Instant3dit: Multiview Inpainting for Fast Editing of 3D Objects

Amir Barda
Tel Aviv University, Israel
amirbarda@mail.tau.ac.il

Matheus Gadelha
Adobe Research, USA
gadelha@adobe.com

Vladimir G. Kim
Adobe Research, USA
vokim@adobe.com

Noam Aigerman
Université de Montréal, Canada
noam.aigerman@umontreal.ca

Amit H. Bermano
Tel Aviv University, Israel
amberman@mail.tau.ac.il

Thibault Groueix
Adobe Research, USA
groueix@adobe.com

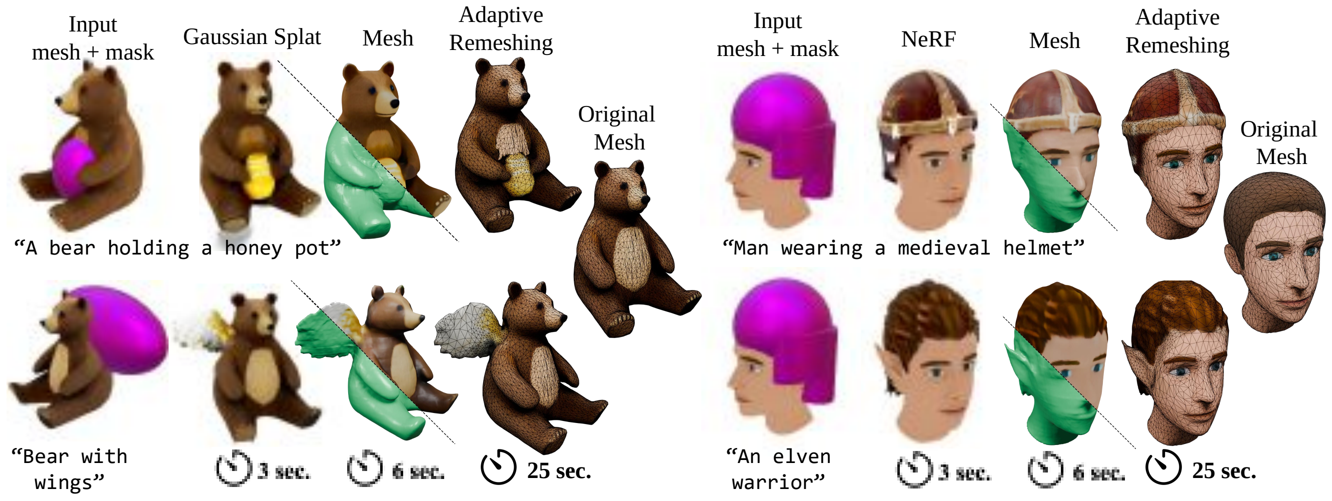


Figure 1. Our method takes as input a 3D object along with a 3D mask (first column) and a text prompt, and uses our multiview inpainting diffusion model to consistently paint the mask in four rendered views of the object. Off-the-shelf reconstructors can be used on the multiview output to give an NeRF, a Gaussian Splat (second column), or a mesh (third column) that can be used along with adaptive remeshing to ensure the unmasked region is exactly preserved *e.g*. topology, uvs, (fourth and fifth column). This feedforward approach is orders of magnitude faster than previous works in generative 3D editing, **taking just** ≈ 3 **seconds per multiview edit**, then 0.7 seconds to reconstruct a GS or a NeRF, 3 seconds for a mesh, and ≈ 20 seconds for optional mesh post-processing.

Abstract

We propose a generative technique to edit 3D shapes, represented as meshes, NeRFs, or Gaussian Splats, in ≈ 3 seconds, without the need for running an SDS type of optimization. Our key insight is to cast 3D editing as a multiview image inpainting problem, as this representation is generic and can be mapped back to any 3D representation using the bank of available Large Reconstruction Models. We explore different fine-tuning strategies to obtain both multiview generation and inpainting capabilities within the same diffusion model. In particular, the design of the inpainting mask is an important factor of training an inpainting model, and we propose several masking strategies to mimic the types of edits a user would perform on a 3D shape. Our approach takes 3D generative editing from hours to seconds and produces higher-quality results compared to previous works. project page: https://amirbarda.github.io/Instant3dit.github.io/

1. Introduction

Recent years have seen an explosion in AI-guided 3D generative techniques for creation and editing of 3D content, allowing the user to control the generation process with often as little as a single text prompt [4, 9, 10, 40, 48, 53]. While text prompts provide an easy interface for novice users, they lack *fine control* over the generation, such as generating a specific object in a specific location over a pre-existing 3D model, e.g., make a bear hold a previously non-existent honey pot (see Figure 1).

Thus, several recent works have focused on such *localized* generation of 3D objects, namely 3D *inpainting* [6, 40, 48], i.e., filling-in masked-out content in a 3D object, conditioned on a textual description of the desired fill-in, similarly to 2D inpainting [8, 48]. Figure 1 shows several examples of such inpaintings produced by our method, where the mask (made up of coarse 3D primitives) is highlighted in purple color, the text prompt describing the target inpainting is written below, and the resulting inpainted 3D reconstructions are shown next to it.

Current 3D inpainting methods cannot be integrated into production pipelines, as they suffer from two fundamental issues: 1) long runtimes, and 2) low quality. We observe these two issues can be explained by previous works' reliance on optimization of the 3D model by *distilling* knowledge from a generative model for 2D images via some variant of Score Distillation Sampling (SDS) [35]. Such an optimization process is extremely slow, as it relies on running an image diffusion model over multiple renderings of the 3D object and back-propagating gradients, thus explaining the first issue of slow runtimes. However, we argue that it also lies at the root of the second issue, of degradation in quality: as we show through extensive comparisons (see Figure 6), this class of approaches significantly harms the *quality* of the inpainting, as SDS-like optimization produces inaccurate and fuzzy results. We attribute it to the fact observed in the original DreamFision paper [35], which stated that *"2D image samples produced using SDS tend to lack diversity"* and *"3D results exhibit few differences across random seeds,"* explaining this behavior with the tendency to seek specific distribution modes regardless of the seed. Thus, it stands to reason it will struggle to inpaint a region of an existing object that is not one of those seeked modes.

In this paper, we propose an alternative approach to 3D inpainting that significantly improves the quality of the inpainted results and the runtime of the process. Drawing inspiration from previous works for fast 3D generation [24, 29, 50, 56], we eliminate both the above issues, by turning the problem on its head: instead of directly optimizing a 3D object using, e.g., SDS, we instead train an image generator to create 2D images of the inpainted 3D object from canonical viewpoints, and then reconstruct the newly-inpainted 3D object in a post-process, via either a feed-forward prediction [24, 52, 56], or lightweight optimization [29, 50]. By doing so we show we avoid both the slow runtimes as well as issues with masking that arise from approaches such as SDS when applied to 3D inpainting.

Adapting this multiview generation technique to the inpainting problem leads to a core challenge: how to ensure that the inpainting of masked region in the different 2D generated image is 3D-consistent from different viewpoints (otherwise, it does not represent a coherent 3D object that can be reconstructed), while at the same time adhering to the 3D mask painted over the original 3D object. Thus, following our observation above, our main technical contribution is designing a scheme to obtain a diffusion model that provides multiview-consistent inpainting.

In order to achieve a multiview-consistent inpainting diffuser, we devise a custom training strategy, along with a novel dataset of 3D masks for 3D inpainting. Namely, we produce multiview consistent masks, avoiding problems resulting from occlusion (that our experiments validate is critical for performance). We design the dataset to support three designated editing modes with different levels of granularity (see Section 3.2)

Finally, we craft a custom training strategy to leverage the priors learned by pretrained text-conditioned image generators. We propose to start from a pretrained image inpainter, and use our dataset and training regime to make it multiview consistent. We provide extensive empirical evidence to the specific design choices of our approach, e.g., compare our strategy with the opposite route – fine-tune a multiview diffuser to perform inpainting.

We show through extensive experiments that our approach enables performing elaborate 3D editing using simple masking of a 3D region, rendering the object and the mask from multiple views, running the inpainting model, and using fast multiview reconstruction techniques to obtain the edited 3D object (see Figure 1). Our approach is thus agnostic to the underlying 3D representation, and we show it supports meshes [52], Gaussian Splats [56] and Radiance Fields [52]. Our experiments validate the speed of our multiview image-based representation and the higher quality of the resulting inpainting compared to previous techniques. To summarize, our contributions are:

- The first high-quality, fast method for 3D inpainting, enabling fast and localized generation on NeRFs, Gaussian Splats and meshes.
- A 3D masking approach along with a novel training dataset of masks for multiview inpainting, with three types of masks corresponding to different edit modes.
- A finetuning strategy to efficiently leverage pretrained representations, along with an empirical study of alternative training strategies for multiview inpainting.

2. Related Work

We position our work against other generative 3D editing approaches and separately discuss multiview 3D generation and inpainting. Closest to our approach are NeRFiller [48] and the concurrent work MVInpainter [6].

3D Generative Editing. As we aim for a general tool, we focus the discussion on category-agnostic approaches. The literature can be organized by the core technique used to distill the 2D generative prior - CLIP optimization [15, 21, 32], Score-Distillation Sampling (SDS) [4, 8–10, 22, 25, 31, 33, 40, 58], Iterative Dataset Update (IDU) [16, 48] - or by the type of representation targeted - meshes [4, 15, 21, 22, 32], NeRFs [8, 16, 31, 33, 40, 48], or SDF [9, 10, 25, 58]. All generative editing approaches take advantage of Text-to-Image (T2I) models, whose prowess stems from the billions of text-image pairs available as training data. This abundance of data does not extend to the 3d domain, with the largest public datasets, Objaverse and Objaverse-XL [11, 12], containing millions of meshes, with the vast ma-

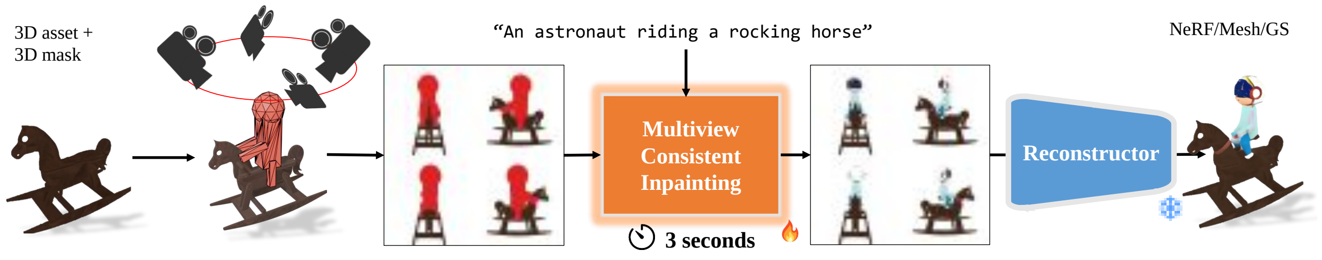

3D asset +
3D mask
"An astronaut riding a rocking horse"
Multiview Consistent Inpainting
3 seconds
Reconstructor
NeRF/Mesh/GS

Figure 2. **Overview.** Given a NeRF, a Gaussian Splat, or a mesh, the user draws a 3D mask to mark a region to be filled and provides a text prompt to guide the generation. Instant3dit renders four canonical views of the masked object and uses our multiview inpainting network to fill the mask. We use off-the-shelf 3D reconstructors to convert the multiview representation into a NeRF, a Gaussian Splat, or a mesh.

jority being very low-quality data.

The Score Distilation Sampling (SDS) loss, first defined in [35], use T2I models as guidance for 3d generation, by iteratively denoising renderings. SDS has multiple downsides : the denoised images lack consistency at different iterations, and the optimization is therefore very noisy, requiring careful parameter tuning to converge. Second, the optimization typically takes around one hour to converge, as it requires a lot of steps [4]. Combining SDS with Gaussian Splatting partially alleviate this problem by making the rendering operation in each step much faster, as shown in GaussianDreamer [53], but is still not interactive (about 15 minutes per generation). Lastly, SDS produces saturated, low-quality textures due to the high classifier-free guidance used, even though recent variations alleviate this issue [30].

Iterative Dataset Update, first introduced in [16], is a variant of SDS, taking multiple denoising steps to produce a clean image, used for several optimization steps in a row. NeRFiller [48] extends this approach for editing of NeRFs.

Compared to prior work, our approach does not require an optimization to leverage the generative priorhttps://git.corp.adobe.com/pages/adobe-research/mvgenfill-client/. We propose to perform the edit using a multiview image grid to represent the 3D asset, with a single diffusion inference. As a result, editing is orders of magnitude faster (*i.e*. from hours to seconds) and more stable (*i.e*. our approach always robustly produces a result and does not require careful tuning of hyperparameters). Lastly, several off-the-shelf LRM transformers [18] can instantly convert our inpainted image grid to meshes [49], NeRFs [24], or Gaussian Splats [56], making our editing technique representation agnostic.

Multiview generative 3D. To encourage multiview 3D consistency, several works have explored modifying the attention layers with *cross*-attention blocks to facilitate the exchange of information between views [27, 28, 42]. A simpler approach consists of directly generating a 2x2 image grid, without modifying the attention layers. In this case, the exchange of information between view is done via *self*-attention. This effectively trades resolution with consistency, and can be achieved by conditionning a T2I model on multiview depth [7, 44], or by fine-tuning on a dataset of 3D multiview renderings [24, 41, 45]. Unique3D [50] and direct 2.5d [29] extend this approach to image grids of normals, and CRM [47] to canonical coordinate map. These works have shown that adopting this representation to generate 3D shape is orders of magnitude faster than SDS, from hours to seconds. In this work, we extend these ideas to 3D editing, and explore fine-tuning strategies to get inpainting and multiview generation in the same model. Of note, NeRFiller [48] proposes to inpaint a grid of 2x2 images with a single-view inpainting model *without* fine-tuning it for multiview consistency. We compare against this baseline and show the importance of fine-tuning the inpainting model. Furthermore, NeRFiller relies on a more costly IDU optimization, which requires about 30k steps.

Image inpainting. The ability to collect and annotate images at scale has fueled major progress in image inpainting. Current inpainting tools have a wide range of applications ranging from real photographs to illustrative drawing [1, 39]. We refer the reader to the recent survey of Xiang *et al*. [51]. In this paper, we are interested in leveraging these pre-trained priors to edit 3D assets. As discussed in the previous paragraph, the main challenge is achieving multiview consistent results, as open-source models like Stable Diffusion [39] inpaint each frame independently without 3D consistency. In this work, we propose to teach multiview consistency to a single-view inpainting network. As noted in Zeng *et al*. [55], the design of the inpainting masks is a critical part of training an inpainting model. The masks should resemble the type of masks that users will draw at inference as closely as possible. However, how to apply that insight to 3D editing is not straightforward. We propose three masking strategies that address different workflows in Section 3.2.

Concurrent work. Similarly observing that IDU and SDS are unstable and lengthy optimizations, the recent approach MVInpainter [6] adds a video priors to a single-view inpainting model, via a LoRA [19], to encourage multiview consistency. Their focus in on object insertion and removal in captured 3D scenes, while ours in on 3D generative editing of objects, which leads us to different masking strategies to train the inpainting model.

3. Method

The input to our system consists of tuples $\langle S, M, y\rangle$ representing a 3D shape S, a 3D mask region M, and a text description y – henceforth referred to as *prompt*. The goal of our method is to create a new shape S' whose areas covered by M are modified to follow y.

Multiview representation. We propose to perform 3D inpainting by inpainting multiple renderings of an object in a view-consistent manner. Consider a rendering operator $\mathcal{R}$ that renders a set of a shapes (a scene) $\mathcal{S}$, from a viewpoint π. We refer to $\mathcal{R}_c$ and $\mathcal{R}_b$ to indicate a rendering to a **c**olor RGB image and a **b**inary mask, respectively. Finally, we refer to $\mathcal{R}^U$ to indicate that we only render the visible pixels belonging to the shape U while assuming $U \in \mathcal{S}$.Using this operator, we define the multiview representations as :

$$I_k(U,V) := \bigoplus_{\pi\in\Pi} \mathcal{R}_k^U[\{U\}\cup\{V\};\pi] \tag{1}$$

where $\bigoplus$ concatenates the images in a 2×2 grid, k is rendering modality (**c**olor or **b**inary), and $\Pi = \{C(\alpha, \frac{\pi}{4}) | \alpha \in \{0, \frac{\pi}{2}, \pi, 3\frac{\pi}{2}\}\}$; $C(\alpha,\beta)$ is a function that returns a viewpoint configuration corresponding to a camera pointing at the origin of the coordinate system and positioned on the surface of a canonical sphere according to azimuth α and elevation β. Using this operator, the input to our multiview diffusion model is defined as $I_c(S,M)$ and $I_b(M,S)$. Intuitively, $I_c(S,M)$ is just an image containing the visible pixels of S in the scene $\{S\}\cup\{M\}$ rendered from multiple views organized in a grid, and $I_b(M,S)$ is binary rendering of the visible pixels of M. An illustration of this representation is presented in Figure 3.

3D editing. Given $I_c(S,M)$ and $I_b(M,S)$, we propose to use a diffusion model ϵ_θ to generate an inpainted multiview representation $\hat{I}_c$. From that, we can then use a posed multiview reconstruction process Φ to obtain the edited shape $\hat{S} = \Phi(\hat{I}_c)$. Different choices for Φ yield different applications and tradeoffs. For example, we can have very fast learning-based reconstruction from posed multiview images using various representations, like NeRFs [24], meshes [49] and gaussian splats [56]. We can also apply lightweight optimization routines based on differentiable rendering such as ROAR [3], ISOMER [50] or Direct2.5d [29]. This class of methods is usually slightly slower but has the additional benefit of allowing us to carefully craft the optimization procedure in order to achieve several desirable properties like geometric regularization and preservation of the original asset attributes – color, connectivity, UVs, and so on. Figure 1 shows results using various types of reconstructors. We use NeRF-LRM [24] to compare against other NeRF editing approaches in Table 1 and Figure 4. Generating an image with ϵ_θ takes about 3 seconds on an A100, and the various reconstructors range from a few milliseconds [24, 56] to a few seconds [29, 49, 50].

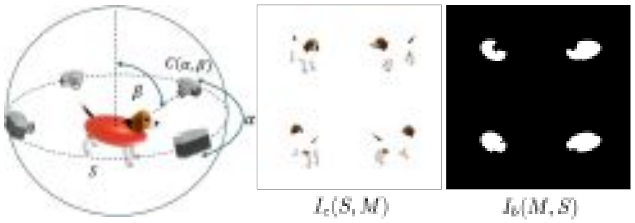


Figure 3. **Multiview representation.** We represent 3D shapes multiview renderings. Editing is done using an image-based diffusion model that operates on $I_c(S,M)$ and $I_b(M,S)$.

In the rest of this section, we explain how to train the diffusion model ϵ_θ for multiview generation (Section 3.1) and inpainting (Section 3.2).

3.1. Background on Diffusion Models.

Training. During training, we sample an image x from the dataset, with condition c (*e.g.* text, mask, or depth), a time step t between 0 and T, and a noise $\epsilon \sim \mathcal{N}(0, \mathrm{I})$, injected to x to create a noisy image $\tilde{x}(t)$:

$$\tilde{x}(t) = \sqrt{\alpha(t)}\cdot x + \sqrt{1-\alpha(t)}\cdot\epsilon, \tag{2}$$

where $\alpha(t)$ controls the amount of noise to inject *i.e.* $\alpha(0) = 1$ is no noise and $\alpha(T) = 0$ is pure noise. A denoising unet ϵ_θ is trained to denoise $\tilde{x}(t)$ by minimizing:

$$L_{\text{diff}} = w(t)||\epsilon_\theta(\tilde{x}(t); t, c) - x||^2, \tag{3}$$

where $w(t)$ a scheme to scale the gradients according to t. Once ϵ_θ is trained, $\epsilon_\theta(\tilde{x}(t); t, c)$ is the projection of $\tilde{x}(t)$ to the manifold of images defined by the training dataset. Details about the training procedure can be found in the supplemental material.

Inference. We start from pure noise $\tilde{x}(T)$ and follow the direction of the manifold defined by $\epsilon_\theta(\tilde{x}(t); t, c)$. There exists multiple samplers to discretize this trajectory into a discrete number of steps. In practice, we use the Euler scheduler [20] and 29 steps.

Latent Diffusion. In this work, we use latent diffusion models *i.e.* the diffusion happens in a 4-dimensional latent space instead of RGB space, and a pretrained VQ-VAE [38] encodes and decodes images from that space. Our method is agnostic to this, so we keep the presentation general.

Multiview diffusion. To generate 2x2 consistent views, we follow [24] and simply replace the training distribution with a distribution of 2x2 images. While this approach would yield poor result if it were trained from scratch, given the scarcity of high-quality 3D data, it performs well if the models are fine-tune from "foundation" text-to-image models, *i.e.* pre-trained on millions of images. To create the dataset, we render a curated list [24] of 5K objects from Objaverse [12], filtered for high-quality, and generate high-quality captions y_n for each 3D object with LLaVa [26]. We now explain how to train for inpainting jointly.

Inpainting. The main specificity is that the condition c is composed on the text prompt y, the base image with holes, and the inpainting mask *i.e.* $c = \{y, I_c(S, M), I_b(M, S)\}$. In practice, for latent models, $I_c(S, M)$ is passed through the encoder $\mathcal{E}$ of the VQ-VAE, concatenated with the downsampled version of the mask $I_b(M, S)$, and the noisy latents $\tilde{x}(t)$, leading to a 9-channel tensor input to the denoising unet ϵ_θ, along with the encoding of the text condition. During training, we randomly drop the mask 10% of the time, falling back to multiview diffusion training.

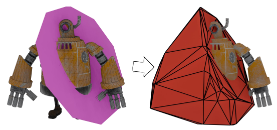
3.2. Multi-View Inpainting Masks

Dataset creation. A key part of our method consists in generating multiview masks that are 3D consistent. Thus, the binary masks used for training our multiview inpainting model are obtained by rendering 3D shapes; *i.e.* $I_b(M, S)$. Consider the example in Figure 3. Even though M is simply an ellipsoid, its multiview representation $I_b(M, S)$ is not – it has the occlusions from its interaction with S. We empirically demonstrate that 3D-aware masks and multiview consistent images are crucial for better performance (see Table 2). Our model is trained based on a set of shapes $\mathcal{D}$. For every shape $S \in \mathcal{D}$, we create a set of 3D masks $\mathcal{M}_S$. Using those, we can define our training dataset $\mathcal{I}$ as:

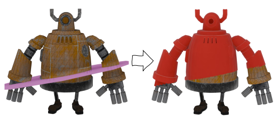
$$\mathcal{I} := \left\{ \underset{\text{where } S \in \mathcal{D}, M \in \mathcal{M}_S}{\langle I_c(S, \varnothing), I_c(S, M), I_b(M, S), y_S \rangle} \right\} \quad (4)$$

where $I_c(S, \varnothing), I_c(S, M), I_b(M, S)$ are the color ground-truth image, color input image, and binary input mask, respectively. y_S is a text prompt describing the shape S obtained from a VLM model [26]. Since generating the masks online would slow down training, we preprocess them offline. In practice, we have $|\mathcal{M}_S| = 30$, containing equal portions of 3 different kinds of masks. Considering that $|\mathcal{D}| \approx 5K$, our multiview dataset $\mathcal{I}$ has $\approx 150K$ data points. We will release our dataset upon publication.

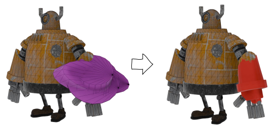
A key remaining issue is how $\mathcal{M}_S$ is created. As noted in [55], the design of the training masks plays a central role in training an inpainting model. The best performances are naturally obtained when the distribution of training masks closely follows the distribution of edits that users will make at test time. We confirm this in our experiments with a simple ablation, where we naively use random 2D mask (see Table 2). We thus propose three types of masks, corresponding to three types of editing. The remaining of this section will explain how each type of mask is created.



Type I: coarse edit. In this setting, the inpainted part of S is fully contained inside M. M is computed by randomly sampling a part of S and taking its convex hull. To select such part, we randomly sample a plane passing through S, effectively splitting S into two parts, and we select one part at random. More precisely, we start by sampling a random point p inside the bounding box of S and a random direction n. The plane P passing through p with normal n is defined by $\{x \in \mathbb{R}^3 | x \cdot p = p \cdot n\}$. M is defined as the convex hull of all the face midpoints that are above P *i.e.* $\{f = (v_1, v_2, v_3) | f \in F, \frac{v_1+v_2+v_3}{3} \cdot p >= p \cdot n\}$, where F denotes the list of faces. To avoid Z-fighting during rendering between S and M, we scale M by 20% while keeping its center of mass the same, ensuring it completely envelopes the part of S above P.



Type II: mesh sculpting. This type of mask is designed to represent a more precise edit where the user expects content to be created in a portion of space similar to the mask. It requires more expertise and time from the user than Type I, but also provides more precise control over the generated content. As we can see in the illustration to the right, M is a tight fit over S – no volume inside M is not also inside S. To generate these masks, we sample a plane P, as in Type I masks, and select all the faces that have their midpoint above P *i.e.* $M = \{f = (v_1, v_2, v_3) | f \in F, \frac{v_1+v_2+v_3}{3} \cdot p >= p \cdot n\}$.



Type III: surface editing. We aim to support local texture modifications, where the user selects a surface patch and prompts Instant3dit to modify its texture. There are various ways to generate this type of mask if we assume that the meshes have good triangulation. Unfortunately, this is typically not the case for shape datasets. Thus, propose a simple heuristic that works for any triangle soup. We sample a vertex p in S, then several cylinders with elliptical bases of varying sizes, all centered on p (depicted in purple). The number of cylinders is uniformly sampled between 3 and 6, the revolution axis is sampled on the unit sphere, the height and radii are sampled between 0.1 and 0.3. We call this volume C. Finally select all the faces whose midpoint falls within C *i.e.* $M = \{f = (v_1, v_2, v_3) | f \in F, \frac{v_1+v_2+v_3}{3} \in C\}$ (depicted in red).

4. Experiments

In this section, we describe our novel multiview inpainting benchmark, which we use to compare several baselines and run ablations.

Benchmark. We hold out 500 2x2 multiview images and their synthetic multiview masks from the training set to create our benchmark. We condition all inpainting on BLIP [23] captions,and ensure no training object is present in the benchmark. We then evaluate how well different methods inpaint all the views.

Evaluation Metrics. To evaluate the quality of multiview inpainting, we use three types of metrics: measuring prompt adherence, multiview consistency, and visual quality. To measure prompt adherence, we use CLIP [36] similarity score between the generated 2x2 multiview image and the text prompt. We follow previous work [34] and use two models, CLIP-ViT-L-14 (ClipL) and CLIP-ViT-BigG-14 (ClipG), for encoding. To measure 3D consistency we reconstruct NeRF from the sparse inpainted views [24], and re-render from same camera angles as the input sparse views. We then compare the NeRF renderings to generated inpainted images using various image similarity scores: SSIM [46], LPIPS [57] and DreamSim [13]. Finally, we measure visual quality using FID score [17] comparing to a distribution of held-out images.

Comparison to Baselines. We use these metrics to compare our method (Table 1, bottom) to several baseline alternatives (Table 1, top). As there are no off-the-shelf multiview consistent inpainting methods, we couple different image diffusion techniques with blended diffusion [2] to inpaint the multiview image. We try two different architectures from the commonly used SDXL diffusion model [34], SDXL (as-is) and SDXL-inpainting (fine-tuned on 2D inpainting task). Both SDXL baselines perform poorly on multiview consistency since they are not explicitly trained to complete images consistently and, in the case of SDXL-inpainting, are only trained with 2D image masks. Finally we test Instant3D [24] coupled with blended diffusion. Since this architecture is trained for 3D reconstruction it achieves a better multiview consistency, however, our method still outperforms this baseline with respect to all other metrics. See Figure 4 for some example results produced with baselines and our method.

Ablating Diffusion Backbones. Our method can be fine-tuned from any image diffusion model, and thus we further evaluate how performance changes with respect to different backbones (Table 1, bottom section). Unsurprisingly, weaker backbones with fewer parameters (SD 1.5, SD 2.0) [38] lead to inferior performance and very poor multiview consistency. Using Instant3D [24] gives the best multiview consistency, but fine-tuning does not seem to improve visual quality and prompt adherence. Our method fine-tunes from SDXL-inpainting [34], and generally learns to better adhere to the prompt with high visual fidelity. We believe this observation is consistent with insights from prior works [24, 48] that show that existing diffusion models already have some implicit understanding of multiview images and can be fine-tuned for 3D consistency with little training data. Inpainting, on the other hand, is a more challenging task, and thus, it is easier to fine-tune a model that was trained at scale for an inpainting task (SDXL-inpainting) to create a multiview consistent image. See supplemental material for qualitative examples.

Method	Prompt Adherence		Multi-view consistency			Visual quality
	ClipL↑	ClipG↑	SSIM↑	LPIPS↓	DreamSim↓	FID↓
SDXL [34]	28.63	42.58	0.874	0.065	0.134	127.0
SDXL-inpaiting [34]	27.57	41.33	0.857	0.064	0.137	159.5
Instant3D [24]	28.78	43.06	0.892	0.044	0.102	120.0
ablations from different diffusion backbones (our final method is at the bottom)						
Ours (SD 1.5-inpainting)	27.79	41.33	0.719	0.09	0.599	128.3
Ours (SD 2.0-inpainting)	27.59	41.23	0.729	0.096	0.589	124.5
Ours (Instant3D)	28.57	42.50	**0.894**	**0.043**	**0.097**	121.1
Ours (SDXL-inpainting)	**29.01**	**43.48**	**0.894**	0.045	0.100	**118.4**

Table 1. **Multiview text-to-image inpainting.** This table demonstrates quantitative performance of several baselines and ablations of our method, with respect to metrics that capture prompt adherence, multiview consistency, and visual quality. Results are color-coded worst and best, with best highlighted in **bold**.

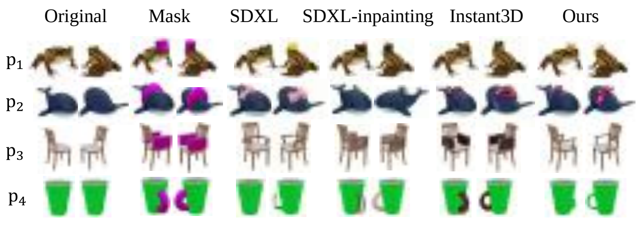


Figure 4. **Comparison to baselines.** We show different inpainting results from different baselines; our multiview inpainting method offers the highest quality while maintaining consistency.

Ablating Masks.

The key ingredient in our training data is the three types of multiview consistent masks. To evaluate the importance of training masks, we ablate on different types of masks, and fine-tune SDXL-inpainting model using only one mask type.

Each row in Table 2 is trained with a different type of mask: Random 2D follows prior 2D inpainting work [38]

Method	Type I	Type II	Type III	I+II+III	User Generated
	FID↓				ClipG↑
Random 2D	145.4	129.6	102.3	131.1	24.22
Type I	131.2	128.6	102.3	121.3	26.24
Type II	170.9	**122.6**	101.7	128.2	25.53
Type III	193.0	148.1	**99.05**	142.2	24.2
I+II+III	**130.0**	124.9	100.0	**118.4**	**26.5**

Table 2. **Mask ablation.** We ablate the choice of 3D masks used during training (rows), and evaluate on different subsets of the benchmark containing only some types of masks (columns).

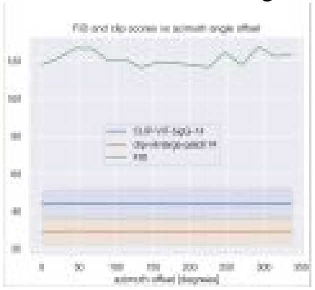
sampling mask independently per image (see inset),

Type I only uses coarse 3D blobs, Type II uses large surface selections, Type III uses local surface patches (see Section 3.2). Finally, the bottom row is our method trained with all types of masks.

To compare these different techniques, we split our benchmark into single-mask vs all-masks data, and compute FID score for each test subset (see columns in Table 2). Unsurprisingly, there is a strong bias in the diagonal, indicating that training on a particular type of mask helps inpainting it at inference time. We note, however, that simple random rectangular masks perform substantially worse. Our fine-tuning on all mask types offers the best performance on arbitrary masks at inference time.

We further take 15 user-generated masks and inpaint them to evaluate how well different types of training masks generalize to real use-cases. We found that our multi-mask training offers the highest Clip-similarity of inpainted results to the user prompts, unfortunately FID score is not meaningful for such a small sample size. We refer the reader to supplemental material where we provide qualitative inpainting results for user-generated masks, demonstrating that multi-mask training data yields the highest inpainting quality.

Generalization to Novel Camera Angles. We note that unlike Instant3d [24], the inpainting model adapts its output to the orientation of the non-masked region, indicating an ability to generalize to unseen azimuth angles. We quantify this by offsetting the 2x2 camera positions by 16 equally spaced azimuth offsets from 0 to 337.5 degrees, generating 500 images for each offset (using the same models as in our benchmark). See how evaluation metrics change for different azimuth offsets in the inset.



The FID, ClipL, and ClipG scores remain consistent throughout. This adaptability is crucial for inpainting, as the orientation of the unmasked portion is unknown beforehand. Fine-tuning follows [24], with multiview renders and masks at fixed angles for each object. Remarkably, the fine-tuned model generalizes to unseen camera orientations and FOV angles, likely due to the strong inductive bias from the masked image and prior knowledge from the baseline diffusion model. We show some visual examples in the supplementary.

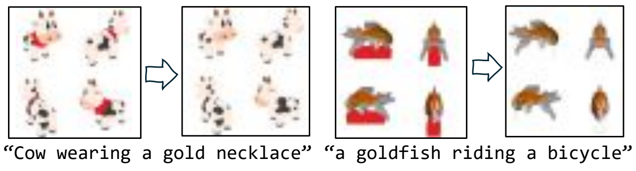


Figure 5. **Failure Cases**. Typical failure cases include failure to adhere to the prompt for thin masks or large masks, which have little inductive bias from the unmasked area.

Limitations. Our multiview inpainting might ignore thin masks (a behavior also observed in 2D inpainting models) (Figure 5, left). Additionally, due to training renders having only a white background, we find that in some cases without existing inductive bias for the mask, the network prefers to generate a white background at the expense of aligning with the prompt (Figure 5, right). These cases can usually be resolved by choosing a different random seed.

5. Applications

While the main contribution of our work is the multiview consistent inpainting method, which can be trivially complemented with a large reconstruction model (LRM) to infer the full 3D shape from the completed views. In this section we explore applications of our technique for editing various neural and traditional 3D representations. We also developed an interactive application where users can load their own 3D shapes, create masks with basic primitives and visualize the results of our method. More information about this application can be found in supplemental.

NeRF Editing. Given an input NeRF, user's mask and a prompt, we use our multiview inpainting to generate the edited views. These views are used as input to the LRM from Instant3D [24] to create the modified representation. We show a few edits in Figure 2 (our result is on the right). In addition to our method we show results for several alternative techniques that allow editing NeRFs. Vox-E [40] and MVEdit [8] do not allow for a user-provided mask, and instead infer the area to be edited using the attention weights from the prompt, which may result in undesirable changes in texture and geometry throughout the object in areas the user would like to keep unchanged. Progressive3d [10] and NeRFFiller [48] struggle with multiview consistency,

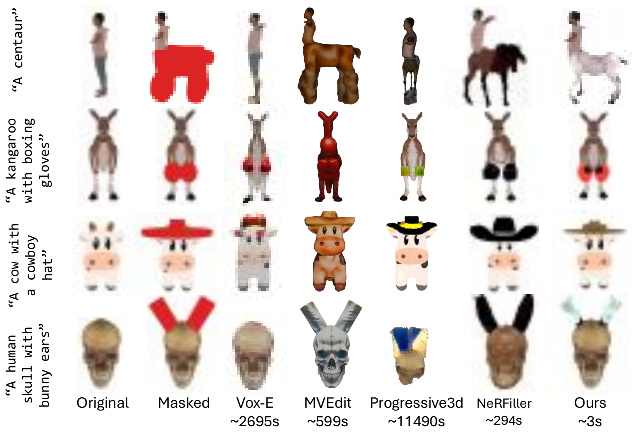


Figure 6. **Application: NeRF editing.** We compare our method (rightmost column) to other methods that allow editing NeRFs, by rendering a front-facing view of the 3D edited asset for all methods. Note that the only method that takes arbitrary masks as input is NeRFiller [48], we also conduct an informal user study with 15 users choosing between our output and NeRFiller for 208 pairs of results. We found that users prefer our method in 86% of the cases. We report average timing for all methods, and confirm Instant3dit is substantially faster than other alternatives.

as they rely on 2D diffusion models for SDS and IDU, respectively. This may result in incorrectly positioned edits, as in the case of the centaur example. Note that our approach is also significantly faster than all existing timelines and produces results in seconds rather than minutes.

We further conduct an informal user preference study between our method and the closest approach that can also inpaint multiview masks, NeRFiller [48]. We showed the masked input and the prompt to 15 users and asked them to choose between two outputs (see supplemental for exact verbiage). Out of 208 pairs, the users preferred our method in 180 cases (86%) vs 28 (14%) for NeRFiller.

Since our multiview inpainting is not tied to any particular 3D representation, our method can also be used to edit Gaussian Splats (GS) by simply using an appropriate LRM [56], see the supplementary for GS editing examples.

Mesh Editing. For the users that work on traditional mesh representation, we propose using MeshLRM [49] with an adaptive remeshing layer [3] that uses the LRM result as a guidance. This leads to a fast and fully controllable mesh editing pipeline that is guaranteed to preserve the original mesh attributes (e.g., UVs, rigging) and triangulation in the unedited parts of the mesh [4]. We noticed that the guidance meshes tend to be over-smoothed due to the relatively low resolution of the LRM's triplane. To maintain fine details, we use a normal estimator [14] directly on the diffusion output and use the normals as targets in the vertex optimization. See Figure 7, supplemental figures, and a video for mesh editing examples.

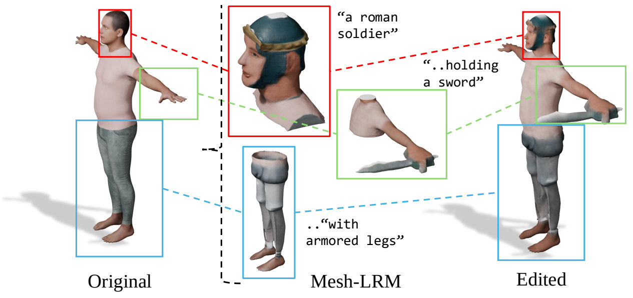
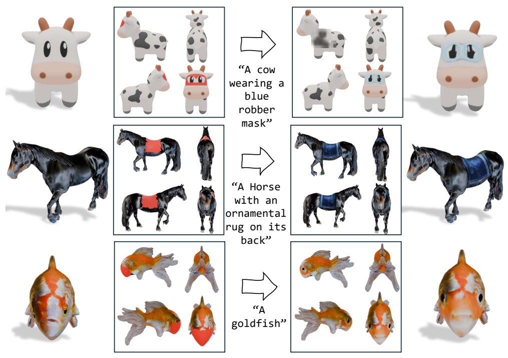


Figure 7. **Application: mesh editing**. Each of the 3 edits shown is performed locally on the original mesh, only modifying regions selected by the user. Locality is achieved by running the ROAR [3] geometry optimization, which takes 20 extra seconds per edit.



"A cow wearing a blue robber mask"
"A Horse with an ornamental rug on its back"
"A goldfish"

Figure 8. **Application: texture editing.** Our method can be used to modify texture on a user-selected region. In this case, we run through our NeRF editing pipeline, but only sample colors from the NeRF in the selected region.

Texture Editing. The user can use our method to edit surface texture details. We use NeRF-LRM to reconstruct a 3D representation and back-project the colors to the user-selected mesh region. Starting with an auto-generated texture from previous work [37] (Fig. 8, first column), the user labels a mask (second column) to either add new texture elements (facemask, saddle) or fix artifacts in the texture (goldfish exhibiting inconsistent left-right colors due to lack of multiview consistency in the previous texturing method).

6. Conclusion

We introduced Instant3dit, a multiview inpainting diffusion model, and show its application to *fast* and *localized* editing of 3D assets. We propose three types of inpainting masks to train the model, corresponding to three types of user edits, and plan to release this dataset. We justify our training strategy and choice of 3d masks by measuring a comprehensive set of metrics: prompt adherence, 3d consistency of the generated content, and generation quality. Instant3dit shows superior quality to the contemporary 3D editing pipelines and is orders of magnitude faster, running as fast as a single image generation using diffusion.

We see clear opportunities to improve this tool further. First, several approaches such as DMD [54] distill diffusion models into one-step or few-step models, which would make Instant3dit run in a fraction of a second and bring a truly interactive editing experience. Second, video diffusion model [5, 43] have shown remarkable 3D consistency and visual quality, which can be harnessed to improve the quality of 3D generation further, although, in terms of speed, they are currently much slower than text-to-image models.

References

[1] Adobe Inc. Adobe photoshop. 3

[2] Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. *CVPR*, 2022. 6

[3] Amir Barda, Yotam Erel, Yoni Kasten, and Amit H. Bermano. Roar: Robust adaptive reconstruction of shapes using planar projections. *arXiv preprint arXiv:2307.00690*, 2023. 4, 8

[4] Amir Barda, Vladimir G. Kim, Noam Aigerman, Amit H. Bermano, and Thibault Groueix. Magicclay: Sculpting meshes with generative neural fields. *SIGGRAPH Asia*, 2024. 1, 2, 3, 8

[5] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, et al. Video generation models as world simulators. *https://openai.com/research/*, 2024. 9

[6] Chenjie Cao, Chaohui Yu, Yanwei Fu, Fan Wang, and Xiangyang Xue. Mvinpainter: Learning multi-view consistent inpainting to bridge 2d and 3d editing. *arXiv preprint arXiv:2408.08000*, 2024. 1, 2, 4

[7] Duygu Ceylan, Valentin Deschaintre, Thibault Groueix, Rosalie Martin, Chun-Hao Huang, Romain Rouffet, Vladimir Kim, and Gaëtan Lassagne. Matatlas: Text-driven consistent geometry texturing and material assignment. *arXiv preprint arXiv:2404.02899*, 2024. 3

[8] Hansheng Chen, Ruoxi Shi, Yulin Liu, Bokui Shen, Jiayuan Gu, Gordon Wetzstein, Hao Su, and Leonidas Guibas. Generic 3d diffusion adapter using controlled multi-view editing. *arXiv preprint arXiv:2403.12032*, 2024. 1, 2, 7

[9] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. *ICCV*, 2023. 1, 2

[10] Xinhua Cheng, Tianyu Yang, Jianan Wang, Yu Li, Lei Zhang, Jian Zhang, and Li Yuan. Progressive3d: Progressively local editing for text-to-3d content creation with complex semantic prompts. *ICLR*, 2024. 1, 2, 7

[11] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-xl: A universe of 10m+ 3d objects. *NeurIPS*, 2023. 2

[12] 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. *CVPR*, 2023. 2, 5

[13] Stephanie Fu, Netanel Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. Dreamsim: Learning new dimensions of human visual similarity using synthetic data. *NeurIPS*, 2023. 6

[14] Xiao Fu, Wei Yin, Mu Hu, Kaixuan Wang, Yuexin Ma, Ping Tan, Shaojie Shen, Dahua Lin, and Xiaoxiao Long. Geowizard: Unleashing the diffusion priors for 3d geometry estimation from a single image. *ECCV*, 2024. 8

[15] William Gao, Noam Aigerman, Groueix Thibault, Vladimir Kim, and Rana Hanocka. Textdeformer: Geometry manipulation using text guidance. *SIGGRAPH*, 2023. 2

[16] Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. *CVPR*, 2023. 2, 3

[17] 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. *arXiv preprint arXiv:1706.08500*, 2018. 6

[18] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. *ICLR*. 3

[19] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 2021. 4

[20] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *NeurIPS*, 2022. 4

[21] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Popa Tiberiu. Clip-mesh: Generating textured meshes from text using pretrained image-text models. *SIGGRAPH Asia*, 2022. 2

[22] Hyunwoo Kim, Itai Lang, Noam Aigerman, Thibault Groueix, Vladimir G. Kim, and Rana Hanocka. Meshup: Multi-target mesh deformation via blended distillation. *3DV*, 2024. 2

[23] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICLR*, 2022. 6

[24] Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model. *ICLR*, 2024. 2, 3, 4, 5, 6, 7

[25] Yuhan Li, Yishun Dou, Yue Shi, Yu Lei, Xuanhong Chen, Yi Zhang, Peng Zhou, and Bingbing Ni. Focaldreamer: Text-driven 3d editing via focal-fusion assembly. *AAAI*, 2024. 2

[26] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *NeurIPS*, 2023. 5

[27] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. Syncdreamer: Generating multiview-consistent images from a single-view image. *ICLR*, 2023. 3

[28] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. Wonder3d: Single image to 3d using cross-domain diffusion. *CVPR*, 2024. 3

[29] Yuanxun Lu, Jingyang Zhang, Shiwei Li, Tian Fang, David McKinnon, Yanghai Tsin, Long Quan, Xun Cao, and Yao Yao. Direct2.5: Diverse text-to-3d generation via multi-view 2.5d diffusion. *CVPR*, 2024. 2, 3, 4

[30] Artem Lukoianov, Haitz Sáez de Ocáriz Borde, Kristjan Greenewald, Vitor Campagnolo Guizilini, Timur Bagautdinov, Vincent Sitzmann, and Justin Solomon. Score distillation via reparametrized ddim. *NeurIPS*, 2024. 3

[31] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. *CVPR*, 2023. 2

[32] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. *CVPR*, 2022. 2

[33] Aryan Mikaeili, Or Perel, Mehdi Safaee, Daniel Cohen-Or, and Ali Mahdavi-Amiri. Sked: Sketch-guided text-based 3d editing. *ICCV*, 2023. 2

[34] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *ICLR*, 2024. 6

[35] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *ICLR*, 2023. 2, 3

[36] 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. *ICML*, 2021. 6

[37] Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. Texture: Text-guided texturing of 3d shapes. *SIGGRAPH*, 2023. 8

[38] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *CVPR*, 2022. 5, 6

[39] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *CVPR*, 2022. 3

[40] Etai Sella, Gal Fiebelman, Peter Hedman, and Hadar Averbuch-Elor. Vox-e: Text-guided voxel editing of 3d objects. *CVPR*, 2023. 1, 2, 7

[41] Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. Zero123++: a single image to consistent multi-view diffusion base model. *arXiv preprint arXiv:2310.15110*, 2023. 3

[42] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *ICLR*, 2024. 3

[43] Genmo Team. Mochi 1. *GitHub repository*, 2024. 9

[44] Christina Tsalicoglou, Fabian Manhardt, Alessio Tonioni, Michael Niemeyer, and Federico Tombari. Textmesh: Generation of realistic 3d meshes from text prompts. *3DV*, 2024. 3

[45] Peng Wang and Yichun Shi. Imagedream: Image-prompt multi-view diffusion for 3d generation. *arXiv preprint arXiv:2312.02201*, 2023. 3

[46] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 2004. 6

[47] Zhengyi Wang, Yikai Wang, Yifei Chen, Chendong Xiang, Shuo Chen, Dajiang Yu, Chongxuan Li, Hang Su, and Jun Zhu. Crm: Single image to 3d textured mesh with convolutional reconstruction model. *CoRR*, 2024. 3

[48] Ethan Weber, Aleksander Holynski, Varun Jampani, Saurabh Saxena, Noah Snavely, Abhishek Kar, and Angjoo Kanazawa. Nerfiller: Completing scenes via generative 3d inpainting. *CVPR*, 2024. 1, 2, 3, 6, 7, 8

[49] Xinyue Wei, Kai Zhang, Sai Bi, Hao Tan, Fujun Luan, Valentin Deschaintre, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Meshlrm: Large reconstruction model for high-quality mesh. *arXiv preprint arXiv:2404.12385*, 2024. 3, 4, 8

[50] Kailu Wu, Fangfu Liu, Zhihan Cai, Runjie Yan, Hanyang Wang, Yating Hu, Yueqi Duan, and Kaisheng Ma. Unique3d: High-quality and efficient 3d mesh generation from a single image. *arXiv preprint arXiv:2405.20343*, 2024. 2, 3, 4

[51] Hanyu Xiang, Qin Zou, Muhammad Ali Nawaz, Xianfeng Huang, Fan Zhang, and Hongkai Yu. Deep learning for image inpainting: A survey. *Pattern Recognition*, 2023. 3

[52] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024. 2

[53] Taoran Yi, Jiemin Fang, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussians by bridging 2d and 3d diffusion models. *CVPR*, 2024. 1, 3

[54] Tianwei Yin, Michael Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. *CVPR*, 2024. 9

[55] Yu Zeng, Zhe L. Lin, Jimei Yang, Jianming Zhang, Eli Shechtman, and Huchuan Lu. High-resolution image inpainting with iterative confidence feedback and guided upsampling. *ECCV*, 2020. 3, 5

[56] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. *ECCV*, 2024. 2, 3, 4, 8

[57] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *CVPR*, 2018. 6

[58] Jingyu Zhuang, Chen Wang, Lingjie Liu, Liang Lin, and Guanbin Li. Dreameditor: Text-driven 3d scene editing with neural fields. *SIGGRAPH Asia*, 2023. 2