

MeshReGen: A Unified 3D Geometry Regeneration Framework

Geon Yeong Park^{1,2}, Roman Shapovalov², Rakesh Ranjan², Jong Chul Ye¹,
Andrea Vedaldi² and Thu Nguyen-Phuoc²

¹ Korea Advanced Institute of Science and Technology (KAIST)

² Meta Reality Labs

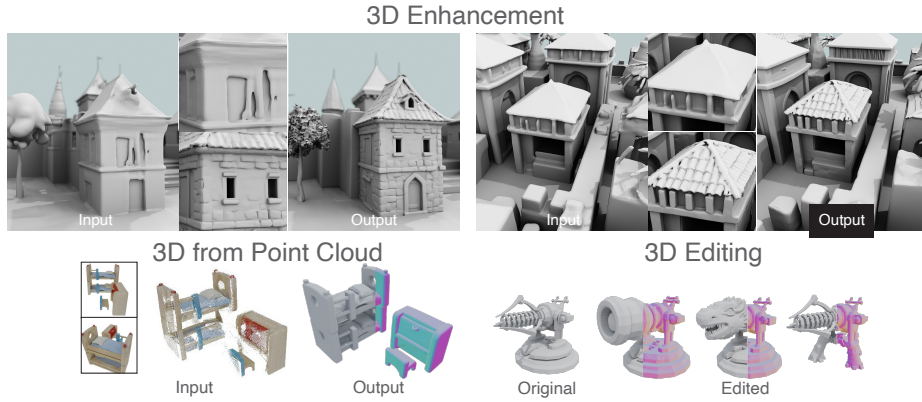


Fig. 1: MeshReGen is a diffusion-based 3D regeneration framework that reconstructs complete 3D shapes from coarse geometry using 2D image cues. The regeneration prior is learned through self-supervised pretext tasks and augmentations, without task-specific architectures or extra annotations. Users can input a coarse mesh (*3D Enhancement*), incomplete point cloud (*3D from Point Cloud*), or masked mesh (*3D Editing*) to recover detailed, complete 3D shapes. Please refer to the [project page](#) for more details.

Abstract. We consider the problem of *regenerating* 3D objects from 2D images and initial 3D shapes. Most 3D generators operate in a one-shot fashion, converting text or images to a 3D object with limited controllability. We introduce instead MeshReGen, a 3D **regenerator** that is conditioned on an initial 3D shape. This conceptually simple formulation allows us to support numerous useful tasks, including 3D enhancement, reconstruction, and editing. MeshReGen uses a new conditioning mechanism based on VecSet, which allows the regenerator to update or improve the input geometry with consistent fine-grained details. MeshReGen learns a widely applicable regeneration prior from off-the-shelf 3D datasets via self-supervised pretext tasks and augmentations, without additional annotations. We evaluate both the geometric consistency and fine-grained quality of MeshReGen, achieving state-of-the-art performance in controllable 3D generation across several tasks.

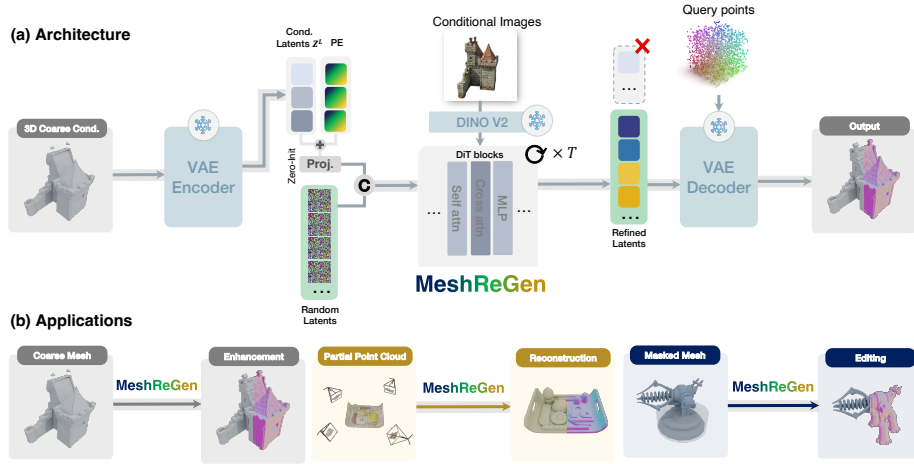


Fig. 2: Overview. MeshReGen takes both 2D image and initial 3D geometry as input, enabling explicit control over global geometry (e.g., pose, coarse shape) while improving fine-grained details. The 3D condition is encoded as VecSet latents (z^L) that compactly represent the global geometry. After taking the sum with the positional embeddings, these conditioning and random latents are diffused by a DiT into enhanced latents and then decoded into a complete, high-quality 3D shape.

1 Introduction

While the fidelity of 3D generators keeps improving, control is still mostly limited to providing a textual or image prompt, which is often insufficient in applications. For example, 3D artists often start with a 2D concept art that captures the 3D content that must be created, but are often given additional specifications of a different nature too. An example workflow may have the artist sketch the 3D shape using a 3D sculpting tool, followed by iterative refinement or editing. In other cases, the artist starts from a noisy scan (3D point cloud) of a real-life object, obtained using photogrammetry, and reconstructs that into a clean, faithful 3D shape that can be used in production. All these examples require the 3D generation process to be controlled by 3D signals of various types (sculpts, scans, block-outs) in addition to 2D signals (concept art).

Recent advances in 3D generation [56, 65, 77, 80] have focused on an image-based workflow, where a 3D object $\mathbf{x}^H \subset \mathbb{R}^3$ (a surface) is generated from a 2D image $I \in \mathbb{R}^D$.³ Hence, generation amounts to drawing a sample $\mathbf{x}^H$ from the distribution $p(\mathbf{x}^H | I)$. However, in the more elaborate workflows discussed above, it is necessary to control the generation process by an additional signal $\mathbf{y}^L$, like a sculpt, a block-out, or a 3D scan. While the nature and interpretation of these signals are widely different, there are some important similarities too. First, all such 3D signals can be represented as 3D shapes $\mathbf{y}^L \subset \mathbb{R}^3$. Secondly, these

³ This is true for text-based generation as well, as the text prompt is converted to a 2D image via a text-to-image model.

3D shapes $\mathbf{y}^L$ contain *significantly less or distorted information* than the target shape $\mathbf{x}^H$. For example, $\mathbf{y}^L$ may be (i) a coarse *but* geometrically aligned proxy (e.g., voxel grid or bounding box) or (ii) a degraded and potentially misaligned geometry (e.g., a noisy scan or a distorted intermediate generation). Hence, the common goal is to map a **Low**-information 3D shape $\mathbf{y}^L$ to a **High**-information 3D shape $\mathbf{x}^H$. In this paper, we thus ask whether, despite their different nature, all these tasks can be solved by a *single* architecture and learning formulation.

We call this common problem 3D shape *regeneration*. Formally, the goal of 3D regeneration is to draw a 3D shape $\mathbf{x}^H$ from a distribution $p(\mathbf{x}^H \mid \mathbf{y}^L, I)$ where I is a 2D image and $\mathbf{y}^L$ is an auxiliary 3D shape that may contain significantly less information than $\mathbf{x}^H$. Two key challenges arise in training general 3D regenerators: (a) developing a task-agnostic architecture that solves all these tasks, and (b) addressing the scarcity of paired 3D training datasets.

Prior work has considered specific instances of regeneration problems like editing [26] and enhancement [12] or considered controls that align accurately with the target shape like bounding boxes [20, 80] using task-specific architectures. It is less clear whether a single architecture can handle a wide variety of inputs, including those that contain distortion or poorly-aligned geometric information such as noisy scans, lossy-compressed assets, or outputs from low-quality 3D generators. Here, we seek a *unified architecture* that can handle all such cases.

To design such a general-purpose architecture, we start from VecSet [77], a powerful and highly flexible 3D latent space representation. VecSet is based on a pair of encoder and decoder functions $\mathbf{z} = \mathcal{E}(\mathbf{x})$ and $\mathbf{x} = \mathcal{D}(\mathbf{z})$, where $\mathbf{z} \in \mathbb{R}^{N \times d}$ is a set of N latent vectors of dimension d . In prior works, this latent space has been used to train latent conditional diffusion models, usually in the form of a Diffusion Transformer (DiT) [44] implementing a denoising function $\mathbf{z}' = \Phi(\mathbf{z}, I)$, where $\mathbf{z}$ is a noisy latent and I is a conditioning image. As we show in Sec. 3 and Fig. 2, we can adapt this architecture to solve the 3D regeneration problem by conditioning the diffusion process on the input coarse shape $\mathbf{y}^L$, also encoded in the same latent space $\mathbf{z}^L = \mathcal{E}(\mathbf{y}^L)$.

Rather than injecting 3D controls via separate cross-attention [20, 80], we encode both the conditioning shape $\mathbf{y}^L$ and the target shape $\mathbf{x}^H$ using the same VecSet encoder and concatenate their tokens as joint input to the DiT. This design minimally modifies the pretrained generator, allowing the 3D condition to be processed by the same self-attention mechanism used during pretraining. We show that this simple VecSet-level concatenation strategy empirically outperforms cross-attention and other conditioning mechanism alternatives.

Besides the architecture, the other significant challenge is the lack of *training data* for these tasks. Although large-scale 3D base datasets are now generally available [9], these datasets lack the annotations required to learn 3D regeneration models (namely pairs $(\mathbf{x}^H, \mathbf{y}^L)$). Another contribution of this paper is to introduce protocols that can automatically convert a generic 3D dataset into training data for 3D regeneration, where $\mathbf{y}^L$ is a degraded version of $\mathbf{x}^H$, obtained via data augmentation, without additional expensive annotations.

To show the generality of our approach, we experiment with three different tasks: **(a)** Enhancement: Given a very low-resolution or even corrupted 3D component of a large 3D scene, we upgrade its quality; **(b)** Editing: Given an initial 3D shape, we allow the model to change significant parts of it; **(c)** Reconstruction: Given an initial 3D scan obtained from a sparse set of images using photogrammetry, we recover a full and geometrically faithful 3D object. Importantly, all these tasks are solved using the same model architecture. Our experiments show that MeshReGen achieves state-of-the-art results even compared to architectures that are designed specifically for each task.

2 Related Work

Advances in 3D generative models. The increasing availability of large-scale 3D datasets [9, 10] has fueled progress in 3D generative modelling. Some methods generate 3D assets using only 2D supervision from multi-view rendered images [49, 67]. These approaches typically learn to generate multi-view images of the same object, which are used to reconstruct 3D volumetric representations such as radiance fields, tri-plane features, or Gaussian splats, from which a 3D mesh can be extracted using Marching Cubes [39]. However, the lack of 3D geometric priors can result in open surfaces or inaccurate interiors. In contrast, others [8, 25, 28, 31, 56, 65, 84, 85] learn to generate objects with clean and complete surfaces from data. Most of these 3D generators take text or image prompts as input, which restrains controllability. To address the latter, recent works introduce explicit 3D-aware controls such as voxels, bounding boxes, or other structured primitives (e.g., CLAY [80], PoseMaster [72], Hunyuan3D-Omni [19]).

3D generative priors for downstream tasks. Although 3D generative models have advanced, it is an open question whether the 3D generative priors they learn can be useful in different *downstream tasks*. While foundational *image* generators support editing, reconstruction, and other tasks within a single architecture [24, 55, 64, 76], downstream applications in 3D often rely on task-specific architectures or training-free adaptations [12, 26]. In this work, we explore generalized architectures among 3D applications and scalable construction of paired training data, with the goal of encouraging progress toward unified, generalizable, and versatile 3D regeneration priors.

3 Method

Let $\mathbf{x}^H \in \mathbb{R}^3$ denote the surface of a 3D object that we wish to generate, where the index stands for **H**igh information. In *3D regeneration*, we are interested in sampling $\mathbf{x}^H$ from a probability distribution $p(\mathbf{x}^H \mid \mathbf{y}^L, I)$ conditioned on a 3D shape $\mathbf{y}^L \in \mathbb{R}^3$ (“**L**ow information”) and, optionally, an image $I \in \mathbb{R}^D$. In practice, $\mathbf{x}^H$ and $\mathbf{y}^L$ may be represented by 3D meshes or point clouds with normals.

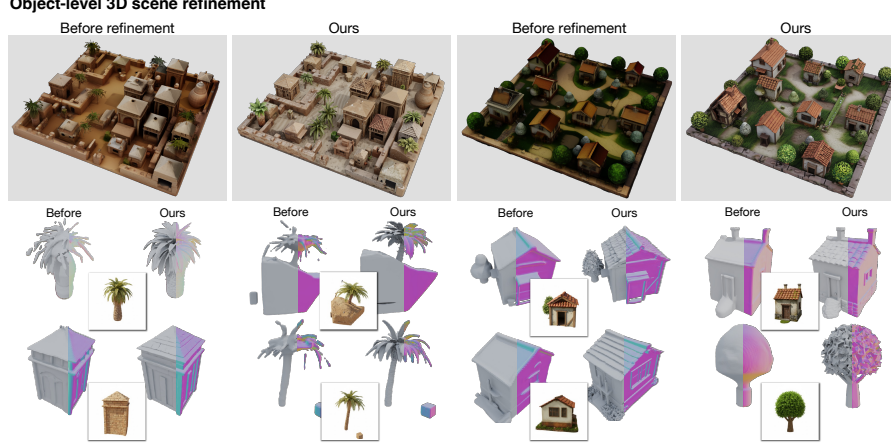


Fig. 3: Qualitative results of object-level 3D scene refinement. (Top) A single-scene image is converted into a coarse monolithic 3D scene by Sparc3D [32], where details of each object are degraded due to the limited capacity of the latent code. To mitigate this degradation, we segment the scene with AutoPartGen [6], refine individual objects with MeshReGen (Bottom), and update each object to produce a high-quality compositional 3D scene. Image conditions are automatically regenerated from initial renders of each coarse object, as detailed in Sec. 4.1, using an LLM-VLM. Each scene and object are textured using a proprietary texture generator for visualization only.

3.1 Model architecture

MeshReGen builds on a pre-trained latent diffusion model, operating in a compact latent space of 3D shapes [28, 30, 31, 65, 84], for which we use VecSet [77].

3D Generation using VecSet. The VecSet [77] encoder is a transformer neural network that maps a 3D shape $\mathbf{x} \in [-1, 1]^3$ to K D -dimensional latent vectors $\mathbf{z} = \mathcal{E}(\mathcal{P}; K) \in \mathbb{R}^{K \times D}$. It does so by sampling a point cloud $\mathcal{P} = \{(p_i, n_i)\}_{i=1}^N$ from the shape, where $p_i \in \mathbf{x}$ is a surface point and $n_i \in \mathbb{S}^2$ its corresponding normal. Then, it further subsamples $K \ll N$ points from $\mathcal{P}$ using Farthest Point Sampling (FPS) to initialise the tokens with, and then applies cross-attention to pool information from the full point cloud $\mathcal{P}$ into the tokens. The decoder $\mathcal{D}$, another transformer, evaluates the Signed Distance Function (SDF) $\mathcal{D}(q | \mathbf{z}) \in \mathbb{R}$ of the 3D shape $\mathbf{x}$ at any query location $q \in \mathbb{R}^3$ by attending to the latents $\mathbf{z}$.

Given this latent space, VecSet models the conditional distribution $p(\mathbf{z} | I)$ over latent codes instead of the distribution $p(\mathbf{x} | I)$ over 3D shapes. Following [28, 30], we do so by using a Diffusion Transformer (DiT) [44], which learns a denoiser neural network $\mathbf{z}_{t-1} = \Phi(\mathbf{z}_t | I, t)$ that takes as input a noised version $\mathbf{z}_t$ of the latent code $\mathbf{z}$ and partially denoises it to $\mathbf{z}_{t-1}$, conditioned on the image I and step t . The image I is tokenized using a pre-trained image encoder (DINOv2 [43]) and plugged into the DiT through cross-attention layers.

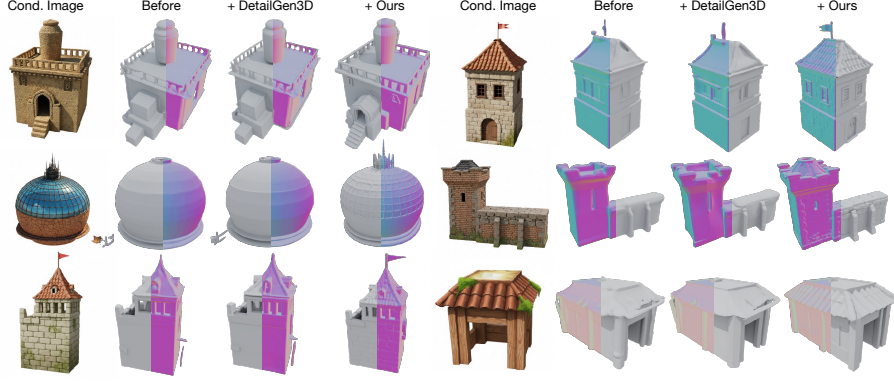


Fig. 4: Qualitative comparison with 3D generation and enhancement baselines. Coarse 3D conditions are obtained as parts of the original scene [31], and 2D image conditions are automatically regenerated given initial renders of each coarse object, as detailed in Sec. 4.1. Using these multi-modal conditions, MeshReGen regenerates the fine-grained details of 3D shapes, outperforming the recent enhancement baseline [12].

From Generation to Regeneration. We now show how to generalize the model above to sample the conditional distribution $p(\mathbf{x}^H | \mathbf{y}^L, I)$. As noted in Sec. 1, while the meaning of the conditioning signal $\mathbf{y}^L$ differs depending on the task, they can all be written as 3D shapes $\mathbf{y}^L \subset \mathbb{R}^3$. Hence, we propose to reuse VecSet representation for $\mathbf{y}^L$, and to learn the conditional distribution $p(\mathbf{z}^H | \mathbf{z}^L, I)$, where $\mathbf{z}^H$ and $\mathbf{z}^L$ are the latent codes of $\mathbf{x}^H$ and $\mathbf{y}^L$, respectively.

Utilizing the same latent space for both the conditioning and generated signals $\mathbf{z}^L$ and $\mathbf{z}^H$ opens up a very simple yet effective design for the DiT $\mathbf{z}_{t-1}^H = \Phi(\mathbf{z}_t^H | \mathbf{z}^L, I, t)$; namely, we can simply *concatenate* the conditioning tokens $\mathbf{z}^L$ to the noisy tokens $\mathbf{z}_t^H$ and pass them together as input to the DiT. In the experiments, we show that this is more efficient than alternative designs where other encodings are used, such as using cross-attention to plug in the conditioning tokens instead of concatenation.

Fewer tokens for coarser shapes. A useful property of VecSet is that the number of latent tokens K can be adjusted according to the complexity of the encoded shape $\mathbf{x}$. In particular, because the conditioning shape $\mathbf{y}^L$ contains less information, it can be encoded using much fewer tokens $\mathbf{z}^L = \mathcal{E}(\mathcal{P}^L; C)$ than the generated shape $\mathbf{x}^H$, where $C \ll K$. Due to the quadratic complexity of self-attention, this significantly improves the model’s efficiency. Note that sparse voxel grids [65], also popular in 3D generation, lack this property, as they require equally high voxel resolution to encode even a simple shape.

Token pre-processing. AutoPartGen [7] recently noted that concatenating two VecSets approximates merging the two underlying 3D shapes, which counteracts the regenerative effect we aim to achieve. We thus pre-process the conditioning tokens $\mathbf{z}^L$ before concatenation so that the model can better distinguish them

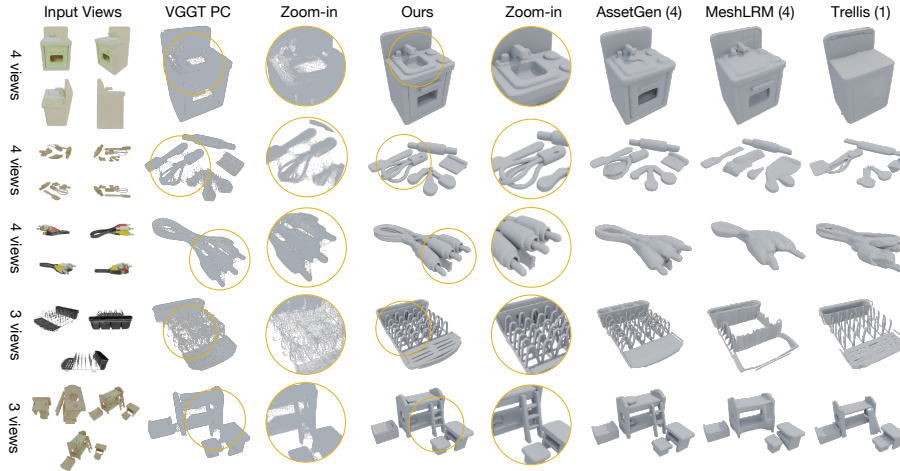


Fig. 5: Qualitative comparisons for faithful Image-to-3D generation on the GSO dataset [13]. MeshReGen preserves the geometry of the initial VGGT point cloud [58] while producing clean surfaces. Baselines that rely solely on image conditioning tend to be less faithful and smooth out fine details. For VGGT and MeshReGen, the number of input views is shown on the left; for the baselines, it is shown in parentheses.

from the noisy latent tokens $\mathbf{z}^H$. We do so by applying an MLP with zero-initialized weights, similar to ControlNet [81]. The zero initialization prevents “shocking” the model when new control tokens $\mathbf{z}^L$ are introduced. We also find it useful to add positional embeddings to $\mathbf{z}^L$ to help the model identify their spatial locations. Formally, we can write our diffusion step as

$$\mathbf{z}_{t-1}^H = \Phi\left(\mathbf{z}_t^H \oplus \text{MLP}(\mathcal{E}(\mathcal{P}^L; C) + \text{PE}(\hat{\mathcal{P}}^L)) \mid I, t\right),$$

where $\oplus$ denotes concatenation along the token dimension. Note that only $\mathbf{z}_t^H$ is denoised, whereas $\mathbf{z}^L$ remains fixed as a condition.

Training Data. In order to train the model to capture the conditional distribution $p(\mathbf{z}^H \mid \mathbf{z}^L, I)$, we require a dataset of triplets $\{(\mathbf{z}_i^H, \mathbf{z}_i^L, I_i)\}_{i=1}^n$ comprising the encodings of high- and low-quality 3D shapes $\mathbf{x}^H$ and $\mathbf{x}^L$ together with the (optional) corresponding guidance images I_i . The triplets are specific to the different tasks one wishes to solve (e.g., enhancement, reconstruction, editing). The existing 3D training datasets are already relatively small, and procuring task-specific paired data is much harder still. Hence, we develop data-engineering pipelines that construct such triplets automatically, starting from a collection of 3D objects $\mathbf{x}^H$. We detail this process for each target application in Sec. 3.2.

3.2 Applications

We introduce several example applications of MeshReGen to show that the same model architecture and 3D dataset (Sec. 3.3) can be used for multiple purposes.

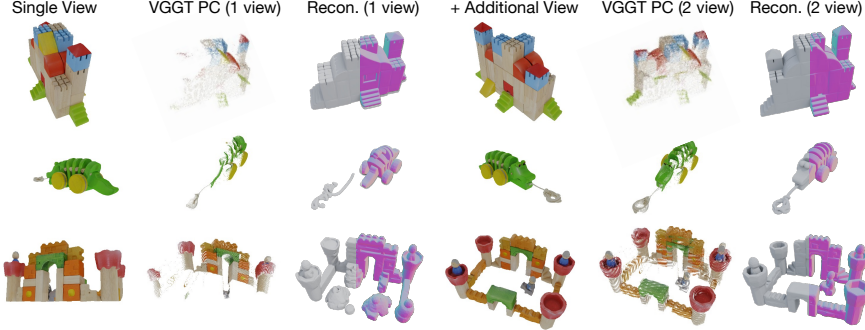


Fig. 6: Reconstruction results of VGGT+MeshReGen from one and two images. Note the improvement in the details on the side that is occluded in the first view. The trend continues with more views, as shown by metrics in Tab. 2.

Base dataset. We utilize the *same* base 3D dataset $\mathcal{D}$ for all applications and only change how the data is automatically augmented in pre-processing. The dataset comprises approximately 1M proprietary artist-authored 3D objects similar in spirit to Objaverse [10].

Compositional Shape Enhancement. In *shape enhancement*, we map a low-resolution 3D shape $\mathbf{y}^L$ to a high-resolution one $\mathbf{x}^H$. We apply this idea to a *compositional* setting [5, 7, 36, 73], where a large 3D object is first decomposed and then upgraded one component at a time. The reason is that 3D generators operate at a finite resolution, and even high-resolution ones like Sparc3D [31] fail to generate sufficient details for large 3D objects such as gaming levels [57].

Data Construction. To simulate the compositional (re)generation process, we start by arranging random ground-truth (and thus high-quality) 3D objects $\mathbf{x}^H \in \mathcal{D}$ in $k \times k$ 2D grids. These pseudo-compositions are then normalized (as a whole) in the $[-1, 1]^3$ cube and encoded using VecSet into tokens. Inspired by SDEdit [41], given a pre-trained DiT, we add noise to these tokens at step $t^* < T$ and denoise them back to $t = 0$. The 3D scene is then decoded from the denoised tokens and the degraded 3D shape $\mathbf{y}^L$ of each object is extracted from the resulting geometry. Increasing k means that more objects are represented by the same number of tokens, controlling how much information is removed per-object, which also determines the final number of tokens $\mathbf{z}^L$ we use to represent $\mathbf{y}^L$. $\mathbf{z}^H$ is instead obtained by encoding the original object $\mathbf{x}^H$ with the full token count. Please see the Appendix for examples.

Generality. While we simulate the degradation process due to generating multiple objects together, in practice, as we show in the Appendix, our technique results in pairs $(\mathbf{z}^H, \mathbf{z}^L)$ that are representative of degradations common in other applications too, more so than, say, applying generic 3D smoothing filters [16, 47, 52] to $\mathbf{x}^H$. As a result, the resulting model can also be potentially applied to a wide range of degradations, including low-resolution 3D assets (e.g., from old games) and models affected by lossy 3D compression.

Guided Image-to-3D. In *guided image-to-3D* we map an initial sparse but accurate 3D point cloud $\mathbf{y}^L$ reconstructed from one or more images $\{I_1, \dots, I_L\}$ into a full surface of a 3D object $\mathbf{x}^H$ faithful to the input data. We are motivated by the fact that current image-to-3D generators produce high-quality and detailed 3D shapes that are, however, not necessarily faithful to the input image and/or cannot account for more than one view as input. On the other hand, large 3D reconstruction neural networks like VGGT [58] and many others [23, 33, 61] reconstruct visible geometry as faithfully as possible from several views, resulting in 3D reconstructions $\mathbf{y}^L$ that are accurate but incomplete due to occlusions, lack details, and can be noisy.

Data Construction. Given a ground-truth 3D object $\mathbf{x}^H$, we render 1–4 random views $\{I_1, \dots, I_L\}$ and pass them to VGGT to obtain a sparse reconstruction $\mathbf{y}^L$. For rendering, camera poses are sampled uniformly in azimuth, focal length, and with elevations in $[-10^\circ, 30^\circ]$. In order to apply $\mathcal{E}$ to encode $\mathbf{y}^L$, we also require the normals for each surface point, which VGGT does not provide, so we estimate them using covariance analysis [42]. Because $\mathbf{y}^L$ is reconstructed up to scale and pose, we align it with $\mathbf{x}^H$. The conditioning image I is randomly chosen from the rendered views, shapes $\mathbf{x}^H$ and $\mathbf{y}^L$ encoded, and the resulting triplet $(\mathbf{z}^H, \mathbf{z}^L, I)$ added to the training dataset.

3D Shape Editing. In *3D shape editing*, the model maps the initial 3D shape $\mathbf{y}^L$ to modified shape $\mathbf{x}^H$ that reproduces $\mathbf{y}^L$ except for a specific 3D region M that is edited to match a guide image I . The 3D masks can be as simple as a box or ellipsoid that fully contain the region of geometry we want to edit. Following [14, 27], the image I is obtained automatically by (1) rendering a view of the original object $\mathbf{y}^L$, (2) masking the 2D region corresponding to the 3D edit M , and (3) using a text-to-image model [24] to follow the inpainting instructions.

Data Construction. We simulate the editing process as follows. The target 3D object $\mathbf{x}^H$ is sampled as usual from the base dataset $\mathcal{D}$, the image I is a render of this object, and $\mathbf{z}^H$ is the encoding of $\mathbf{x}^H$. It only remains to show how to construct the *source* shape $\mathbf{y}^L$ (and its code $\mathbf{z}^L$).

To this end, given $\mathbf{x}^H$, we first randomly sample an edit region M as a 3D mask. The 3D masks can be as simple as a box or cylinder containing the geometry we want to edit. Then we remove the original points within M and the corresponding normals from $\mathcal{P}^H$ to form $\mathcal{P}^L$. We obtain $\tilde{\mathbf{z}} = \mathcal{E}(\mathcal{P}^L; C - k)$, resulting in $C - k$ tokens instead of all C tokens. The remaining k edit tokens are used to tell the model where the edit should be applied and are initialized as learnable mask positional embeddings $\mathbf{z}^M$ in latent space. The final encoding for the degraded shape $\mathbf{y}^L$ is the concatenation $\mathbf{z}^L = \tilde{\mathbf{z}} \oplus \mathbf{z}^M$ of $\tilde{\mathbf{z}}$ with k learnable mask tokens $\mathbf{z}^M$. We thus obtain the final training triplet $(\mathbf{z}^H, \mathbf{z}^L, I)$.

The number of edit tokens k represents the editing capacity. It is adaptively set as $k = \max\{[C \cdot r], k_{\min}\}$, where r is the ratio of masked to total source points and $[\cdot]$ is rounding down. Capping k with $k_{\min} \in [0, C]$ guarantees the minimal editing capacity. This adaptive allocation balances identity preservation with editing flexibility, enabling larger masks to produce greater geometric changes.

3.3 Implementation Details

Our base model is an in-house VecSet-based 3D generator, which is trained on the base dataset $\mathcal{D}$ and their multi-view renderings, in a similar manner to prior works [28, 30, 65, 84]. This results in initial weights for both the VAE and the DiT. We also apply the augmentation strategies described in Sec. 3.2 to the base dataset to obtain the training datasets for each application.

The *enhancement* DiT is trained by combining the augmentations used for the enhancement and editing tasks. Following the literature [19, 56, 78], we train the DiT by progressively increasing the number of tokens $K = 512, 1024, 2048$ for 49K, 112K, and 390K steps. To train the *reconstruction* DiT, we fine-tune the enhancement model on the VGGT outputs for 25K steps. To train the *Shape editing* DiT we used the masked training set and $K=2048$ high-quality tokens. Training converges in just 25K steps. For every model, we used $C = 512$ *low-quality* conditioning tokens.

For additional robustness, we further augment the conditioning 3D shape by adding Gaussian blobs imitating floaters and by perturbing point normals, and we augment the image conditioning via random crops, color jitter, random backgrounds, and blur. We adopt the v -prediction [46] with a scaled linear schedule and $p = 0.1$ image-condition dropout. For inference, we use DDPM sampling with 100 steps and a CFG scale of 5. Please see the Appendix for more details.

4 Results

We evaluate the effectiveness of MeshReGen on the applications of Sec. 3.2.

4.1 Compositional 3D Enhancement

We first evaluate MeshReGen on the task of compositional 3D enhancement of whole scenes, as a component of the WorldGen [57] system. Starting from an image I (concept art), the entire scene is reconstructed as a single 3D object by an image-to-3D model [31], is then decomposed [7] into individual objects $\mathbf{y}^L$, and those are upgraded to high-quality versions $\mathbf{x}^H$ using MeshReGen, using corresponding guidance images I' generated by a VLM [57].

Qualitative results are given in Fig. 3. Quantitatively, we evaluate the performance of the model on 623 decomposed low-quality objects extracted from 21 scene meshes. We use ULIP-3D [71] to measure 3D shape-view alignment and Multi-View (MV) ImageReward [69] to measure perceptual quality. Specifically, we generate concise text captions of conditional images with LLaMA 3.2 [15] and

Metrics	Input [31]	+DetailGen3D [11]	+Ours
ULIP-3D [71]	0.2280	0.2294	0.2626
MV-ImageReward [69]	0.1716	0.1003	0.3394

Table 1: 3D shape enhancement evaluation. MV-ImageReward [70] reports the average ImageReward over four renders per object.

Table 2: 3D object reconstruction on GSO [13]. “# View” denotes the number of input images per baseline. VGGT results are based on meshes converted by Poisson reconstruction [22].

	Model	# View	CD ($\downarrow$)	F-score ($\uparrow$)	IoU ($\uparrow$)	PSNR ($\uparrow$)	SSIM ($\uparrow$)	LPIPS ($\downarrow$)
View > 1	InstantMesh [68]	1	0.0188	0.2840	0.6025	20.3317	0.9188	0.1389
	TRELLIS [66]	1	0.0122	0.4216	0.6442	22.4605	0.9342	0.1059
	AssetGen [50]	4	0.0111	0.4663	0.6703	24.3809	0.9206	0.1024
	TRELLIS [66]	4	0.0123	0.4318	0.6552	22.7584	0.9351	0.1036
	MeshLRM [63]	4	0.0115	0.4237	0.7035	22.8111	0.9290	0.1180
Hybrid	VGGT [59]	1	0.1286	0.0087	0.0247	9.0717	0.7979	0.5356
	+ MeshReGen	1	0.0135	0.3658	0.6438	21.8982	0.9315	0.1097
	VGGT [59]	2	0.1340	0.0083	0.0278	8.8537	0.7867	0.5695
	+ MeshReGen	2	0.0112	0.4299	0.7053	22.9795	0.9353	0.0985
	VGGT [59]	3	0.1435	0.0089	0.0274	8.3342	0.7615	0.6077
	+ MeshReGen	3	0.0090	0.4650	0.7372	23.6394	0.9385	0.0918
	VGGT [59]	4	0.1276	0.0119	0.0193	8.7823	0.7704	0.5952
	+ MeshReGen	4	0.0081	0.4913	0.7574	<u>24.2754</u>	0.9408	0.0873

use these prompts to compute the averaged ImageReward across four rendered views per object. As shown in Tab. 1 and Fig. 4, MeshReGen substantially improves geometric fidelity and visual quality of initial decomposed objects [31], surpassing state-of-the-art baselines [12] for geometry refinement.

4.2 Guided Image-to-3D

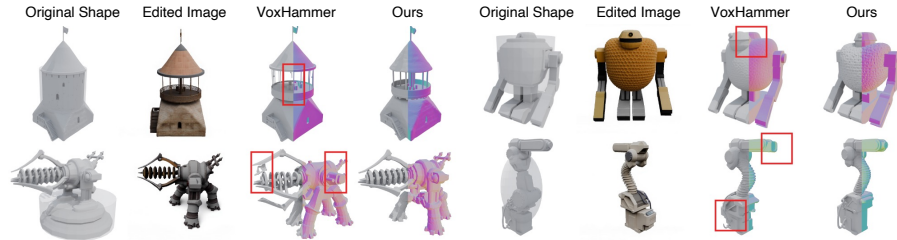
We consider 3D generation guided by the output of a reconstruction network, VGGT. Following [49, 62], we benchmark the reconstruction performance on the GSO dataset [13]. For perceptual evaluation, we compute PSNR, SSIM [60], and LPIPS [83] between the rendered and ground-truth normal maps by rescaling unit normals to $[0, 255]$. For geometric accuracy, we report Chamfer Distance (CD), F-score@1%, and IoU.

We compare against both single-view baselines (InstantMesh [67], TRELLIS [65]) and multi-view methods (3D AssetGen [49], TRELLIS (multi-image variant) [65], and MeshLRM [62]). TRELLIS (multi-image) randomly selects one of the conditioning images at each sampling step. We also evaluate the unrefined VGGT reconstructions $\mathbf{x}^L$ as a separate series of baselines, for which we convert them to meshes using Poisson reconstruction [22].

As shown in Tab. 2 and Fig. 5, MeshReGen significantly improves the geometric accuracy over the baselines. Single-view methods suffer from inaccurate hallucinations due to limited visibility. Moreover, MeshReGen matches or even outperforms specialized multi-view methods by using the same number or even

Table 3: Ablation study of the conditioning mechanism. Designs of ID = 1 and ID = 2 are inspired by CLAY [80] and Hunyuan3D-Omni [20], respectively.

Id	Method	Cond. Space	CD ($\downarrow$)	F-score ($\uparrow$)	IoU ($\uparrow$)	PSNR ($\uparrow$)	SSIM ($\uparrow$)	LPIPS ($\downarrow$)
1	Additional CA	CA	0.0066	0.6107	0.6889	24.4104	0.9258	0.0900
2	Extended CA	CA	0.0271	0.2374	0.2818	18.7145	0.9051	0.1410
3	Add z^L	Input	0.0112	0.5259	0.5960	22.7384	0.9175	0.1060
4	Ours	Input	0.0044	0.7538	0.7735	26.3958	0.9338	0.0765

**Fig. 7:** 3D object editing examples. Original shapes and edited images (shown in inset) are from [27]. The resulting shape maintains both the orientation and the level of detail of the input shape. The 3D edit M masking the edit region is also visualized.**Fig. 8:** Qualitative comparisons of object editing with [27]. Red boxes highlight artifacts in the editing results or differences between the edited and original shapes. The 3D edit mask M is also visualized.

fewer input views (i.e., two to three views). Fig. 6 demonstrates robust reconstruction from sparse views. Given a single image, it reconstructs the visible regions in detail, and progressively refines the occluded areas as more views are provided, combining feed-forward prediction and generative restoration.

4.3 3D Shape Editing

To assess 3D shape editing, we utilize 3D source objects, editing masks, and image prompts from the Edit3D-Bench dataset [26]. The image prompts were

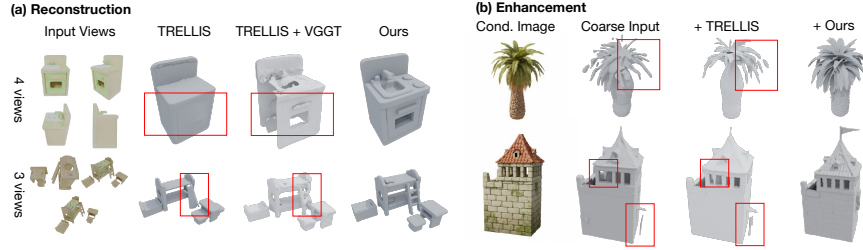


Fig. 9: (a) Reconstruction from sparse views. We treat VGGT predictions as sparse structures in TRELLIS’s first stage (TRELLIS + VGGT). (b) Enhancement from degraded geometry. TRELLIS struggles to correct artifacts. Both TRELLIS and ours use the exact same input shape or VGGT predictions.

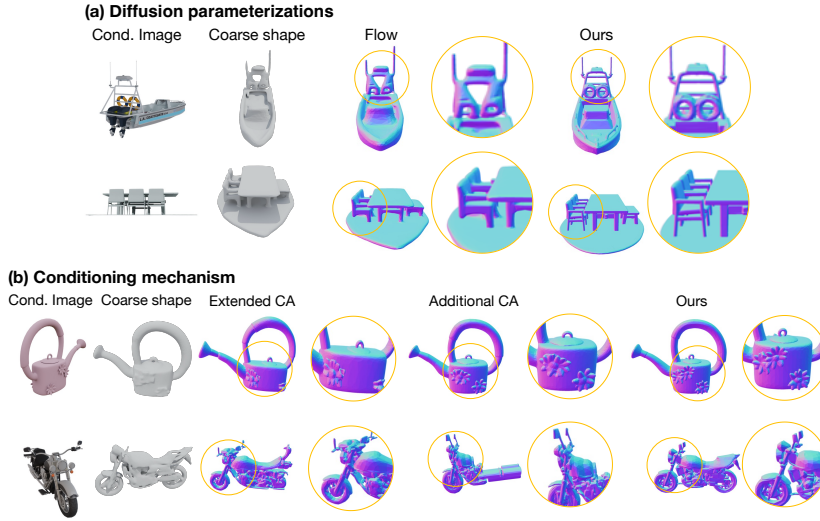


Fig. 10: Qualitative examples from the ablation study. (a) Samples from the ablation study of different diffusion parameterizations in Tab. 4. (b) Samples from the ablation study of the conditioning mechanism in Tab. 3. Designs of *Extended CA* and *Additional CA* are inspired by [20] and [80], respectively. The proposed method consistently outperforms other design choices with fine-grained details.

obtained by [26] by rendering the masked source view and inpainting them using FLUX.1 Fill [24] conditioned on the intended target text prompt.

As shown in Fig. 7, our framework enables effective 3D shape editing without architectural modifications. Fig. 8 shows that MeshReGen can capture better fine-grained and large-scale structural changes compared to [26], while preserving the original source identity. Unlike prior methods requiring expensive inversion [26] or post-optimization [37, 48, 86], our approach is computationally efficient and scalable. Moreover, by operating directly in the 3D VecSet space,

Table 4: Ablation study of different diffusion parameterizations. The design of “Flow” is inspired by [12].

Method	CD (↓)	F-score (↑)	IoU (↑)	PSNR (↑)	SSIM (↑)	LPIPS (↓)
Flow	0.0061	0.6026	0.6594	24.6295	0.9285	0.0893
Ours	0.0051	0.7003	0.6961	25.4428	0.9298	0.0787

it inherently preserves multi-view coherence without the need for 2D-to-3D lifting [1, 3].

4.4 3D shape generation from blockouts

As discussed in Sec. 1 in a typical 3D creation workflow, artists start from a 2D concept image and a coarse 3D block-out that defines the overall extent, proportions, and rough geometry. We show that 3D regeneration with MeshReGen naturally supports this setting: given simple block-out shapes and a guide image (Fig. 11), MeshReGen synthesizes fine geometric details consistent with the image while preserving the block-out’s structure and proportions. We compare against CLAY [80], a point-cloud-conditioned 3D generator, using its official interface and point-cloud sampling tool to condition on the blockouts.⁴ CLAY struggles to enrich the blockouts with fine-grained detail, highlighting the effectiveness of our 3D regeneration formulation and curated paired training data even for extremely coarse shape guidance, without needing a dedicated architecture module. As MeshReGen is not explicitly trained on block-out datasets, this demonstrates that the proposed framework is potentially generalizable with a wider range of degraded coarse geometry conditions. Please refer to Appendix D for more discussions.

4.5 Ablation Studies

Conditioning Mechanism. A key design choice (Sec. 3) is the mechanism for conditioning the generator on the low-information shape $\mathbf{y}^L$. We compare MeshReGen’s approach to alternative strategies: (1) introducing a separate cross-attention block dedicated to the 3D condition, parallel to the image-conditioned attention block; (2) extending the existing image-conditioned attention by concatenating coarse point cloud embeddings with DINO image features; and (3) directly adding $\mathbf{z}^L$ to the noisy latents. Designs (1) and (2) are inspired by CLAY [80] and Hunyuan3D-Omni [19], respectively. In Tab. 3 and Fig. 10, our design outperforms baselines, potentially due to its flexible learning capacity.

Diffusion Parameterization. We further compare our diffusion formulation with a rectified flow baseline [35, 38], where the model directly maps coarse latents $\mathbf{z}^L$ to the ground-truth latent distribution without explicit Gaussian noise. This alternative choice is inspired by DetailGen3D [12]. Tab. 4 shows that our diffusion parameterization achieves better reconstruction on the validation set.

⁴ Rodin.ai (Hyper3D): <https://hyper3d.ai/>

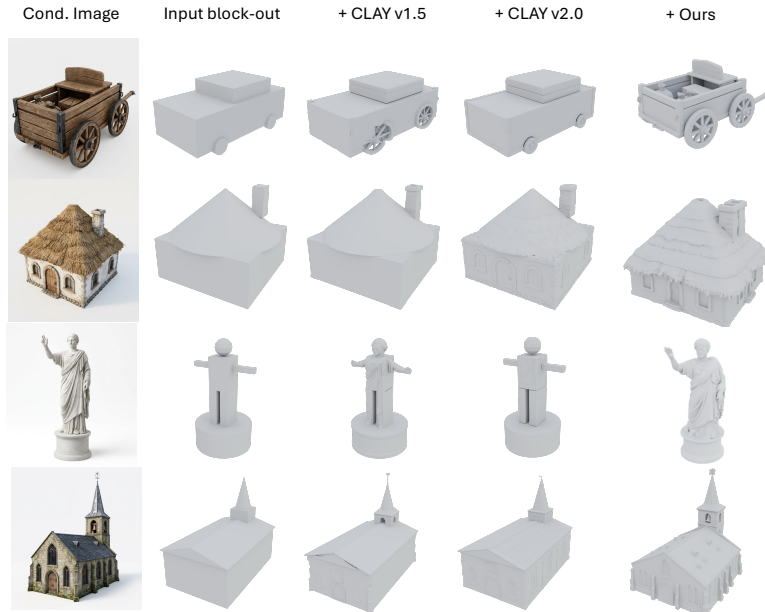


Fig. 11: Qualitative results of MeshReGen on 3D shape regeneration from blockouts. Compared to CLAY [80], MeshReGen is capable of adding fine details while preserving the underlying shapes and proportions of the blockouts.

*Applicability of existing generation frameworks to **re**generation.* We further test whether existing (controllable) 3D *generation* frameworks are readily generalizable to *regeneration* tasks. First, we apply TRELLIS to enhancement and reconstruction by respectively encoding degraded geometry or VGGT predictions into sparse structures in its first stage, followed by SLAT generation in the second stage. As shown in Fig. 9, TRELLIS fails to recover fine geometry, as it is trained on *clean geometry* which does not generalize to broken, incomplete inputs. Similarly, although CLAY [80] is not publicly available, we evaluate Hunyuan3D-Omni [19], a closely related *concurrent* framework, on reconstruction tasks and observe similar performance degradation (see Appendix). These results suggest that existing *generation* frameworks do not generalize to regeneration tasks with degraded/incomplete geometry without fine-tuning, motivating dedicated regeneration formulations and data.

Overall, using official baselines (when available) and controlled reimplementations under a fixed backbone and dataset, we show that our improvements stem from both dataset design *and* a generalizable architecture.

5 Conclusion

We have introduced MeshReGen, a versatile 3D regenerator that unifies diverse 3D-to-3D tasks under a single framework. Future work includes extending the framework to dynamics [75] and incorporating richer control signals, such as skeletons for rigged assets. Regarding the limitations, while MeshReGen demonstrates strong controllability, exploring alternative 3D latent representations (e.g., SLAT on active voxels [66]) may further improve generalizability.

References

1. Barda, A., Gadelha, M., Kim, V.G., Aigerman, N., Bermano, A.H., Groueix, T.: Instant3dit: Multiview inpainting for fast editing of 3d objects. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 16273–16282 (2025) [14](#) [11](#)
2. Boss, M., Huang, Z., Vasishtha, A., Jampani, V.: SF3D: stable fast 3D mesh reconstruction with UV-unwrapping and illumination disentanglement. arXiv **2408.00653** (2024) [10](#)
3. Chen, H., Shi, R., Liu, Y., Shen, B., Gu, J., Wetzstein, G., Su, H., Guibas, L.: Generic 3d diffusion adapter using controlled multi-view editing. arXiv preprint arXiv:2403.12032 (2024) [14](#) [11](#)
4. Chen, M., Laina, I., Vedaldi, A.: DGE: Direct gaussian 3D editing by consistent multi-view editing. In: Proceedings of the European Conference on Computer Vision (ECCV) (2024) [11](#)
5. Chen, M., Shapovalov, R., Laina, I., Monnier, T., Wang, J., Novotny, D., Vedaldi, A.: PartGen: Part-level 3D generation and reconstruction with multi-view diffusion models. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2025) [8](#)
6. Chen, M., Wang, J., Shapovalov, R., Monnier, T., Jung, H., Wang, D., Ranjan, R., Laina, I., Vedaldi, A.: Autopartgen: Autogressive 3d part generation and discovery. arXiv preprint arXiv:2507.13346 (2025) [5](#)
7. Chen, M., Wang, J., Shapovalov, R., Monnier, T., Jung, H., Wang, D., Ranjan, R., Laina, I., Vedaldi, A.: AutoPartGen: Autogressive 3D part generation and discovery. In: Proceedings of Advances in Neural Information Processing Systems (NeurIPS) (2025) [6](#) [8](#) [10](#)
8. Chen, R., Zhang, J., Liang, Y., Luo, G., Li, W., Liu, J., Li, X., Long, X., Feng, J., Tan, P.: Dora: Sampling and benchmarking for 3D shape variational auto-encoders. arXiv **2412.17808** (2024) [4](#)
9. Deitke, M., Liu, R., Wallingford, M., Ngo, H., Michel, O., Kusupati, A., Fan, A., Laforte, C., Voleti, V., Gadre, S.Y., Vanderbilt, E., Kembhavi, A., Vondrick, C., Gkioxari, G., Ehsani, K., Schmidt, L., Farhadi, A.: Objaverse-XL: A universe of 10M+ 3D objects. CoRR **abs/2307.05663** (2023) [3](#) [4](#)
10. Deitke, M., Schwenk, D., Salvador, J., Weihs, L., Michel, O., Vanderbilt, E., Schmidt, L., Ehsani, K., Kembhavi, A., Farhadi, A.: Objaverse: A universe of annotated 3D objects. In: Proc. CVPR (2023) [4](#) [8](#)
11. Deng, K., Guo, Y.C., Sun, J., Zou, Z.X., Li, Y., Cai, X., Cao, Y.P., Liu, Y., Liang, D.: Detailgen3d: Generative 3d geometry enhancement via data-dependent flow. arXiv preprint arXiv:2411.16820 (2024) [10](#) [3](#)
12. Deng, K., Guo, Y., Sun, J., Zou, Z., Li, Y., Cai, X., Cao, Y., Liu, Y., Liang, D.: DetailGen3D: generative 3D geometry enhancement via data-dependent flow. arXiv **2411.16820** (2024) [3](#) [4](#) [6](#) [11](#) [14](#)
13. Downs, L., Francis, A., Koenig, N., Kinman, B., Hickman, R., Reymann, K., McHugh, T.B., Vanhoucke, V.: Google Scanned Objects: A high-quality dataset of 3D scanned household items. In: Proc. ICRA (2022) [7](#) [11](#) [9](#)
14. Gao, W., Wang, D., Fan, Y., Bozic, A., Stuyck, T., Li, Z., Dong, Z., Ranjan, R., Sarafianos, N.: 3D mesh editing using masked LRMs. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7154–7165 (2025) [9](#)

15. Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al.: The llama 3 herd of models. arXiv preprint arXiv:2407.21783 (2024) [10](#)
16. Herrmann, L.R.: Laplacian-isoparametric grid generation scheme. *Journal of the Engineering Mechanics Division* **102**(5), 749–756 (1976) [8](#) [3](#)
17. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems* **33**, 6840–6851 (2020) [2](#)
18. Hong, Y., Zhang, K., Gu, J., Bi, S., Zhou, Y., Liu, D., Liu, F., Sunkavalli, K., Bui, T., Tan, H.: LRM: Large reconstruction model for single image to 3D. In: *Proc. ICLR* (2024) [11](#)
19. Hunyuan3D, T., Yang, S., Yang, M., Feng, Y., Huang, X., Zhang, S., He, Z., Luo, D., Liu, H., Zhao, Y., Lin, Q., Lai, Z., Yang, X., Shi, H., Zhao, Z., Zhang, B., Yan, H., Wang, L., Liu, S., Zhang, J., Chen, M., Dong, L., Jia, Y., Cai, Y., Yu, J., Tang, Y., Guo, D., Yu, J., Zhang, H., Ye, Z., He, P., Wu, R., Wei, S., Zhang, C., Tan, Y., Sun, Y., Niu, L., Huang, S., Zheng, B., Liu, S., Chen, S., Yuan, X., Yang, X., Liu, K., Zhu, J., Chen, P., Liu, T., Wang, D., Liu, Y., Linus, Jiang, J., Huang, J., Guo, C.: Hunyuan3D 2.1: From images to high-fidelity 3D assets with production-ready PBR material. arXiv **2506.15442** (2025) [4](#) [10](#) [14](#) [15](#)
20. Hunyuan3D, T., Zhang, B., Guo, C., Liu, H., Yan, H., Shi, H., Huang, J., Yu, J., Li, K., Wang, P., et al.: Hunyuan3d-omni: A unified framework for controllable generation of 3d assets. arXiv preprint arXiv:2509.21245 (2025) [3](#) [12](#) [13](#) [4](#) [7](#) [9](#) [11](#)
21. Jin, H., Jiang, H., Tan, H., Zhang, K., Bi, S., Zhang, T., Luan, F., Snively, N., Xu, Z.: LVSM: a large view synthesis model with minimal 3D inductive bias. In: *Proc. ICLR* (2025) [11](#)
22. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: *Proceedings of the fourth Eurographics symposium on Geometry processing* (2006) [11](#)
23. Keetha, N., Müller, N., Schönberger, J., Porzi, L., Zhang, Y., Fischer, T., Knapitsch, A., Zauss, D., Weber, E., Antunes, N., Luiten, J., Lopez-Antequera, M., Bulò, S.R., Richardt, C., Ramanan, D., Scherer, S., Kotschieder, P.: MapAnything: universal feed-forward metric 3D reconstruction. In: *Proc. 3DV* (2026) [9](#)
24. Labs, B.F., Batifol, S., Blattmann, A., Boesel, F., Consul, S., Diagne, C., Dockhorn, T., English, J., English, Z., Esser, P., Kulal, S., Lacey, K., Levi, Y., Li, C., Lorenz, D., Müller, J., Podell, D., Rombach, R., Saini, H., Sauer, A., Smith, L.: FLUX.1 kontext: Flow matching for in-context image generation and editing in latent space. arXiv **2506.15742** (2025) [4](#) [9](#) [13](#)
25. Lei, B., Li, Y., Liu, X., Yang, S., Xu, L., Huang, J., Tang, R., Weng, H., Liu, J., Xu, J., Zhou, Z., Zhu, Y., Xing, J., Xu, J., Ma, C., Yan, X., Yang, Y., Wang, C., Xu, D., Ma, X., Chen, Y., Li, J., Yang, M., Zhang, S., Feng, Y., Huang, X., Luo, D., He, Z., Jiang, P., Hu, C., Qin, Z., Miao, S., Liu, H., Zhao, Y., Lai, Z., Lin, Q., Zhao, Z., Li, K., Yang, X., Shi, H., Yang, X., Wang, Y., Yao, Z., Lian, Y., Liu, S., Han, X., Qin, W., Ouyang, C., Liu, J., Yuan, T., Jiang, S., Duan, H., Niu, Y., Lin, W., Sun, Y., Huang, S., Niu, L., Gong, G., Xiao, G., Zheng, B., Yuan, X., Chen, Q., Xiao, J., Zheng, D., Yang, X., Liu, K., Zhu, J., Wang, L., Lu, Q., Liu, J., Dong, L., Jiang, F., Chen, R., Wang, L., Zhang, C., Lin, J., Zhang, H., Ye, Z., He, P., Wu, R., Wu, Y., Du, J., Chen, J., Mao, X., Guo, D., Tang, Y., Tsai, Y., Tan, Y., Yu, J., Yu, J., Zhang, K., Li, Y., Chen, P., Liu, T., Wang, D., Liu, Y., Linus, Jiang, J., Chen, Z., Guo, C.: Hunyuan3D studio: End-to-end AI pipeline for game-ready 3D asset generation. arXiv **2509.12815** (2025) [4](#)

26. Li, L., Huang, Z., Feng, H., Zhuang, G., Chen, R., Guo, C., Sheng, L.: VoxHammer: training-free precise and coherent 3d editing in native 3d space. arXiv **2508.19247** (2025) [3](#) [4](#) [12](#) [13](#)
27. Li, L., Huang, Z., Feng, H., Zhuang, G., Chen, R., Guo, C., Sheng, L.: Voxhammer: Training-free precise and coherent 3d editing in native 3d space. arXiv preprint arXiv:2508.19247 (2025) [9](#) [12](#) [11](#)
28. Li, W., Liu, J., Chen, R., Liang, Y., Chen, X., Tan, P., Long, X.: CraftsMan: high-fidelity mesh generation with 3d native generation and interactive geometry refiner. arXiv **2405.14979** (2024) [4](#) [5](#) [10](#)
29. Li, W., Liu, J., Yan, H., Chen, R., Liang, Y., Chen, X., Tan, P., Long, X.: Craftsman3d: High-fidelity mesh generation with 3d native generation and interactive geometry refiner. arXiv preprint arXiv:2405.14979 (2024) [11](#)
30. Li, Y., Zou, Z.X., Liu, Z., Wang, D., Liang, Y., Yu, Z., Liu, X., Guo, Y.C., Liang, D., Ouyang, W., Cao, Y.P.: TripoSG: high-fidelity 3D shape synthesis using large-scale rectified flow models. arXiv **2502.06608** (2025) [5](#) [10](#)
31. Li, Z., Wang, Y., Zheng, H., Luo, Y., Wen, B.: Sparc3D: Sparse representation and construction for high-resolution 3d shapes modeling. arXiv **2505.14521** (2025) [4](#) [5](#) [6](#) [8](#) [10](#) [11](#) [9](#)
32. Li, Z., Wang, Y., Zheng, H., Luo, Y., Wen, B.: Sparc3d: Sparse representation and construction for high-resolution 3d shapes modeling. arXiv preprint arXiv:2505.14521 (2025) [5](#)
33. Lin, H., Chen, S., Liew, J., Chen, D.Y., Li, Z., Shi, G., Feng, J., Kang, B.: Depth anything 3: Recovering the visual space from any views. arXiv **2511.10647** (2025) [9](#)
34. Lin, S., Liu, B., Li, J., Yang, X.: Common diffusion noise schedules and sample steps are flawed. arXiv.cs **abs/2305.08891** (2023) [1](#)
35. Lipman, Y., Chen, R.T.Q., Ben-Hamu, H., Nickel, M., Le, M.: Flow matching for generative modeling. arXiv.cs **abs/2210.02747** (2022) [14](#)
36. Liu, A., Lin, C., Liu, Y., Long, X., Dou, Z., Guo, H.X., Luo, P., Wang, W.: Part123: Part-aware 3D reconstruction from a single-view image. In: Proc. SIGGRAPH (2024) [8](#)
37. Liu, F., Wang, H., Chen, W., Sun, H., Duan, Y.: Make-your-3d: Fast and consistent subject-driven 3d content generation. In: European Conference on Computer Vision. pp. 389–406. Springer (2024) [13](#) [11](#)
38. Liu, X., Gong, C., qiang liu: Flow straight and fast: Learning to generate and transfer data with rectified flow. In: Proc. ICLR (2023) [14](#)
39. Lorensen, W., Cline, H.: Marching cubes: A high resolution 3D surface construction algorithm. ACM Computer Graphics **21**(24) (1987) [4](#)
40. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: Proc. ICLR (2019) [1](#)
41. Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J.Y., Ermon, S.: SDEdit: guided image synthesis and editing with stochastic differential equations. In: Proc. ICLR (2022) [8](#) [3](#)
42. Mitra, N.J., Nguyen, A.: Estimating surface normals in noisy point cloud data. In: Proceedings of the nineteenth annual symposium on Computational geometry. pp. 322–328 (2003) [9](#)
43. Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: Dinov2: Learning robust visual features without supervision. arXiv preprint arXiv:2304.07193 (2023) [5](#)
44. Peebles, W., Xie, S.: Scalable diffusion models with transformers. In: Proc. ICCV (2023) [3](#) [5](#)

45. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10684–10695 (2022) [1](#)
46. Salimans, T., Ho, J.: Progressive distillation for fast sampling of diffusion models. In: Proc. ICLR (2022) [10](#)
47. Schroeder, W.J., Zarge, J.A., Lorensen, W.E.: Decimation of triangle meshes. In: Proceedings of the 19th annual conference on Computer graphics and interactive techniques. pp. 65–70 (1992) [8](#) [3](#)
48. Sella, E., Fiebelman, G., Hedman, P., Averbuch-Elor, H.: Vox-e: Text-guided voxel editing of 3d objects. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 430–440 (2023) [13](#) [11](#)
49. Siddiqui, Y., Kokkinos, F., Monnier, T., Kariya, M., Kleiman, Y., Garreau, E., Gafni, O., Neverova, N., Vedaldi, A., Shapovalov, R., Novotny, D.: Meta 3D Asset Gen: Text-to-mesh generation with high-quality geometry, texture, and PBR materials. In: Proceedings of Advances in Neural Information Processing Systems (NeurIPS) (2024) [4](#) [11](#)
50. Siddiqui, Y., Monnier, T., Kokkinos, F., Kariya, M., Kleiman, Y., Garreau, E., Gafni, O., Neverova, N., Vedaldi, A., Shapovalov, R., et al.: Meta 3d assetgen: Text-to-mesh generation with high-quality geometry, texture, and pbr materials. Advances in Neural Information Processing Systems **37**, 9532–9564 (2024) [11](#)
51. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. In: Proc. ICLR (2021) [2](#)
52. Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., Seidel, H.P.: Laplacian surface editing. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing. pp. 175–184 (2004) [8](#) [3](#)
53. Tang, J., Chen, Z., Chen, X., Wang, T., Zeng, G., Liu, Z.: LGM: Large multi-view Gaussian model for high-resolution 3D content creation. arXiv **2402.05054** (2024) [11](#)
54. Taubin, G.: Curve and surface smoothing without shrinkage. In: Proceedings of IEEE international conference on computer vision. pp. 852–857. IEEE (1995) [3](#)
55. Team, G., Anil, R., Borgeaud, S., Alayrac, J.B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A.M., Hauth, A., Millican, K., et al.: Gemini: a family of highly capable multimodal models. arXiv preprint arXiv:2312.11805 (2023) [4](#)
56. TripoAI: Tripo3D text-to-3D (2024), <https://www.tripo3d.ai> [2](#) [4](#) [10](#)
57. Wang, D., Jung, H., Monnier, T., Sohn, K., Zou, C., Xiang, X., Yeh, Y.Y., Liu, D., Huang, Z., Nguyen-Phuoc, T., Fan, Y., Oprea, S., Wang, Z., Shapovalov, R., Sarafianos, N., Groueix, T., Toisoul, A., Dhar, P., Chu, X., Chen, M., Park, G.Y., Ranjan, R., Vedaldi, A.: WorldGen: From text to traversable and interactive 3D worlds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2026) [8](#) [10](#)
58. Wang, J., Chen, M., Karaev, N., Vedaldi, A., Rupprecht, C., Novotny, D.: VGGT: Visual geometry grounded transformer. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2025) [7](#) [9](#)
59. Wang, J., Chen, M., Karaev, N., Vedaldi, A., Rupprecht, C., Novotny, D.: Vggt: Visual geometry grounded transformer. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 5294–5306 (2025) [11](#)
60. Wang, J., Thieson, B., Xu, Y., Cohen, M.F.: Image and video segmentation by anisotropic kernel mean shift. In: Proc. ECCV (2004) [11](#)

61. Wang, Y., Zhou, J., Zhu, H., Chang, W., Zhou, Y., Li, Z., Chen, J., Pang, J., Shen, C., He, T.: π^3 : Permutation-equivariant visual geometry learning. arXiv **2507.13347** (2025) [9](#)
62. Wei, X., Zhang, K., Bi, S., Tan, H., Luan, F., Deschaintre, V., Sunkavalli, K., Su, H., Xu, Z.: MeshLRM: large reconstruction model for high-quality mesh. arXiv **2404.12385** (2024) [11](#)
63. Wei, X., Zhang, K., Bi, S., Tan, H., Luan, F., Deschaintre, V., Sunkavalli, K., Su, H., Xu, Z.: Meshlrn: Large reconstruction model for high-quality meshes. arXiv preprint arXiv:2404.12385 (2024) [11](#)
64. Wu, C., Li, J., Zhou, J., Lin, J., Gao, K., Yan, K., Yin, S.m., Bai, S., Xu, X., Chen, Y., et al.: Qwen-image technical report. arXiv preprint arXiv:2508.02324 (2025) [4](#)
65. Xiang, J., Lv, Z., Xu, S., Deng, Y., Wang, R., Zhang, B., Chen, D., Tong, X., Yang, J.: Structured 3D latents for scalable and versatile 3d generation. In: Proc. CVPR (2025) [2](#) [4](#) [5](#) [6](#) [10](#) [11](#)
66. Xiang, J., Lv, Z., Xu, S., Deng, Y., Wang, R., Zhang, B., Chen, D., Tong, X., Yang, J.: Structured 3d latents for scalable and versatile 3d generation. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 21469–21480 (2025) [11](#) [15](#)
67. Xu, J., Cheng, W., Gao, Y., Wang, X., Gao, S., Shan, Y.: InstantMesh: efficient 3D mesh generation from a single image with sparse-view large reconstruction models. arXiv **2404.07191** (2024) [4](#) [11](#)
68. Xu, J., Cheng, W., Gao, Y., Wang, X., Gao, S., Shan, Y.: Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. arXiv preprint arXiv:2404.07191 (2024) [11](#)
69. Xu, J., Liu, X., Wu, Y., Tong, Y., Li, Q., Ding, M., Tang, J., Dong, Y.: Imagere-ward: Learning and evaluating human preferences for text-to-image generation. Advances in Neural Information Processing Systems **36**, 15903–15935 (2023) [10](#)
70. Xu, J., Liu, X., Wu, Y., Tong, Y., Li, Q., Ding, M., Tang, J., Dong, Y.: Imagere-ward: Learning and evaluating human preferences for text-to-image generation. Advances in Neural Information Processing Systems **36** (2024) [10](#)
71. Xue, L., Gao, M., Xing, C., Martín-Martín, R., Wu, J., Xiong, C., Xu, R., Niebles, J.C., Savarese, S.: Ulip: Learning a unified representation of language, images, and point clouds for 3d understanding. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 1179–1189 (2023) [10](#)
72. Yan, H., Luo, K., Li, W., Liang, Y., Li, S., Huang, J., Guo, C., Tan, P.: Posemaster: Generating 3d characters in arbitrary poses from a single image. arXiv preprint arXiv:2506.21076 (2025) [4](#) [9](#)
73. Yang, Y., Guo, Y.C., Huang, Y., Zou, Z.X., Yu, Z., Li, Y., Cao, Y.P., Liu, X.: HoloPart: generative 3d part amodal segmentation. arXiv **2504.07943** (2025) [8](#)
74. Ye, C., Qiu, L., Gu, X., Zuo, Q., Wu, Y., Dong, Z., Bo, L., Xiu, Y., Han, X.: Stablenormal: Reducing diffusion variance for stable and sharp normal. ACM Transactions on Graphics (TOG) **43**(6), 1–18 (2024) [11](#)
75. Yenphraphai, J., Mirzaei, A., Chen, J., Zou, J., Tulyakov, S., Yeh, R.A., Wonka, P., Wang, C.: Shapegen4d: Towards high quality 4d shape generation from videos. arXiv preprint (2025) [15](#)
76. Yin, S., Zhang, Z., Tang, Z., Gao, K., Xu, X., Yan, K., Li, J., Chen, Y., Chen, Y., Shum, H.Y., et al.: Qwen-image-layered: Towards inherent editability via layer decomposition. arXiv preprint arXiv:2512.15603 (2025) [4](#)
77. Zhang, B., Tang, J., Niessner, M., Wonka, P.: 3DShape2VecSet: A 3D shape representation for neural fields and generative diffusion models. In: ACM Transactions on Graphics (2023) [2](#) [3](#) [5](#)

78. Zhang, B., Tang, J., Niessner, M., Wonka, P.: 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Transactions On Graphics (TOG)* **42**(4), 1–16 (2023) [10](#)
79. Zhang, K., Bi, S., Tan, H., Xiangli, Y., Zhao, N., Sunkavalli, K., Xu, Z.: GS-LRM: large reconstruction model for 3D Gaussian splatting. *arXiv* **2404.19702** (2024) [11](#)
80. Zhang, L., Wang, Z., Zhang, Q., Qiu, Q., Pang, A., Jiang, H., Yang, W., Xu, L., Yu, J.: CLAY: A controllable large-scale generative model for creating high-quality 3D assets. In: *Proc. SIGGRAPH* (2024) [2](#), [3](#), [4](#), [12](#), [13](#), [14](#), [15](#), [7](#), [9](#), [10](#), [11](#)
81. Zhang, L., Rao, A., Agrawala, M.: Adding conditional control to text-to-image diffusion models. In: *Proc. CVPR* (2023) [7](#)
82. Zhang, Q., Jian, X., Zhang, X., Wang, W., Hou, J.: Supercarver: Texture-consistent 3d geometry super-resolution for high-fidelity surface detail generation. *arXiv* **2503.09439** (2025) [11](#)
83. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: *Proc. CVPR* (2018) [11](#)
84. Zhao, Z., Lai, Z., Lin, Q., Zhao, Y., Liu, H., Yang, S., Feng, Y., Yang, M., Zhang, S., Yang, X., Shi, H., Liu, S., Wu, J., Lian, Y., Yang, F., Tang, R., He, Z., Wang, X., Liu, J., Zuo, X., Chen, Z., Lei, B., Weng, H., Xu, J., Zhu, Y., Liu, X., Xu, L., Hu, C., Huang, T., Wang, L., Zhang, J., Chen, M., Dong, L., Jia, Y., Cai, Y., Yu, J., Tang, Y., Zhang, H., Ye, Z., He, P., Wu, R., Zhang, C., Tan, Y., Xiao, J., Tao, Y., Zhu, J., Xue, J., Liu, K., Zhao, C., Wu, X., Hu, Z., Qin, L., Peng, J., Li, Z., Chen, M., Zhang, X., Niu, L., Wang, P., Wang, Y., Kuang, H., Fan, Z., Zheng, X., Zhuang, W., He, Y., Liu, T., Yang, Y., Wang, D., Liu, Y., Jiang, J., Huang, J., Guo, C.: Hunyuan3D 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv* **2501.12202** (2025) [4](#), [5](#), [10](#)
85. Zhao, Z., Liu, W., Chen, X., Zeng, X., Wang, R., Cheng, P., Fu, B., Chen, T., Yu, G., Gao, S.: Michelangelo: Conditional 3D shape generation based on shape-image-text aligned latent representation. In: *Proc. NeurIPS* (2023) [4](#)
86. Zhuang, J., Kang, D., Cao, Y.P., Li, G., Lin, L., Shan, Y.: Tip-editor: An accurate 3d editor following both text-prompts and image-prompts. *ACM Transactions on Graphics (TOG)* **43**(4), 1–12 (2024) [13](#), [11](#)