthetic Score Threshold	Filtered Size
ObjaverseXL (sketchfab)	5.5	168307
ObjaverseXL (github)	5.5	311843
ABO	4.5	4485
3D-FUTURE	4.5	9472
HSSD	4.5	6670
All (training set)	–	500777
Toys4k (evaluation set)	4.5	3229

3D objects left (more details listed in Tab. III), which comprise our training dataset.

B.3. Captioning Process

Current available captions [19] for 3D objects either suffer from poor alignment with the objects they describe or lack detailed descriptions [11], which hinders high-quality text-to-3D generation. Therefore, we carefully design a captioning process following [11] to make the model generate precise and detailed text descriptions for each 3D object. To be more specific, we first employ GPT4o to produce a highly detailed description "<raw_captions>" of the input rendered images. Subsequently, GPT4o distills the crucial information from "<raw_captions>" into "<detailed_captions>", typically comprising no more than 40 words. Additionally, we summarize the "<detailed_captions>" into varying-length text prompts for augmentation in training. An illustration of the entire captioning process can be found in Fig. III, which also includes the prompts designed for GPT4o.

B.4. Rendering Process

For VAE training, we sample 150 cameras looking at the origin with a FoV of 40°, uniformly distributed across a sphere with a radius of 2. We render the assets using Blender, with a smooth area lighting. For the image-conditioned generation model, we render a different set of images with augmented FoVs ranging from 10° to 70°, which serves as image prompts during training.

C. More Experiment Details

C.1. Evaluation Protocol

In the main paper, we conduct quantitative comparisons and ablation studies using a series of numerical metrics. We provide detailed protocols for their calculation below.

Reconstruction experiments. We randomly sample a subset of 500 instances from the filtered Toys4k dataset, which comprises 3,229 3D assets (see Tab. III), as the evaluation set to assess the reconstruction fidelity of different latent representations. The evaluation is conducted in the following two aspects.

(a) Appearance fidelity. For each instance, we randomly sample one camera positioned on a sphere with a radius of 2, looking towards the origin with a FoV of 40°. We calculate PSNR and LPIPS between the rendered images from the reconstructed 3D assets and the ground truth images, and average the results as the final metrics. For 3DTopia-XL [4], which focuses on PBR materials, we report the reconstruction fidelity of albedo maps.

(b) Geometry accuracy. We employ Chamfer Distance (CD) and F-score of sampled point clouds to assess the overall geometry accuracy, as well as PSNR and LPIPS for rendered normal maps (*i.e.*, PSNR-N and LPIPS-N) to evaluate surface details. Definitions for the point cloud metrics are listed below:

- *Chamfer Distance:*

$$\begin{aligned} \mathrm{CD}(\boldsymbol{X}, \boldsymbol{Y}) = & \frac{1}{|\boldsymbol{X}|} \sum_{\boldsymbol{x} \in \boldsymbol{X}} \min_{\boldsymbol{y} \in \boldsymbol{Y}} \|\boldsymbol{x} - \boldsymbol{y}\|_2 \\ & + \frac{1}{|\boldsymbol{Y}|} \sum_{\boldsymbol{y} \in \boldsymbol{Y}} \min_{\boldsymbol{x} \in \boldsymbol{X}} \|\boldsymbol{y} - \boldsymbol{x}\|_2 . \end{aligned} \tag{VIII}$$

- *F-score:*

$$
\begin{aligned}
\mathrm{FN} &= \sum[\min_{\boldsymbol{y}\in\boldsymbol{Y}}\|\boldsymbol{x}-\boldsymbol{y}\|_2 > r],\\
\mathrm{FP} &= \sum[\min_{\boldsymbol{x}\in\boldsymbol{X}}\|\boldsymbol{y}-\boldsymbol{x}\|_2 > r],\\
\mathrm{TP} &= |\boldsymbol{Y}| - \mathrm{FP},\\
\mathrm{precision} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}},\\
\mathrm{recall} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}},\\
\textbf{F-score}(\boldsymbol{X},\boldsymbol{Y}) &= \frac{2\cdot\mathrm{precision}\cdot\mathrm{recall}}{\mathrm{precision}+\mathrm{recall}}.
\end{aligned}
\tag{IX}
$$

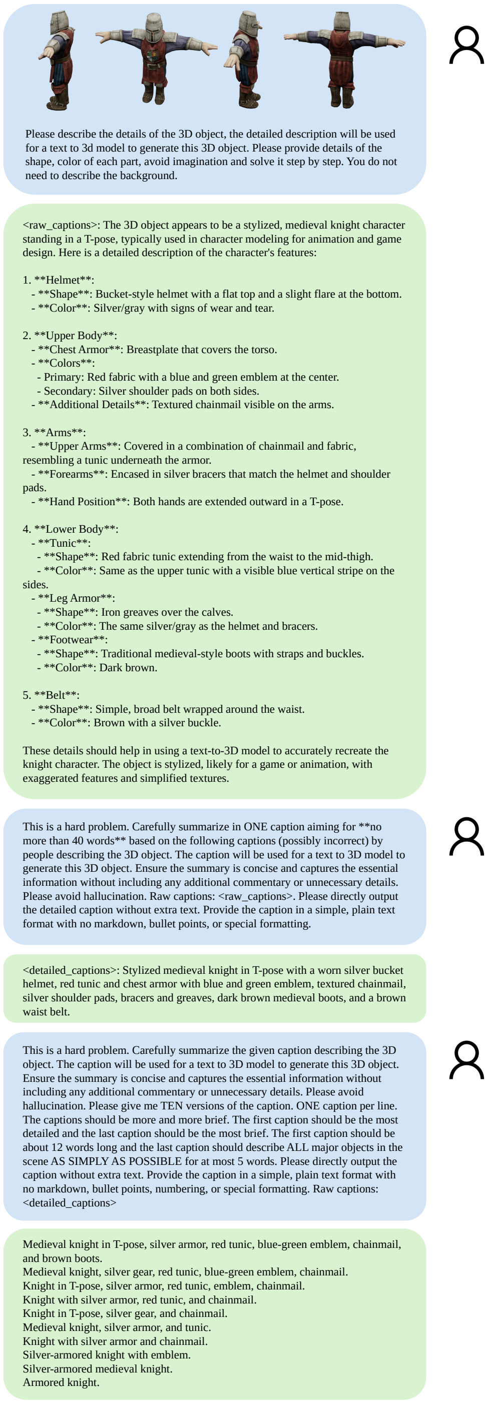
The point clouds used to assess the overall geometry accuracy (CD and F-score with $r = 0.05$) are sampled from the outer surface of the reconstructed meshes. Specifically, we render depth maps for each mesh from 100 uniformly sampled views, with camera settings identical to that for appearance evaluation. The depth maps are then unprojected to 3D points. We randomly sample 100K points from all the 3D points as the point clouds for evaluation.

For PSNR-N and LPIPS-N, as in the appearance metrics, we calculate the mean values across 500 image pairs (rendered results *v.s.* ground truth), with one pair per instance.

Generation experiments. For comparisons and ablation studies regarding generation quality, we utilize two evaluation sets: a subset of Toys4k with 1,250 randomly sampled instances and a subset of the training set with 5,000 instances. We employ Fréchet Distance (FD) [12] and Kernel Distance (KD) [2] with various feature extractors (*i.e.*, Inception-v3 [29], DINOv2, and PointNet++ [23]) to assess the overall quality of the generated outputs. Additionally, the CLIP score [24] is used to evaluate the consistency between the generated results and the input prompts. For each prompt in the evaluation set, we generate one asset using the generation model and use these assets as the generated set for metrics calculation. We provide detailed calculations for each metric below.

(a) Appearance quality. We employ $\mathrm{FD}_{\mathrm{incep}}$, $\mathrm{KD}_{\mathrm{incep}}$, $\mathrm{FD}_{\mathrm{dinov2}}$, and $\mathrm{KD}_{\mathrm{dinov2}}$ as evaluation metrics. For each instance, we render 4 views using cameras with yaw angles of $\{0°, 90°, 180°, 270°\}$, and a pitch angle of 30°. All other camera settings are consistent with those in the reconstruction experiments. The rendered images are then used to calculate different metrics. For Toys4k, we use 5,000 images each for both the real and rendered sets, while for the training set, we use 20,000 images.

(b) Geometry quality. We utilize $\mathrm{FD}_{\mathrm{point}}$. Following Point-E [22], we prepare the point clouds by sampling 4,000 points from unprojected multiview depth maps using the farthest point sampling technique.



```
Please describe the details of the 3D object, the detailed description will be used
for a text to 3d model to generate this 3D object. Please provide details of the
shape, color of each part, avoid imagination and solve it step by step. You do not
need to describe the background.
```

```
<raw_captions>: The 3D object appears to be a stylized, medieval knight character
standing in a T-pose, typically used in character modeling for animation and game
design. Here is a detailed description of the character's features:

1. **Helmet**:
  - **Shape**: Bucket-style helmet with a flat top and a slight flare at the bottom.
  - **Color**: Silver/gray with signs of wear and tear.

2. **Upper Body**:
  - **Chest Armor**: Breastplate that covers the torso.
  - **Colors**:
    - Primary: Red fabric with a blue and green emblem at the center.
    - Secondary: Silver shoulder pads on both sides.
  - **Additional Details**: Textured chainmail visible on the arms.

3. **Arms**:
  - **Upper Arms**: Covered in a combination of chainmail and fabric,
resembling a tunic underneath the armor.
  - **Forearms**: Encased in silver bracers that match the helmet and shoulder
pads.
  - **Hand Position**: Both hands are extended outward in a T-pose.

4. **Lower Body**:
  - **Tunic**:
    - **Shape**: Red fabric tunic extending from the waist to the mid-thigh.
    - **Color**: Same as the upper tunic with a visible blue vertical stripe on the
sides.
  - **Leg Armor**:
    - **Shape**: Iron greaves over the calves.
    - **Color**: The same silver/gray as the helmet and bracers.
  - **Footwear**:
    - **Shape**: Traditional medieval-style boots with straps and buckles.
    - **Color**: Dark brown.

5. **Belt**:
  - **Shape**: Simple, broad belt wrapped around the waist.
  - **Color**: Brown with a silver buckle.

These details should help in using a text-to-3D model to accurately recreate the
knight character. The object is stylized, likely for a game or animation, with
exaggerated features and simplified textures.
```

```
This is a hard problem. Carefully summarize in ONE caption aiming for **no
more than 40 words** based on the following captions (possibly incorrect) by
people describing the 3D object. The caption will be used for a text to 3D model to
generate this 3D object. Ensure the summary is concise and captures the essential
information without including any additional commentary or unnecessary details.
Please avoid hallucination. Raw captions: <raw_captions>. Please directly output
the detailed caption without extra text. Provide the caption in a simple, plain text
format with no markdown, bullet points, or special formatting.
```

```
<detailed_captions>: Stylized medieval knight in T-pose with a worn silver bucket
helmet, red tunic and chest armor with blue and green emblem, textured chainmail,
silver shoulder pads, bracers and greaves, dark brown medieval boots, and a brown
waist belt.
```

```
This is a hard problem. Carefully summarize the given caption describing the 3D
object. The caption will be used for a text to 3D model to generate this 3D object.
Ensure the summary is concise and captures the essential information without
including any additional commentary or unnecessary details. Please avoid
hallucination. Please give me TEN versions of the caption. ONE caption per line.
The captions should be more and more brief. The first caption should be the most
detailed and the last caption should be the most brief. The first caption should be
about 12 words long and the last caption should describe ALL major objects in the
scene AS SIMPLY AS POSSIBLE for at most 5 words. Please directly output the
caption without extra text. Provide the caption in a simple, plain text format with
no markdown, bullet points, numbering, or special formatting. Raw captions:
<detailed_captions>
```

```
Medieval knight in T-pose, silver armor, red tunic, blue-green emblem, chainmail,
and brown boots.
Medieval knight, silver gear, red tunic, blue-green emblem, chainmail.
Knight in T-pose, silver armor, red tunic, emblem, chainmail.
Knight with silver armor, red tunic, and chainmail.
Knight in T-pose, silver gear, and chainmail.
Medieval knight, silver armor, and tunic.
Knight with silver armor and chainmail.
Silver-armored knight with emblem.
Silver-armored medieval knight.
Armored knight.
```

Figure III. An example of our captioning process.

(c) Prompt alignment. We render 8 images per asset with yaw angles at every 45°, a pitch angle of 30°, and a radius of 2. We calculate the cosine similarity between the CLIP features of images from the generated assets and their corresponding text or image prompts. The average of all similarities ($\times 100$) is reported as the final CLIP score.

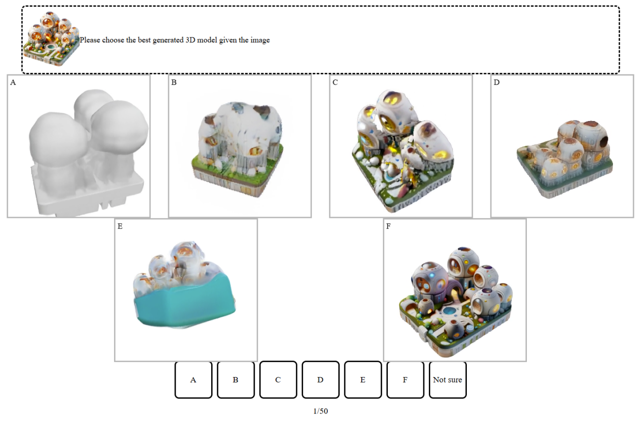


Figure IV. User interface used in our user study.

Table IV. Detailed statistics of the user study.

Method	Text-to-3D		Image-to-3D	
	Selections↑	**Perentage↑**	**Selections↑**	**Perentage↑**
Not Sure	56	4.2%	6	0.4%
Shap-E	42	3.1%	6	0.4%
LGM	70	5.2%	22	1.6%
InstantMesh	123	9.1%	30	2.2%
3DTopia-XL	5	0.4%	5	0.4%
Ln3Diff	9	0.7%	6	0.4%
GaussianCube	139	10.3%	–	–
Ours	**905**	**67.1%**	**1277**	**94.5%**
Total	1349	100%	1352	100%

C.2. User Study

We conducted a user study to evaluate the performance of various methods based on human preferences. Participants were presented with side-by-side comparisons of 3D assets generated by different methods. In each trial, they were given a text prompt or reference image, along with several rotating videos of candidate 3D assets generated using different techniques. The interface, as depicted in Fig. IV, displayed the reference image at the top, followed by options representing the generated 3D models. Participants were asked to select the model that best matched the reference image in terms of visual fidelity and overall quality, or they could choose *"Not sure"* if they were unable to make a decision. Each participant was assigned 50 trials, and their selections were recorded for analysis.

To ensure a diverse and unbiased evaluation, we implemented the following measures:

- The candidate 3D assets were not curated. Specifically, we sampled once per text or image prompt and used those samples directly in the study.
- The 50 trials for each participant were randomly selected from a pool of 68 text-to-3D cases and 67 image-to-3D cases. The order of candidates in each trial was also randomized.

We collected responses from 104 participants. In total, 2,701 trials were answered, with an average of 25.97 responses each. Detailed statistics are in Tab. IV.

D. More Results

D.1. 3D Asset Generation

We present additional examples of 3D assets generated by our method. These include more text-to-3D results with AI-generated prompts in Fig. V and more image-to-3D results from both AI-generated images (Fig. VI) and real world images (Fig. VII). For real-world images, we use segmented objects from SA-1B [15], which feature challenging materials, geometries, and camera views. Each 2×3 grid shows one generated asset, with front-left and back-right views in the top and bottom rows. Rendered images with 3D Gaussians (GS), Radiance Fields (RF), and meshes are displayed from left to right.

D.2. More Comparisons

In Fig. VIII, we provide additional comparisons of 3D assets generated by our method and those produced by alternative approaches described in the main paper.

Figure IX further compares our method with the commercial-level 3D generation model, Rodin Gen-1[2], using its default image-to-3D generation setting. Our method exhibits more detailed geometry structures on these complex cases, while being trained solely on open-source datasets and without commercial-specific designs.

D.3. 3D Editing

Figure X and XI present additional editing results, highlighting the flexible capabilities of our method to edit and manipulate 3D assets.

D.4. 3D Scene Composition

Figure XII and XIII provide two supplementary visualizations of complex scenes constructed with assets from our model, demonstrating its potential for production use.

E. Limitations and Future works

While our model demonstrates strong performance on 3D generation, it still has some limitations. First, it uses a two-stage generation pipeline for the structured latent representation, which first generates the sparse structures, followed by the local latents on them. This approach can be less efficient than end-to-end methods that create complete 3D assets in a single stage.

Second, our image-to-3D model does not separate lighting effects in the generated 3D assets, resulting in baked-in shading and highlights from the reference image. A potential improvement is to apply more robust lighting augmentation for image prompts during training and enforce the model to predict materials for Physically Based Rendering (PBR), which we leave for future exploration.

[2]https://hyperhuman.deemos.com/rodin

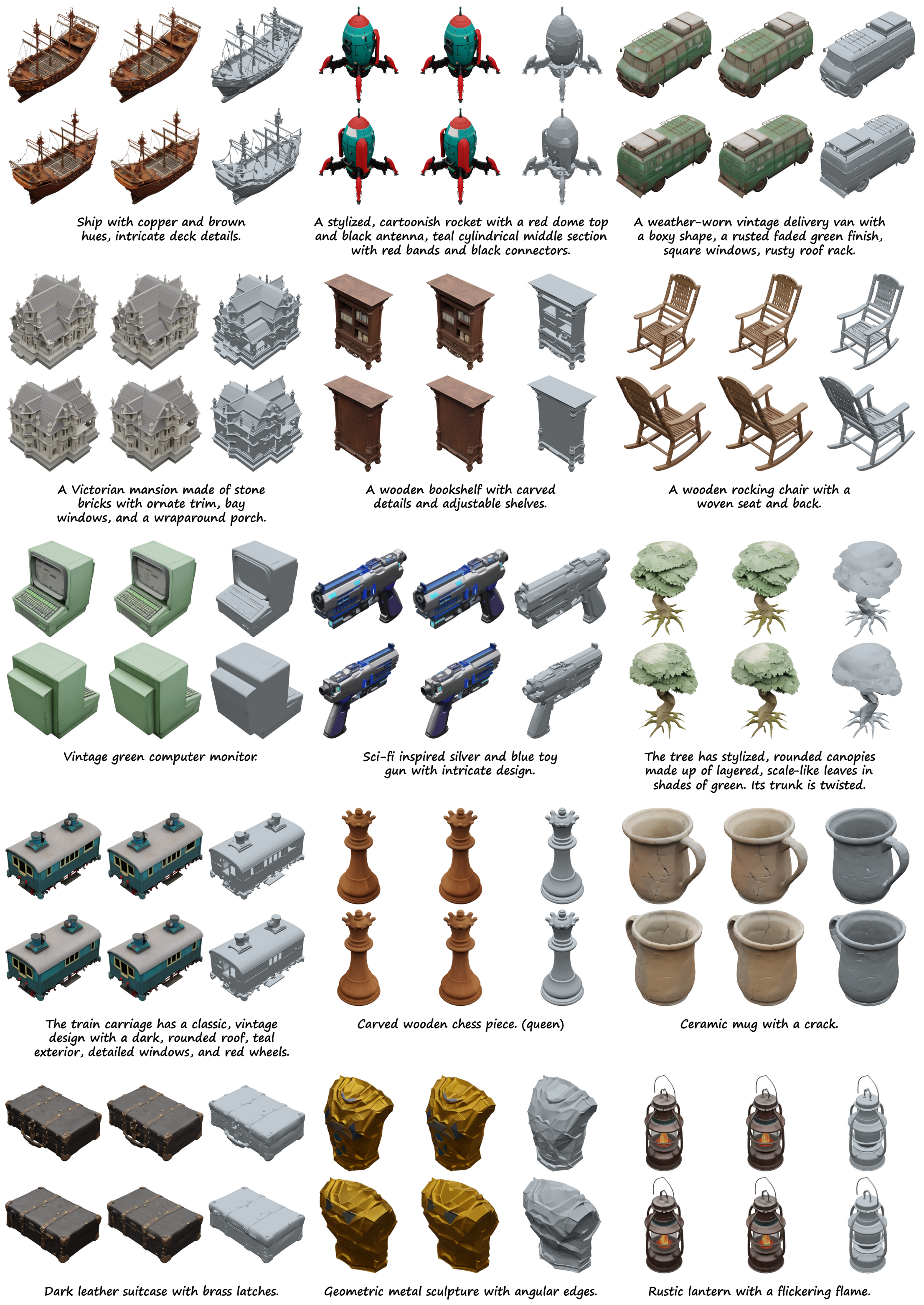

Ship with copper and brown hues, intricate deck details.
A stylized, cartoonish rocket with a red dome top and black antenna, teal cylindrical middle section with red bands and black connectors.
A weather-worn vintage delivery van with a boxy shape, a rusted faded green finish, square windows, rusty roof rack.
A Victorian mansion made of stone bricks with ornate trim, bay windows, and a wraparound porch.
A wooden bookshelf with carved details and adjustable shelves.
A wooden rocking chair with a woven seat and back.
Vintage green computer monitor.
Sci-fi inspired silver and blue toy gun with intricate design.
The tree has stylized, rounded canopies made up of layered, scale-like leaves in shades of green. Its trunk is twisted.
The train carriage has a classic, vintage design with a dark, rounded roof, teal exterior, detailed windows, and red wheels.
Carved wooden chess piece. (queen)
Ceramic mug with a crack.
Dark leather suitcase with brass latches.
Geometric metal sculpture with angular edges.
Rustic lantern with a flickering flame.

Figure V. More results generated by TRELLIS with AI-generated text prompts. (From left to right: GS, RF, and meshes)

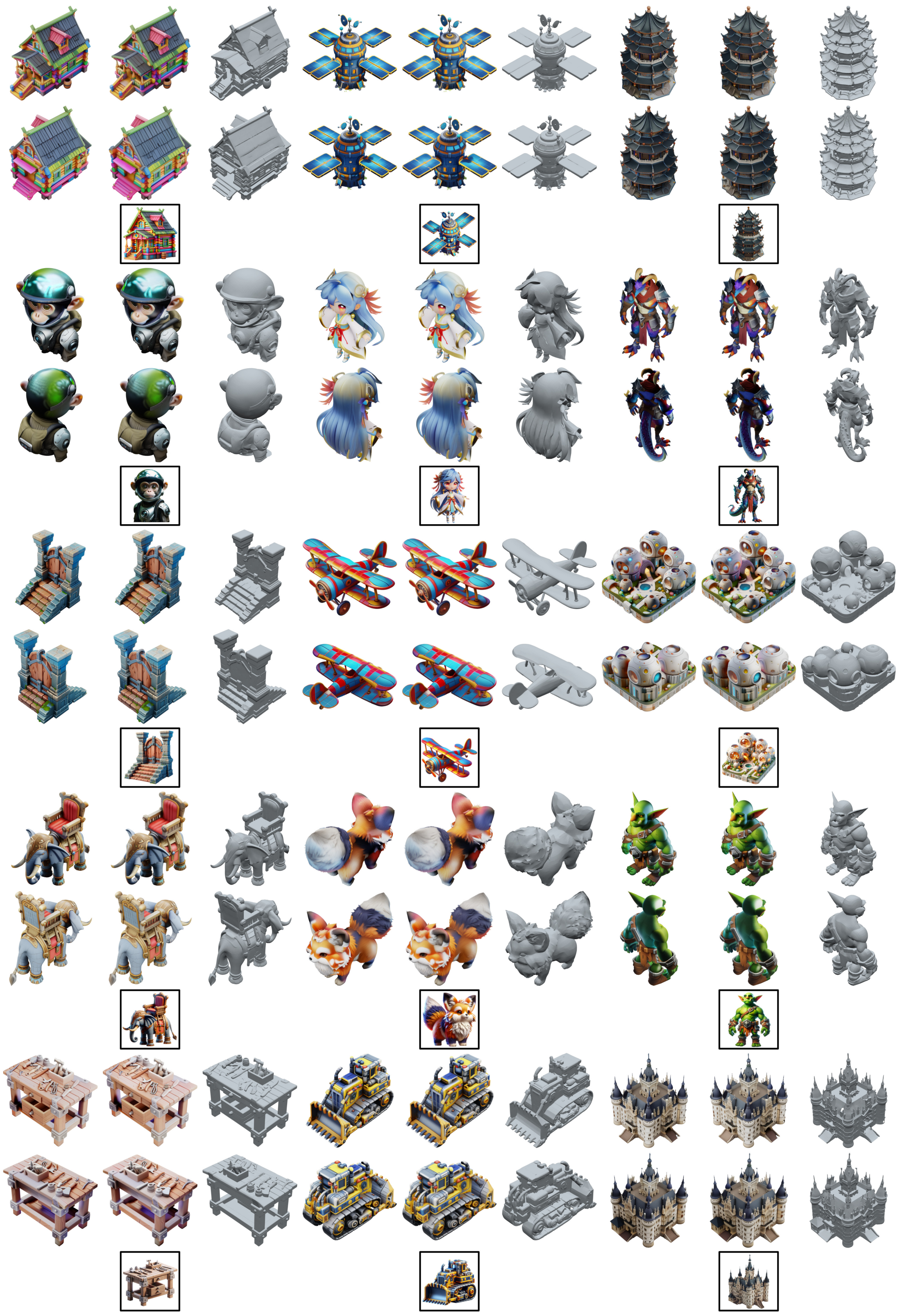


Figure VI. More results generated by TRELLIS with AI-generated image prompts. (From left to right: GS, RF, and meshes)

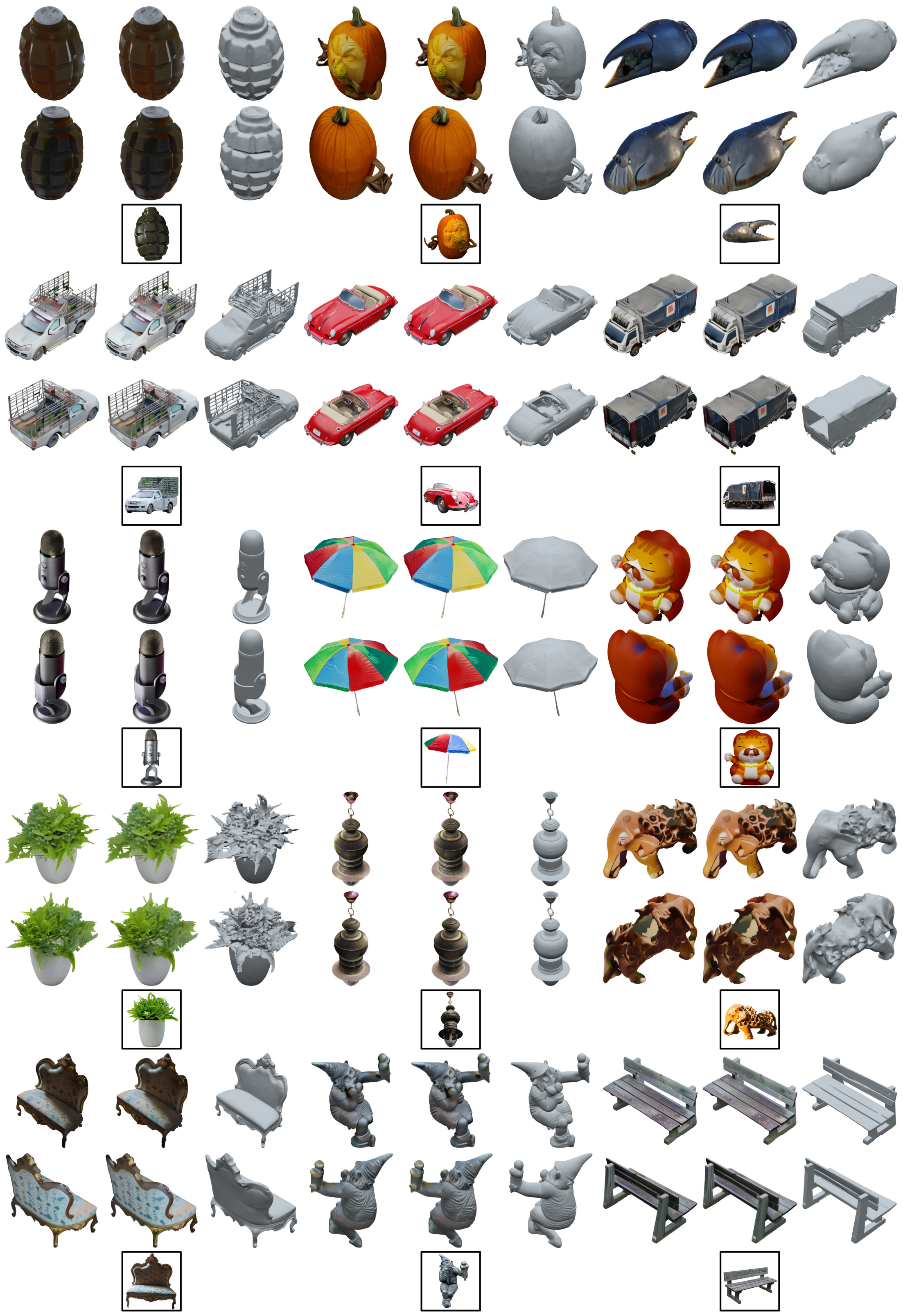


Figure VII. More results generated by TRELLIS with real-world image prompts from SA-1B. (From left to right: GS, RF, and meshes)

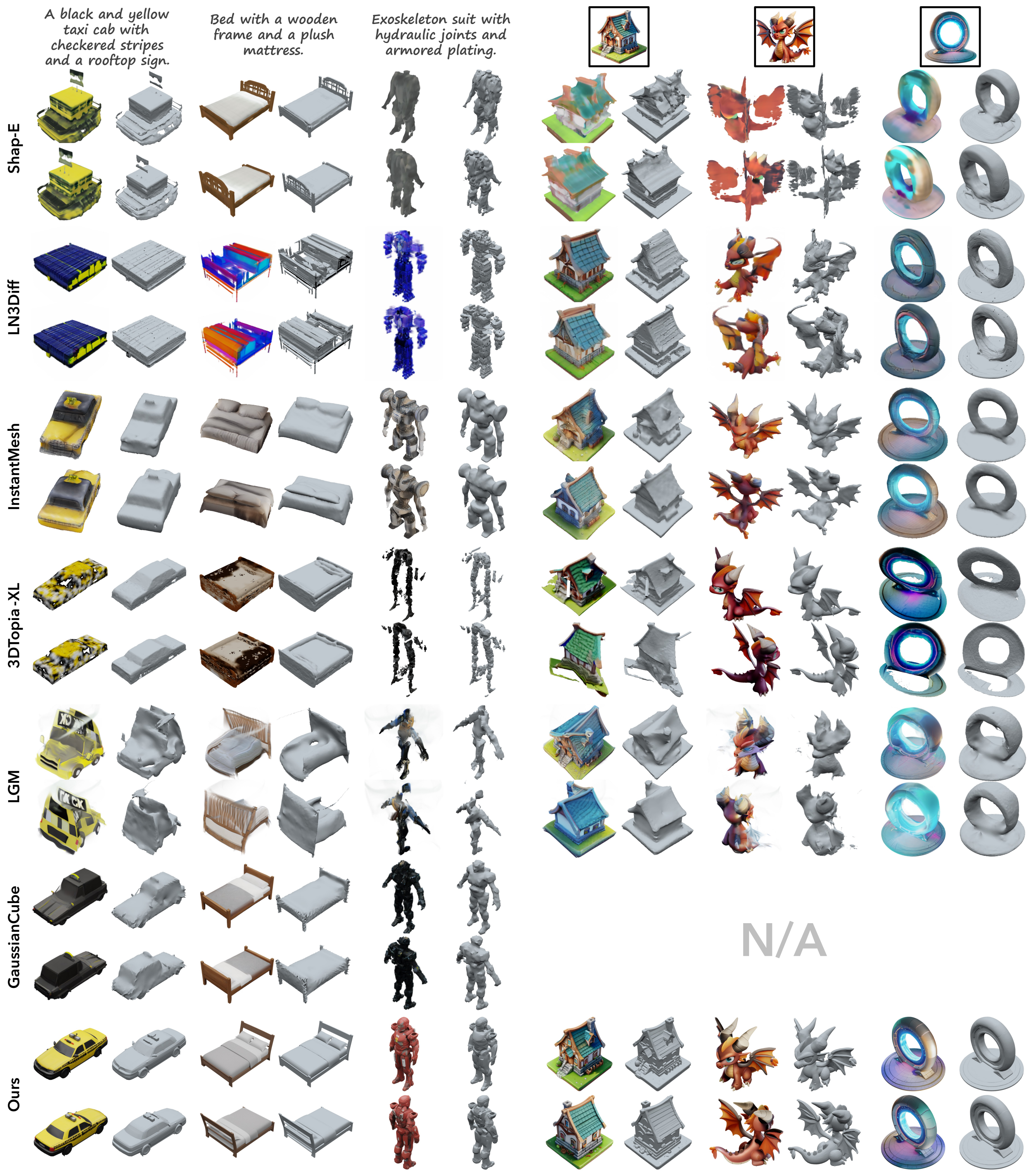

A black and yellow taxi cab with checkered stripes and a rooftop sign.
Bed with a wooden frame and a plush mattress.
Exoskeleton suit with hydraulic joints and armored plating.
Shap-E
LN3Diff
InstantMesh
3DTopia-XL
LGM
GaussianCube
N/A
Ours

Figure VIII. More comparisons of generated 3D assets by our method and prior works, with AI-generated text and image prompts.

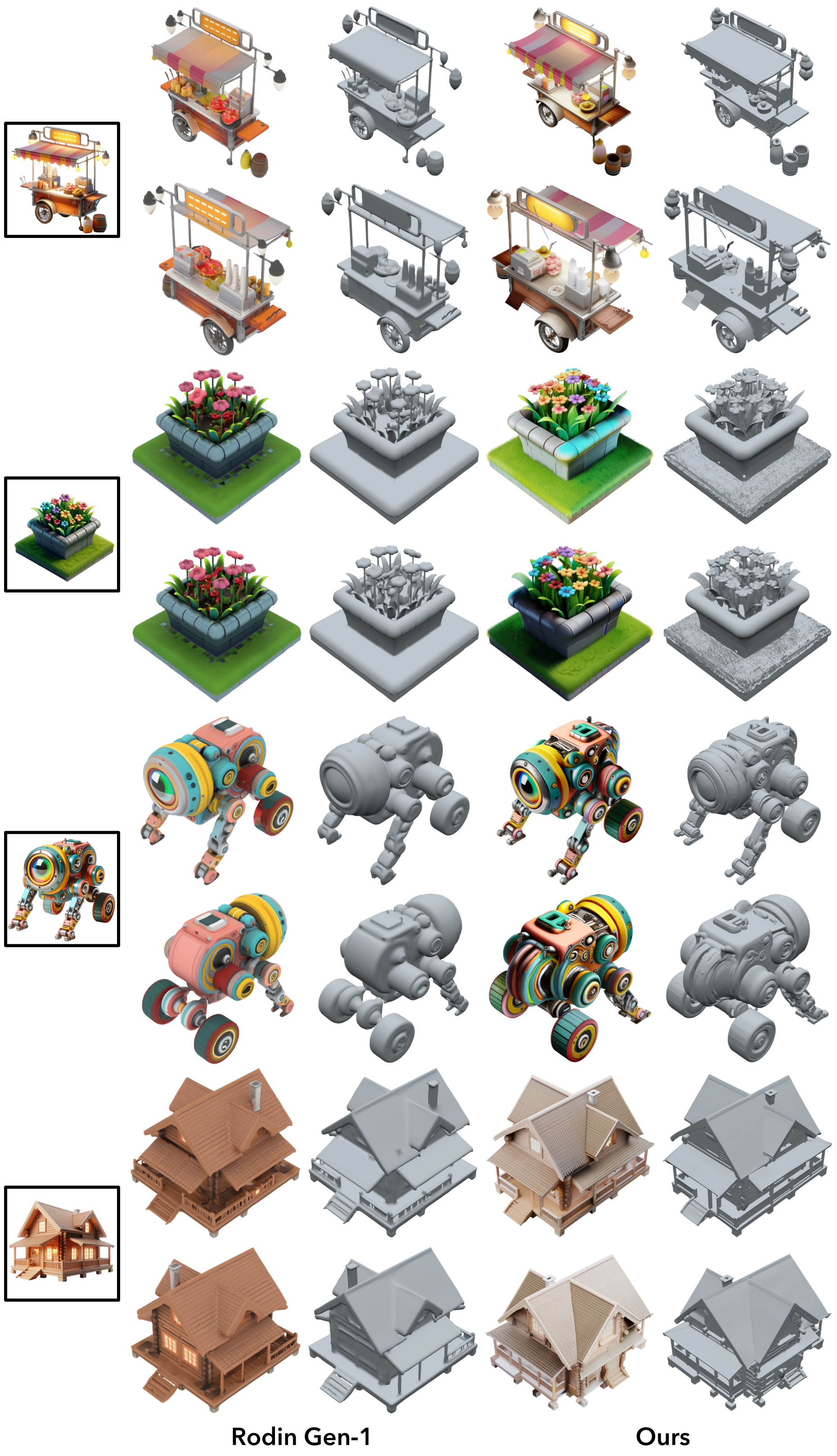


Figure IX. Comparisons between our method and a commercial-level 3D generation model, Rodin Gen-1 (with its default image-to-3D setting). Image prompts are generated by DALL-E 3. Our method exhibits more detailed geometry structures, while being trained solely on open-source datasets without commercial-specific designs.

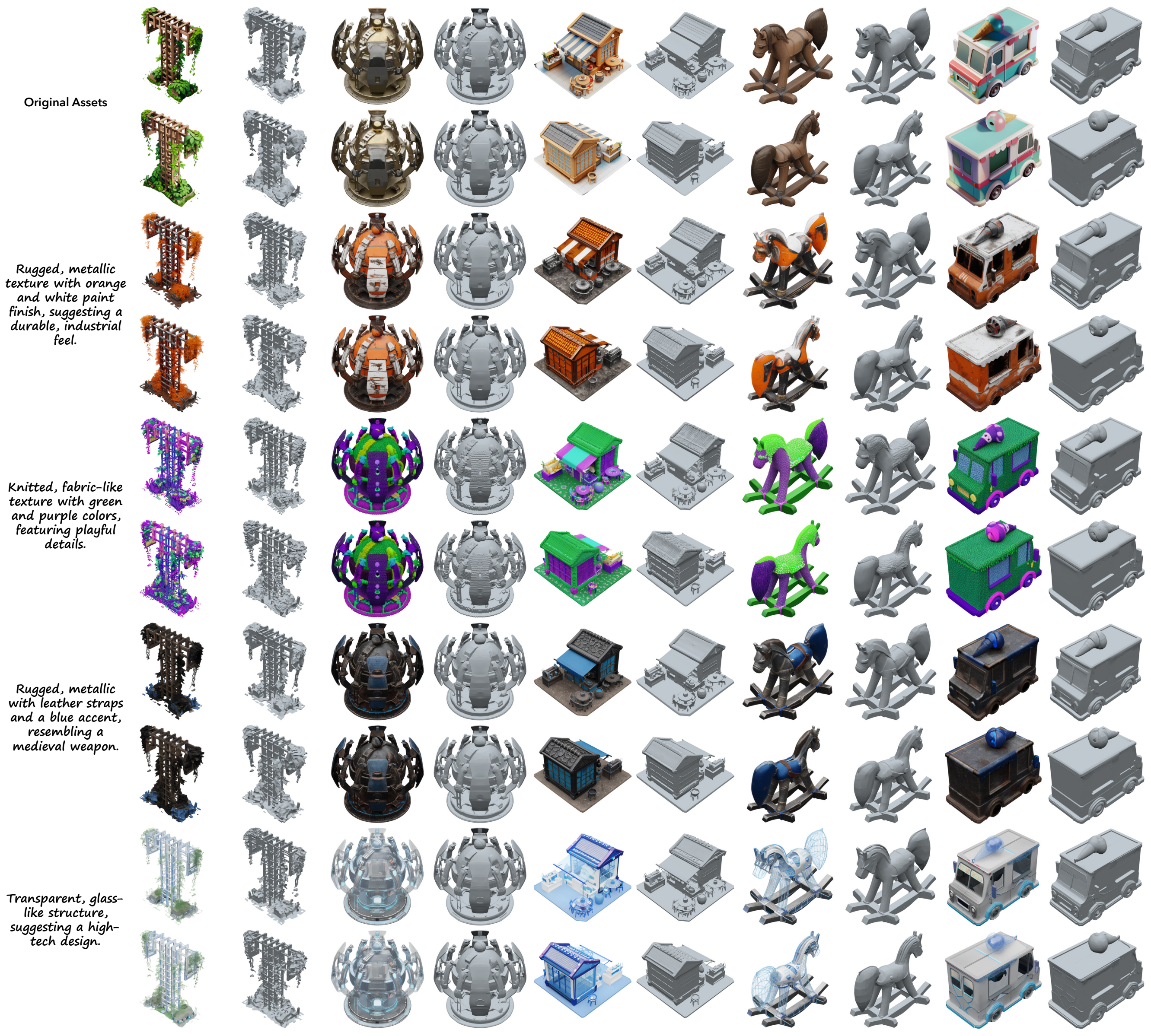

Original Assets
Rugged, metallic texture with orange and white paint finish, suggesting a durable, industrial feel.
Knitted, fabric-like texture with green and purple colors, featuring playful details.
Rugged, metallic with leather straps and a blue accent, resembling a medieval weapon.
Transparent, glass-like structure, suggesting a high-tech design.

Figure X. More examples of asset variations using TRELLIS. (Left: GS; Right: meshes)

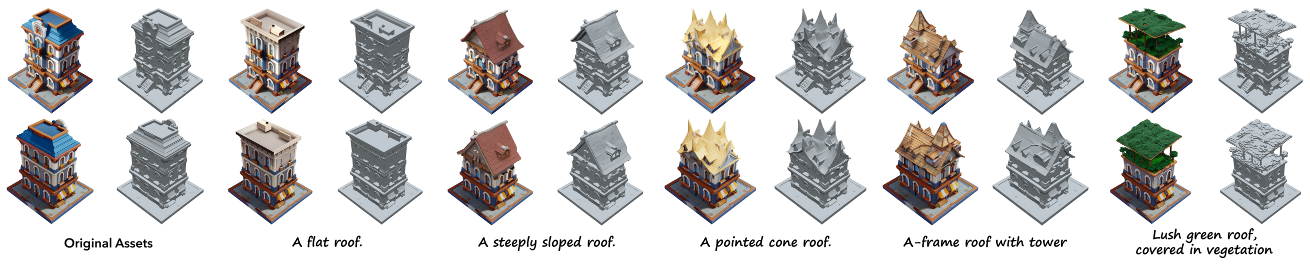


Figure XI. More examples of local editing, replacing the roof of the given building asset.

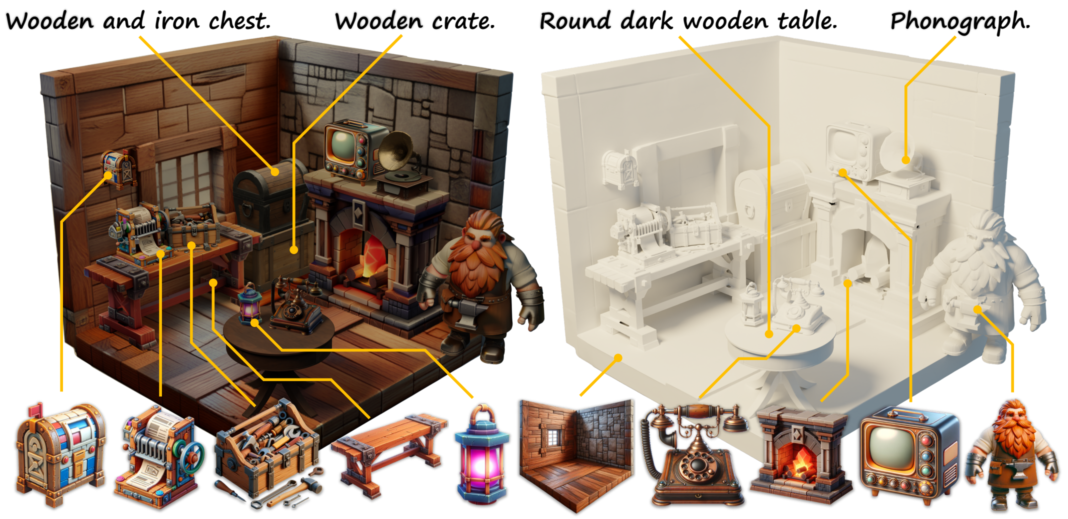


Figure XII. A dwarf blacksmith shop constructed with assets generated by TRELLIS. (*Text and image prompts are linked with yellow lines*)

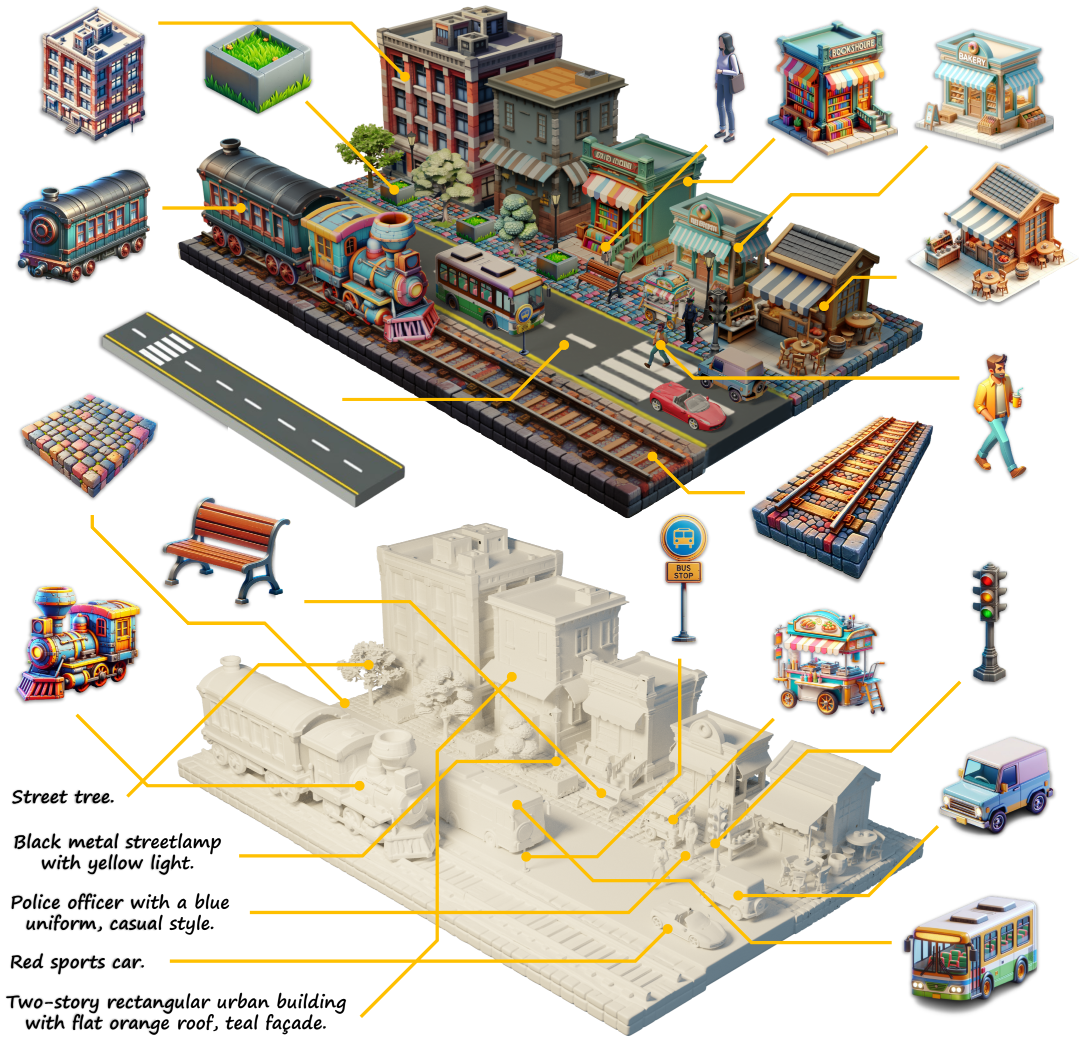

Street tree.
Black metal streetlamp with yellow light.
Police officer with a blue uniform, casual style.
Red sports car.
Two-story rectangular urban building with flat orange roof, teal façade.
BOOKSHOURE
BAKERY
BUS STOP

Figure XIII. A vibrant streetview constructed with assets generated by TRELLIS. (*Text and image prompts are linked with yellow lines*)

References

[1] Gpt-4o system card. 2024. 3

[2] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. In *International Conference on Learning Representations*, 2018. 5

[3] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European conference on computer vision*, pages 333–350. Springer, 2022. 2

[4] Zhaoxi Chen, Jiaxiang Tang, Yuhao Dong, Ziang Cao, Fangzhou Hong, Yushi Lan, Tengfei Wang, Haozhe Xie, Tong Wu, Shunsuke Saito, et al. 3dtopia-xl: Scaling high-quality 3d asset generation via primitive diffusion. *arXiv preprint arXiv:2409.12957*, 2024. 4

[5] Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, et al. Abo: Dataset and benchmarks for real-world 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21126–21136, 2022. 3

[6] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *ICLR*, 2024. 1

[7] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. 3

[8] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36, 2024. 3

[9] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024. 1

[10] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture. *International Journal of Computer Vision*, pages 1–25, 2021. 3

[11] Yunhao Ge, Xiaohui Zeng, Jacob Samuel Huffman, Tsung-Yi Lin, Ming-Yu Liu, and Yin Cui. Visual fact checker: Enabling high-fidelity detailed caption generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14033–14042, 2024. 4

[12] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 5

[13] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 2

[14] Mukul Khanna*, Yongsen Mao*, Hanxiao Jiang, Sanjay Haresh, Brennan Shacklett, Dhruv Batra, Alexander Clegg, Eric Undersander, Angel X. Chang, and Manolis Savva. Habitat Synthetic Scenes Dataset (HSSD-200): An Analysis of 3D Scene Scale and Realism Tradeoffs for ObjectGoal Navigation. *arXiv preprint*, 2023. 3

[15] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023. 6

[16] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics (ToG)*, 39(6):1–14, 2020. 2

[17] Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, Daniel Haziza, Luca Wehrstedt, Jeremy Reizenstein, and Grigory Sizov. xformers: A modular and hackable transformer modelling library. `https://github.com/facebookresearch/xformers`, 2022. 1

[18] Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. Mixture of volumetric primitives for efficient neural rendering. *ACM Transactions on Graphics (ToG)*, 40(4):1–13, 2021. 2

[19] Tiange Luo, Chris Rockwell, Honglak Lee, and Justin Johnson. Scalable 3d captioning with pretrained models. *Advances in Neural Information Processing Systems*, 36, 2024. 4

[20] 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. *Communications of the ACM*, 65(1):99–106, 2021. 2

[21] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. Ieee, 2016. 2

[22] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022. 5

[23] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 5

[24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 5

[25] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of*

the IEEE/CVF conference on computer vision and pattern recognition, pages 10684–10695, 2022. 1

[26] Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. Flexible isosurface extraction for gradient-based mesh optimization. *ACM Trans. Graph.*, 42(4), 2023. 2

[27] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016. 1

[28] Stefan Stojanov, Anh Thai, and James M Rehg. Using shape to categorize: Low-shot learning with an explicit shape bias. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1798–1808, 2021. 3

[29] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 5

[30] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 1

[31] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: octree-based convolutional neural networks for 3d shape analysis. *ACM Trans. Graph.*, 36(4), 2017. 1

[32] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19447–19456, 2024. 2

[33] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019. 1