

3D Mesh Editing using Masked LRM

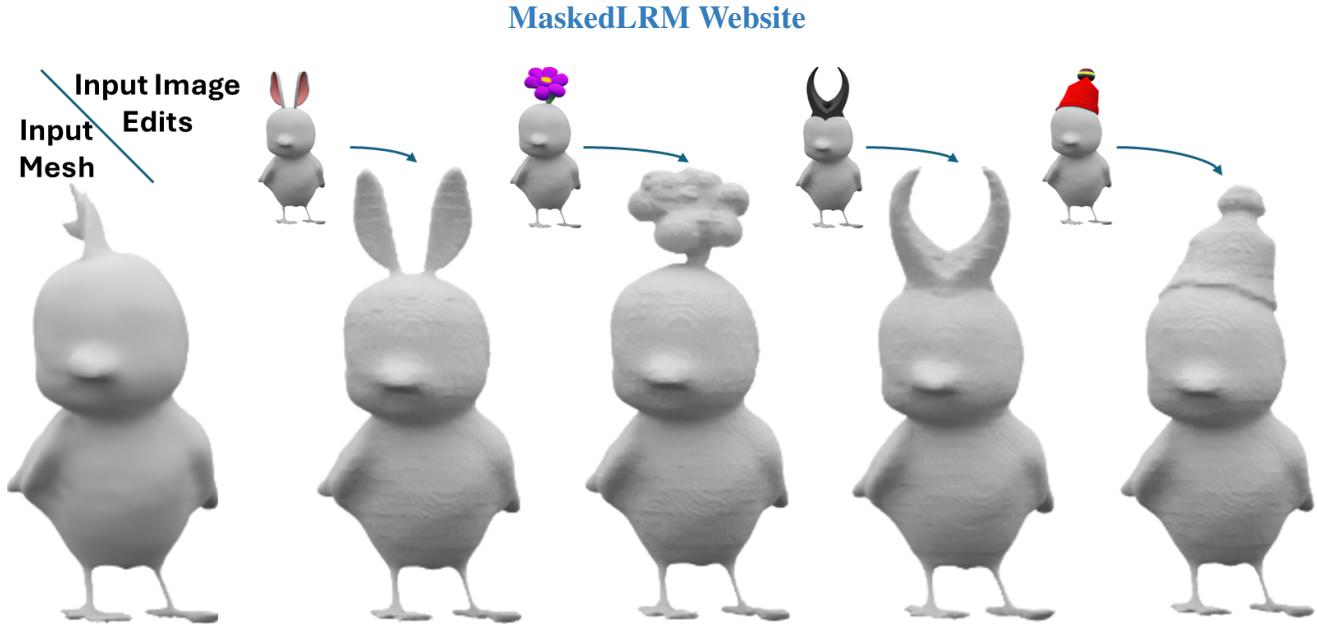
Will Gao^{1*}Zhengqin Li²Dilin Wang²Zhao Dong²Yuchen Fan²Rakesh Ranjan²Aljaz Bozic²Tuur Stuyck²Nikolaos Sarafianos²¹University of Chicago,²Meta Reality Labs

Figure 1. We present a novel and intuitive approach to perform 3D editing through simple interactions. Our method leverages the representation power of large reconstruction models by learning to inpaint randomly masked multi-view images. At inference time, we provide a single input image edit along with a 3D mask and our model faithfully edits the 3D geometry in just a few seconds.

Abstract

We present a novel approach to mesh shape editing, building on recent progress in 3D reconstruction from multi-view images. We formulate shape editing as a conditional reconstruction problem, where the model must reconstruct the input shape with the exception of a specified 3D region, in which the geometry should be generated from the conditional signal. To this end, we train a conditional Large Reconstruction Model (LRM) for masked reconstruction, using multi-view consistent masks rendered from a randomly generated 3D occlusion, and using one clean viewpoint as the conditional signal. During inference, we manually define a 3D region to edit and provide an edited image from a canonical viewpoint to fill in that region. We demonstrate that, in just a single forward pass, our method not only preserves the input geometry in the unmasked region through reconstruction capabilities on par with SoTA, but is also expressive enough to perform a variety of mesh edits from a single image guidance that

past works struggle with, while being 10× faster than the top-performing competing prior work.

1. Introduction

Automated 3D content generation has been at the forefront of computer vision and graphics research, due to applications in various visual mediums like games, animation, simulation, and more recently, virtual and augmented reality. As research on neural methods for content generation has progressed, there has been significant progress in modifying and applying well-studied 2D methods into the 3D domain.

Recent developments in 3D content generation initially followed a similar path to 2D content generation. Operating in 3D voxel space instead of 2D pixel space, models like VAEs [26, 50, 51, 63] and GANs [13, 91] were built, and trained on small-scale datasets [8]. These works often

*This work was conducted during an internship at Meta

demonstrated limited editing capabilities through simple latent operations on their learned representations. Efforts have been made to extend generative diffusion to 3D as well [43, 49, 75, 92]. There has also been work done in generative autoregressive models for 3D, which tokenize 3D data in a unique way [55, 57, 70, 72]. Furthermore, recent research in neural representation techniques such as Neural Radiance Fields (NeRFs) [53] and Gaussian splatting [38] has introduced an entirely new paradigm for 3D content generation.

Despite significant progress in 3D content generation from scratch, research in editing the shape of *existing* 3D models is underdeveloped. Image editing methods benefit from a nearly endless source of data from scraping the internet, while 3D assets typically require a higher level of expertise and specialized tools to create and thus are scarce in comparison. The difference in scale is staggering, with the largest image datasets containing billions of samples [65]. In contrast, the largest 3D datasets contains only millions [19].

A common approach to tackling the issue of 3D data scarcity is to exploit existing text-image foundation models. Many recent efforts in 3D editing involve using these huge models to provide guidance to an optimization process by giving them differentiably rendered images of the manipulated geometry as input [5, 25, 52, 54]. While these approaches demonstrated some success, they face several significant challenges. Firstly, the gradients obtained using foundation models as guidance are often extremely noisy, leading to both unstable and unpredictable optimizations [77]. Furthermore, since these methods often use text as input in lieu of visual input, they are hard to control. Finally, these techniques typically directly optimize properties of an explicit 3D mesh, which severely constrains the type of edits that are possible. For example, it is impossible to add a hole to a shape, since such a modification is not topology-preserving.

A different type of approach that many recent works use is to take a two-stage method, placing the brunt of the “creative effort” onto 2D models, using them to generate and edit content. Then, a model trained to lift 2D images into 3D content is used to produce the final output [42, 83]. Thus, by giving the model edited image inputs, a 3D edit is obtained. However, these methods also have their own limitations. While editing a single image is no longer a challenging task, 3D models that lift a single image into geometry often suffer from ambiguous 3D structure in the occluded regions and do not accurately reconstruct what a shape looks like from every viewpoint. Efforts have been made to train multi-view 2D generation models to combat this problem [48, 68]. Unfortunately, as we demonstrate, editing multi-view images in a consistent manner remains a challenging task and can introduce artifacts corrupting the shape construction.

In this paper, we introduce a method that falls into the second aforementioned direction of lifting 2D images into 3D. Our method differs in that instead of using a 3D model to

simply reconstruct geometry, our model is inherently trained to “inpaint” regions of multi-view images. Specifically, the inputs to our method are a set of masked renders and a single clean conditional image that is provided to infer the missing information from. Our approach solves the issues present in both approaches to shape editing. In contrast to optimization methods, our model is efficient as it constructs shapes in a single, fast forward pass. Furthermore, the output of our model is highly predictable, as it is trained to reconstruct geometry to a high degree of accuracy. Our approach also solves both the consistency and ambiguity problems existing reconstruction methods have, by relying on only a single conditional image while propagating the conditional signal to the rest of the multi-view inputs.

A key challenge of this approach is designing a training procedure that allows the model to learn how to use the conditional information in a multi-view consistent manner. To accomplish this, we employ a new 3D masking strategy. We mask each input view in a consistent manner by rendering an explicit occluding mesh. Then, by supervising both the occluded and unoccluded regions with multi-view reconstruction targets, our model learns to not only fill in the occluded region, but also to accurately reconstruct the rest of the shape. We demonstrate that this training method allows our model to be used downstream for editing tasks while maintaining strong quantitative performance on reconstruction baselines. By manually defining an editing region analogous to the train-time occlusions, and using a single edited canonical view, users can use our model to generate a shape that is faithful both to the original shape, and the edited content. In summary, our contributions are as follows:

- We design a novel conditional LRM trained with a new 3D-consistent multi-view masking strategy. Despite not being our primary intention, our architecture matches SoTA reconstruction metrics, while simultaneously learning to use the conditional input to fill in 3D occlusions.
- We demonstrate that our LRM can be used for 3D shape editing. Our model is much faster than optimization-based editing, and synthesizes arbitrary edits that optimization cannot (*e.g.* genus changes). Furthermore, our model does not suffer from the multi-view consistency and occlusion ambiguity issues that other LRM-based approaches have.

2. Related Work

Large Reconstruction Models: LRM [32] and its recently introduced variants [7, 30, 42, 74, 78, 82–84, 88] showcase the solid capabilities of the transformer architecture for sparse reconstruction. Trained on large-scale 3D [18, 19] and multi-view image datasets [86], these models reconstruct geometry and texture details from sparse inputs or a single image in a feed-forward manner. Most LRMs focus on reconstructing radiance fields, which cannot be consumed by standard graphics pipelines for editing. MeshLRM [82] and

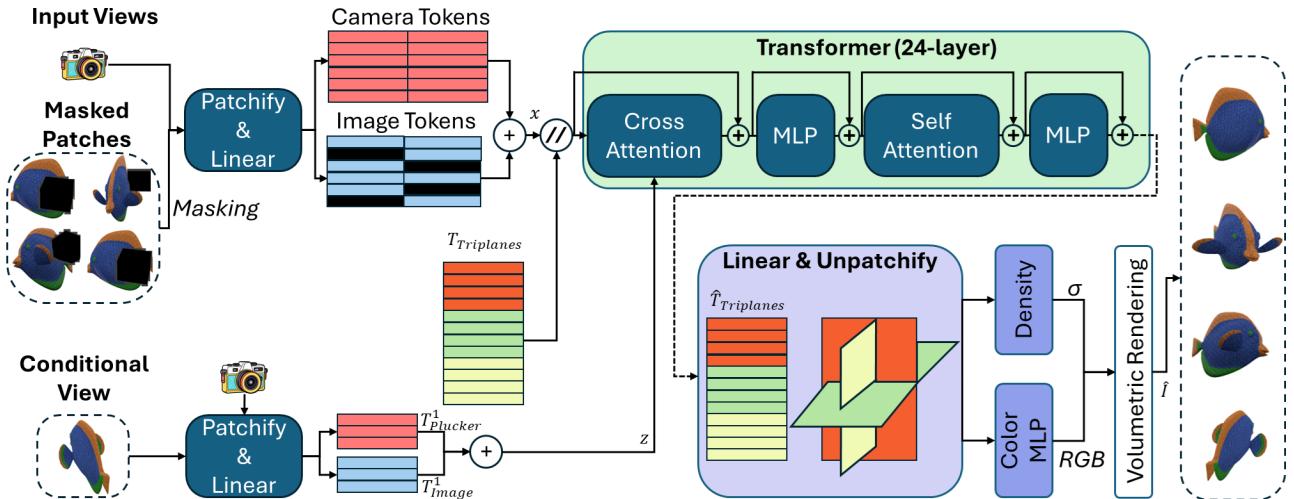


Figure 2. Training Pipeline. The images and camera poses (Plücker rays) are patchified and projected into tokens. A random 3D mask is then generated and tokens corresponding to occluded patches are replaced by a learnable mask token. The camera and image tokens are summed and concatenated with learnable triplane tokens to form the transformer input. A clean, posed image serving as the conditional input is also tokenized, forming the cross-attention input. The output triplane tokens are upsampled and decoded into colors and SDF values, which are then transformed into densities for volumetric rendering.

InstantMesh [83] extract mesh and texture maps, but it remains a challenging to perform shape editing in an intuitive and fast manner. Furthermore, while these models achieve quite high reconstruction quality when given at least four complete views as input [42, 88], the problem is much more ambiguous when given only a single (possibly edited) image [32]. In this work we investigate how to utilize the LRM representation power for 3D shape editing, given a handful incomplete views as input for the shape reconstruction. This makes the reconstructed geometry of the non-edited content match significantly better to the original geometry, while ensuring view-consistency of the edited parts.

Shape Editing: Editing 3D shapes has been an active area of research for at least four decades. Early works focused on deformation [16, 66], cut and paste [6, 62], Laplacian surface editing [24, 46, 69] or fusion [37]. Recent works have tackled this task from different viewpoints depending on the geometry representation, the losses, and the downstream application. Regarding representation, research has been conducted on implicit surface editing [14, 31], voxels [67], mesh-based deformations [25, 39, 64], NeRF [3, 10, 28, 34, 36, 81, 87] and Gaussian splatting [12, 40, 80]. Another line of work has focused on generative local editing using diffusion models. MagicClay [5] proposed to sculpt 3D meshes using a 2D hybrid representation where part of the geometry is frozen and the remaining is optimized using SDS loss [44, 59]. 3D shape editing has been explored in the context of sketches, [47, 56, 71], faces [2, 9, 23, 27, 60] or in an interactive manner [20]. Unlike past work, our approach builds upon the recent advancements of LRMs [32, 82] and introduces a novel architecture trained on multi-view consistent masked

data that enables 3D mesh editing within a seconds.

Masked Vision Models: The original Denoising Autoencoder [76] used random noise as a “masking” method on images during training with the objective of learning better semantic representations. More recently, methods using transformers convert images into sequences and predict unknown pixels [4, 11] which culminated in the development of the Vision Transformer (ViT) [21] as the backbone of modern masked image models. Models like the Masked Autoencoder [29] use ViTs to process randomly masked token sequences, where every token represents a distinct, non-overlapping image patch. Research in diffusion models which also uses random noise as a “masking” procedure, has exploded in popularity, producing increasingly impressive generated images. By taking random Gaussian noise and constraining it to a specific region, diffusion models can be used for image inpainting [17, 41]. Masked autoencoders have been built for 3D data types such as point clouds [35, 58, 90], meshes [45], and NeRFs [33], with each work developing a different way to “patchify” their respective 3D representations. Point clouds have the most natural representation for next token prediction [57, 70], while efforts have also been made into tokenizing triangle meshes for generation as sequences of vertices and faces [55, 72]. Our paper presents a new approach to combine masking with LRMs for editing.

3. Method

Our large reconstruction model, shown in Figure 2, reconstructs a 3D shape from input images. Specifically, the model maps a sparse set of renders from various viewpoints into a latent triplane representation. We sample this representation

to obtain latent features at different 3D points, which are then decoded into distance and RGB values for volumetric rendering. At training, we predict output renders from arbitrary camera poses. During inference, we use marching cubes to produce the reconstructed geometry. Unlike existing LRM, our model uses a conditional branch to accept an additional view of the target shape. The inputs are then corrupted by a random masking procedure, forcing the model to learn to “inpaint” the input views using the conditional branch signal.

3.1. Masked LRM: Architecture

Image and Pose Tokens. The raw input to our model is a set of images with known camera parameters. During both training and inference, the input shapes are oriented in an arbitrary manner. Since we cannot guarantee a canonical viewpoint in the data, we remove the dependence on absolute poses by computing all camera parameters relative to the first randomly selected image which we use as the conditional input. These camera parameters are represented by Plücker rays, forming a 6-channel grid with the same spatial dimensions as RGB pixels. We apply the standard ViT tokenization [21] to the image and the Plücker rays independently, dividing both grids into non-overlapping patches, and linearly projecting them into a high-dimensional embedding.

Masking. After the input images are tokenized, we randomly select a subset of tokens to mask out. For general masked image modeling, [29] demonstrated that dropping out random patches from the encoded image enabled a desirable balance between reconstruction and learned representation quality. However, since our goal is to train a model that fills in the missing geometry from the content of a single clean view, it is not suitable to occlude random patches since they lack correspondence for each input view. Instead, we require a structured, *3D-consistent* form of occlusion. Specifically, we generate a 3D rectangular mesh with uniformly random side lengths. We then render the depth map of this mesh from the same cameras as the input images, obtaining a set of multi-view consistent occlusions. Patches containing pixels that would be occluded by this random mesh are masked out. Instead of dropping the masked patches entirely as in [29], we propose to replace them with a learnable token. This does not suffer the same train-test gap, as occluded images are passed to the model during inference as well. This allows the model to maintain the 3D spatial context of the occlusion. Hence, our masking strategy is specifically designed with downstream editing of an occluded shape in mind.

Model Formulation. Using the above input tokenization and masking procedures, we can write a complete description of our model. Let \mathcal{S} be a shape rendered from n camera poses described by the Plücker ray coordinates $\{\mathbf{C}_i\}_{i=1}^n$ producing RGB renders $\{\mathbf{I}_i\}_{i=1}^n$. The input token sequence

to our model for any image are given by:

$$T_{\text{Image}}^i = \text{PatchEmbed}(\mathbf{I}_i), T_{\text{Plücker}}^i = \text{PatchEmbed}(\mathbf{C}_i), \quad (1)$$

where **PatchEmbed** is the operation of splitting images into non-overlapping patches and applying a linear layer to the channels. We reserve T_{Image}^1 and $T_{\text{Plücker}}^1$ for the clean conditional signal. Now, we sample a random rectangular mesh \mathcal{O} and render it from the same camera poses as \mathcal{S} . Comparing the depth maps of \mathcal{S} and \mathcal{O} , we produce modified tokens $\tilde{T}_{\text{Image}}^i$ where the token for any patch that contains an occluded pixel is replaced by a learnable mask token. Then, the input tokens to the model are constructed as:

$$\mathbf{x} = \left\| \begin{array}{l} (\tilde{T}_{\text{Image}}^i + T_{\text{Plücker}}^i), \\ i=2 \end{array} \right\|, \quad \mathbf{z} = T_{\text{Image}}^1 + T_{\text{Plücker}}^1, \quad (2)$$

where \mathbf{z} is the condition passed to the model and $\|\cdot\|$ the iterated concatenation operation. We choose to add the Plücker ray tokens after masking such that the model can differentiate between different occluded patches. Note that adding a Plücker ray token for each patch means the model does not need a positional embedding to differentiate patches. We use three sequences of learnable tokens $\mathbf{T}_{\text{Triplanes}} = \mathbf{T}_{xy} \parallel \mathbf{T}_{yz} \parallel \mathbf{T}_{xz}$ to produce the predicted triplanes. These tokens are passed to the transformer body of the model, which comprises of iterated standard cross-attention and self-attention operations equipped with MLPs and residual connections:

$$\hat{\mathbf{x}}, \hat{\mathbf{T}}_{\text{Triplanes}} = \text{Self-Att}(\text{Cross-Att}(\mathbf{x} \parallel \mathbf{T}_{\text{Triplanes}}), \mathbf{z}), \quad (3)$$

with \mathbf{x} and $\mathbf{T}_{\text{Triplanes}}$ coming from the previous transformer block (or the input). Finally, we upsample each triplane token to a patch using a single layer MLP, evaluate the learned triplanes at densely sampled points, and decode the latents using MLPs. We obtain predicted images and $\hat{\mathbf{I}}$ pixel-ray opacity maps $\hat{\mathbf{M}}$ through volumetric rendering, and normal maps $\hat{\mathbf{N}}$ by estimating normalized SDF gradients:

$$\begin{aligned} \text{Triplanes} &= \text{MLP}_{\text{Upsample}}(\hat{\mathbf{T}}_{\text{Triplanes}}) \\ \text{SDF}(x, y, z) &= \text{MLP}_{\text{Distance}}(\text{Triplanes}(x, y, z)) \\ \text{RGB}(x, y, z) &= \text{MLP}_{\text{Color}}(\text{Triplanes}(x, y, z)) \\ \sigma &= \text{Density}(\text{SDF}) \\ \hat{\mathbf{N}} &= \text{NormGrad}(\text{SDF}) \\ \hat{\mathbf{I}}, \hat{\mathbf{M}} &= \text{VolRender}(\sigma, \text{RGB}) \end{aligned}$$

where we convert the SDF values to densities σ for rendering following [85]. The learned image tokens $\hat{\mathbf{x}}$ are not used for any remaining task and are thus discarded.

3.2. Supervision

Our LRM is trained with L2 reconstruction and LPIPS perceptual losses. Given a ground truth image \mathbf{I} , normal map \mathbf{N} , and binary silhouette mask \mathbf{M} , we use the following losses:

$$\mathcal{L}_{\text{Recon}} = w_I \|\hat{\mathbf{I}} - \mathbf{I}\|_2^2 + w_N \|\hat{\mathbf{N}} - \mathbf{N}\|_2^2 + w_M \|\hat{\mathbf{M}} - \mathbf{M}\|_2^2$$

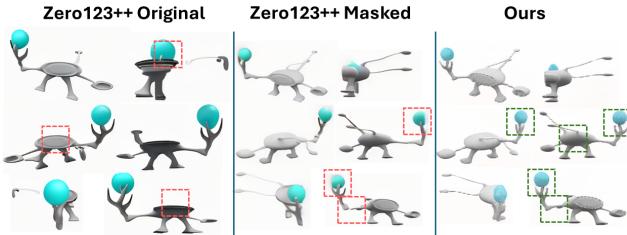


Figure 3. **Addressing Multi-view Artifacts:** Editing using a multi-view diffusion model (Zero-123++) is prone to artifacts and produces inaccurate geometry due to ambiguity caused by occlusion. While explicit masking improves results, it still suffers from bluriness artifacts which propagate into the generated shape when such images are used for 3D reconstruction. Our model, which does not use multi-view diffusion, reconstructs the correct geometry.

$$\mathcal{L}_{\text{Percep}} = w_P \mathcal{L}_{\text{LPIPS}}(\hat{\mathbf{I}}, \mathbf{I})$$

where w_I, w_N, w_M, w_P are tunable weights, and $\mathcal{L}_{\text{LPIPS}}$ is the image perceptual similarity loss proposed in [89]. For the results in this paper, we choose simply $w_I = w_M = w_P = 1$ and $w_N = 0$ or 1 depending on the stage of training.

3.3. Training Stages

Our model is trained in stages following [82], for training efficiency. However, the purpose of our stages differ. Since fully rendering 512×512 output images is computationally expensive, for every stage, we sample a random 128×128 crop from each output image to use as supervision. We maintain the full images for the input throughout every stage.

Stage 1: We downsample the output images to a 256×256 resolution, allowing the random crops to supervise 25% of the image. We use 128 samples per ray for the volumetric rendering. In this initial stage, we observe that the geometric supervision from the normal maps is not yet necessary, so we drop this portion of the reconstruction loss by setting $w_N = 0$ enabling a more efficient backwards pass.

Stage 2: We downsample the output images to 384×384 , meaning that the random crops now only supervise 11% of the image and increase the samples per ray to 512. By increasing the rendering fidelity and decreasing the proportion of the image supervision, we train the model to focus more sharply on geometric details. We observed that without any geometric supervision, the LRM may produce degenerate solutions by generating textures that hide geometric artifacts. Thus, we introduce the normal loss by setting $w_N = 1$.

3.4. Mesh Shape Editing

Since our LRM is trained with 3D-consistent, multi-view occlusions, using the conditional branch to complete the partial observations, it is straightforward to use it for shape editing. Given a shape \mathcal{S} , we manually define an occlusion \mathcal{O} that occludes the region of interest for editing. Then, we edit a representative image within the pixels that are occluded from its camera viewpoint. This may be done a variety of ways

– for our results, we use a text-to-image masked diffusion method [17, 41]. The image edit is used as a conditional signal, while the rest of the occluded images are fed to the main transformer body of the LRM. The LRM is trained to inpaint the occluded region using the content from the conditional branch, and as such it propagates the 2D edit into 3D space. This approach to shape editing is much faster than optimization-based methods (see Table 2), requiring only a single forward pass to lift the desired edit into 3D. Our model also produces more realistic shapes since it is trained on a large collection of scanned objects instead of relying on diffusion model guidance and complex regularizations. It is also far more expressive than optimization-based methods as it can generate arbitrary geometry based on the input condition. For example, it can change the geometric genus of a shape (*e.g.* adding a handle or a hole), which deformation-based optimization methods cannot do as genus changes are not differentiable. Generative methods using LRMs such as InstantMesh [83] rely on a method such as Zero-123++ [68] to generate multi-view images, introducing the risk of view-consistency artifacts. In Figure 3 we provide an example of these artifacts. The original Zero-123++ generates inaccurate views due to the ambiguity of the occluded geometry in the input image. We then restrict Zero-123++ to a masked region in each view, guaranteeing consistency in the rest of the shape (by copying back the original pixels of these regions), but still the model generates severe blurring artifacts. In contrast, our model requires only a single view as conditioning and uses the prior from the dataset to construct the shape in a consistent manner.

4. Experiments

Training Data. We train our Masked LRM on a 480k subset of the Objaverse dataset [18] containing shapes collected from a wide variety of sources. This subset filters out relatively poor quality 3D models¹. Each shape is normalized to fit within a sphere of fixed radius. Our training data consists of 40 512×512 images of each shape, rendered from randomly sampled cameras. We also render the corresponding depth and normal maps for these camera poses. Every iteration, the model inputs and reconstruction targets are chosen randomly from these pre-fixed sets of images.

Evaluation. We evaluate the reconstruction quality of our model on the GSO [22] and ABO [15] datasets and compare the state-of-the-art MeshLRM [82]. We use PSNR, SSIM, and LPIPS on the output renders from novel poses as metrics. To remain consistent with the training setting, we randomly generate a rectangular mask to occlude the input views and provide a different clean view as conditioning. Furthermore, we qualitatively demonstrate the main contribution of our model, the ability to propagate 2D edits from a single viewpoint into 3D. We compare our results to prior works for

¹Sketchfab data were filtered out due their licence

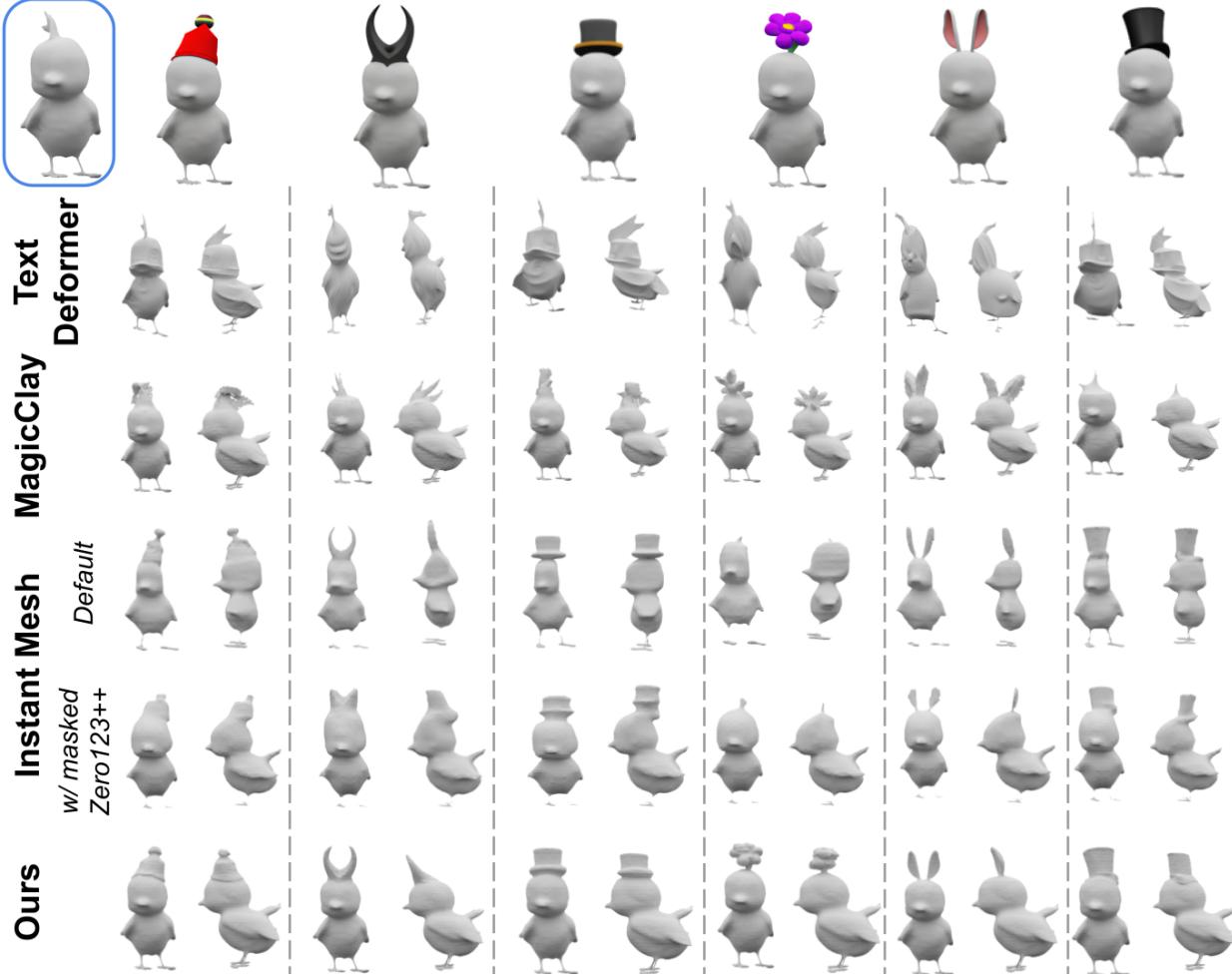


Figure 4. Mesh Editing - Qualitative Comparisons: We compare our approach to other shape editing methods. Given a bird mesh (top left) as well as the various image edits as guidance we demonstrate that our approach is the only one that generates multi-view consistent shape edits that follow the guidance. We omit the colors to clearly visualize the edited geometry.

both text and image based 3D generation.

4.1. Quantitative Comparisons

Table 1 shows novel-view synthesis metrics of our method when compared to MeshLRM. Since our main goal is to edit existing shapes and not to completely generate shapes from scratch, we choose to train our model by randomly selecting 6-8 input views along with one conditional view, giving our LRM a denser input than MeshLRM. We show metrics for both 6 and 8 input views and we compute those on another set of 10 camera poses, different from the input poses. Our method is competitive with the state-of-the-art model on reconstruction, achieving a 2.56 PSNR improvement on the ABO dataset, and a comparable PSNR on GSO. We observe the same phenomenon in perceptual quality measured by LPIPS, where our method significantly outperforms on ABO shapes, and is comparable on GSO shapes. As expected, using 6 views under-performs using 8 views, but only by a

slight margin. Our model achieves performance on par with SoTA on reconstructing a diverse set of output poses, indicating that it has learned to effectively “in-paint” the randomly occluded regions in the input views using context from the available unoccluded signal. Since our end goal is mesh shape editing, it is *not critical that we surpass the reconstruction quality* of prior works, as we only need to ensure a high quality baseline for the output geometry. We further demonstrate qualitatively in Sec. 4.3 that our model indeed learns to inpaint using the conditional signal, instead of only the context from multi-view images, thereby accomplishing feed-forward shape editing through a single view.

4.2. Qualitative Evaluations

Using a bird mesh generated from a text-to-3D model, and several editing targets, we compare our method against other 3D editing methods. Full results are shown in Figure 4. We define a masked region to edit on the head of the bird (omit-

Table 1. Quantitative Evaluation: We evaluate our model using shapes outside of its training set and compare it to the state-of-the-art LRM. We do so using the ABO and GSO shape datasets, reconstructing the meshes from 6 and 8 posed images. Although our main objective is not direct reconstruction of new shapes, and our masking procedure introduces additional difficulty in the task, we still achieve better than SoTA metrics on the ABO dataset, and comparable to SoTA metrics on the GSO dataset.

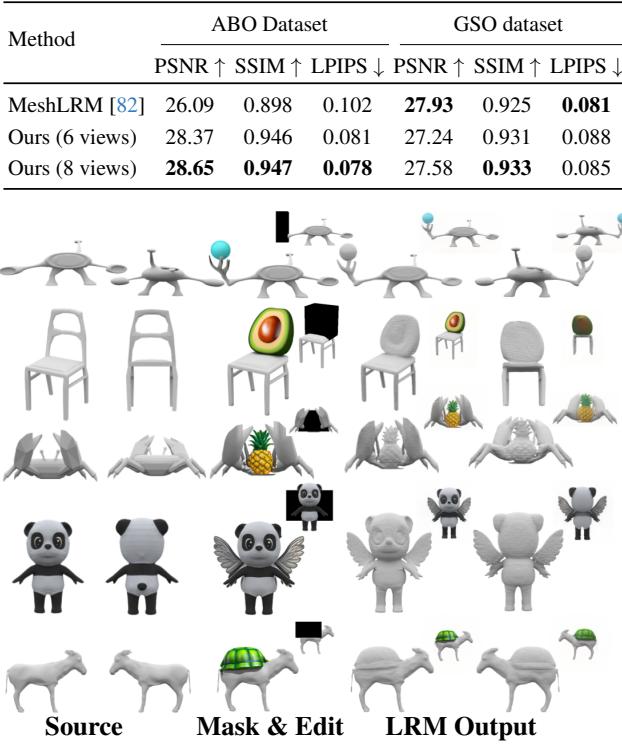


Figure 5. Mesh Editing: The inputs comprise front/back views of the source mesh 1st col and a front image used as the conditional view. The 2nd col shows the masked area rendered from the front (inset) and a 2D edit. The last column shows our generated mesh from the front/back views (Inset: colors from volumetric rendering).

ted from the figure for brevity). Conditional signals provided to our method generated by masked diffusion are shown in the 1st row, and our results are shown in the last row.

Optimization methods. In the 2nd and 3rd rows of Figure 4, we show the results of two text-based mesh optimization methods. Instead of using the edited images themselves, we use the text prompts that we passed to the diffusion model as guidance. The first optimization-based method we compare to (2nd row) is TextDeformer [25] which uses a Jacobian representation [1] for deformations to constrain the edits instead of explicit localization. We observe that TextDeformer struggles with the highly localized nature of our task and globally distorts the mesh, failing to produce an output of acceptable quality in all examples.

The other optimization method we compare to is Mag-

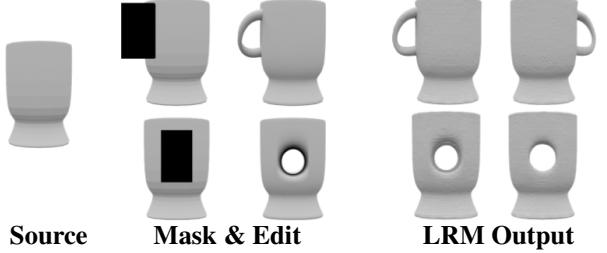


Figure 6. Genus changes: Our method can perform genus-changing edits like adding a handle or a hole to the original vase. We show the output of our model from two opposing views in the third column.

icClay [5], which optimizes both an SDF and a triangle-mesh representation. Note that MagicClay optionally uses a manual “seed” edit so that the optimization task is easier. However, since this requires an additional layer of modeling expertise (*i.e.* a manual user intervention) our method does not require, we opt out of this step for our comparison. In contrast to TextDeformer, MagicClay relies on selecting a subset of vertices to deform to combat noisy SDS [59] gradients. Since we have a 3D mask, we simply choose the vertices that lie within that region. Although this selection serves to localize the editing process, we observe that the deformations are still noisy. In some cases such as the flower (4th column) and rabbit ears (5th column), the editing process produces semantically correct edits. In other cases such as the fedora (3rd column) and top hat (6th column), the optimization process collapses completely. We note that in both cases, noisy gradients from text-guidance result in results in optimizations that are both unpredictable and uncontrollable. In contrast, the output of our LRM is highly predictable from the selected conditional view, which may be re-generated until desirable or even manually edited.

InstantMesh. InstantMesh [83] is an LRM-based pipeline that can be used for editing. It relies on using Zero-123++ [68] to generate multi-view images from a single view and then passing these images to an LRM. We compare our results to two versions of InstantMesh. The first version, shown in the 4th row of Figure 4, simply passes the single edited view directly to InstantMesh. This results in a poorly reconstructed shape that is particularly thin when compared to the ground truth and the output quality suffers due to the inability of Zero-123++ [68] to generate faithful multi-view images, as discussed in Section 3.4. Methods that rely directly on a separate diffusion module to generate the LRM inputs will run the risk of generating artifacts from inaccurate multiview generation. In comparison, our method does not suffer any such reconstruction artifacts since it uses trivially consistent ground truth renders as the main LRM inputs. The second version of InstantMesh, shown in the 5th row, applies the masking procedure from Differential Diffusion [41] to the forward pass of Zero-123++ [68]. Once again, since we define a editing region in 3D, we can restrict the diffusion

Table 2. **Runtime Comparison:** Our method is significantly faster than optimization methods as it is feed-forward and also faster than InstantMesh which must run Zero-123++.

	Optimization-based	LRM-based		
Runtime ↓	TextDeformer [25]	MagicClay [5]	InstantMesh [83]	Ours
	~20mins	~1hour	30sec	< 3sec

process to the pixels corresponding to this region. In this version, we observe that the reconstruction is higher quality. However, artifacts still remain, especially near the edited region, due to artifacts generated by Zero-123++.

4.3. Mesh Editing Characteristics

In Figures 5 and 6, we show more mesh editing examples demonstrating the capabilities of our method. The 1st column shows the source mesh rendered from two different viewpoints. The 2nd column shows the edited conditional image with a render of the original masked region inset. Our LRM accepts the edited view along with a set of occluded ground-truth renders (omitted from the figure), and predicts an SDF. The last column shows the marching-cubes mesh extracted from the output while the insets depict volumetric renders of the predicted SDF.

Expressiveness. The edits throughout this paper show the expressiveness of our method. The meshes used in Figures 1 and 4 as well as in row 1 of Figure 5 are examples of non-standard shapes – a unique bird mesh generated from a text-to-multiview diffusion model [73] and a “Tele-alien” from the COSEG [79] dataset. Despite being novel shapes, our model is able to give the alien a staff and the bird a hat. The other four rows of 5 consist of edits that are “unnatural” – creating an avocado backrest, replacing the body of a crab with a pineapple, giving a panda wings, and giving a donkey a turtle shell. In every example, our method successfully translates the 2D edit into geometry in a realistic manner. The edits in Figure 6 show a critical benefit of our method. Since the final mesh is constructed entirely from the output of a neural network, there are no geometric restrictions on the type of edit we can do. The last two rows demonstrate the ability of our network to change the genus of a shape by adding a handle or a hole through the middle, which would be impossible for geometry optimization-based methods.

Identity Preservation. Although our model discards the initial shape in order to bypass editing limitations, we observe that the LRM still achieves highly faithful reconstructions of the initial geometry outside of the region of interest. This confirms our quantitative observations that our method has near-SoTA reconstruction quality. This also indicates that, due to multi-view masking, our method is able to constrain the edit inside selected 3D region without needing to perform expensive learning over explicit 3D signals.

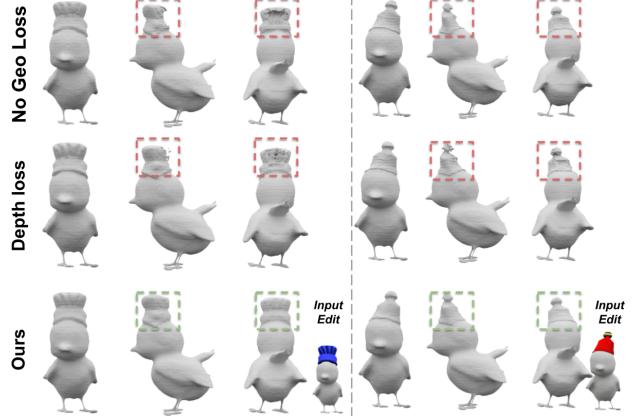


Figure 7. **Impact of Geometric Supervision:** Geometric losses are critical to produce high-quality surfaces. No geometric loss (top) causes severe hole and bump artifacts in the mesh. Depth loss (middle) is not as effective as normal loss (bottom) which allows our model to generate accurate and smooth reconstructions.

4.4. Ablations Studies & Discussion

We investigate the effect of using geometric losses during training. Figure 7 compares three LRMs: a model trained by the pipeline described in Sec. 3, a model trained with no geometric supervision, and a model trained by replacing the normal map loss with a depth map loss. We observe that using no geometric supervision results in poor surface quality, highlighted in the red boxes. Since the main training objective is multi-view image reconstruction, the model easily hallucinates correct geometry using colors, without producing an accurate surface. Supervising the predicted depth somewhat mitigates this issue, but the effect is weak and the surfaces are still incomplete. Normal map supervision gives high quality surfaces as shown in the green boxes.

Runtime: In Table 2 we provide performance comparisons between our approach and 3 top-performing recent works all measured on a single A100 GPU. Our method is not only much faster than both optimization-based approaches [5, 25] as it requires only one forward pass, but it also outperforms LRM approaches such as InstantMesh [83] that make use of a multi-view generation model [68].

Limitations & Future Work: Our method is constrained by the expressiveness of editing in the canonical view. While text-to-image models can create a wide range of results, capturing a specific idea may require significant iteration. Furthermore, while our model accurately reconstructs geometry, it replaces the triangulation with one derived from Marching Cubes, which may be undesirable. Finally, our model is limited by the reconstruction quality of modern LRMs which makes performing edits that require extremely intricate details challenging. Future work could focus on improving localization by developing techniques to merge the existing triangulation with the edited output.

5. Conclusion

In this paper we introduced a new method to perform 3D shape editing. Our work builds upon the recent progress of LRM by introducing a novel multi-view input masking strategy during training. Our LRM is trained to “inpaint” the masked region using a clean conditional viewpoint to reconstruct the missing information. During inference, a user may pass a single edited image as the conditional input, prompting our model to edit the existing shape in just one forward pass. We believe our method is a significant advancement in shape editing, allowing users to create accurate and controllable edits without 3D modeling expertise.

References

- [1] Noam Aigerman, Kunal Gupta, Vladimir G. Kim, Siddhartha Chaudhuri, Jun Saito, and Thibault Groueix. Neural jacobian fields: Learning intrinsic mappings of arbitrary meshes. In *ACM Transactions on Graphics (SIGGRAPH)*, 2022. [7](#)
- [2] Shivangi Aneja, Justus Thies, Angela Dai, and Matthias Nießner. Clipface: Text-guided editing of textured 3d morphable models. In *SIGGRAPH '23 Conference Proceedings*, 2023. [3](#)
- [3] Chong Bao, Yinda Zhang, Bangbang Yang, Tianxing Fan, Zesong Yang, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Sine: Semantic-driven image-based nerf editing with prior-guided editing field. In *The IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2023. [3](#)
- [4] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021. [3](#)
- [5] Amir Barda, Vladimir G. Kim, Noam Aigerman, Amit Bermano, and Thibault Groueix. Magicclay: Sculpting meshes with generative neural fields. In *ACM Transactions on Graphics (SIGGRAPH Asia)*, 2024. [2](#), [3](#), [7](#), [8](#)
- [6] Henning Biermann, Ioana Martin, Fausto Bernardini, and Denis Zorin. Cut-and-paste editing of multiresolution surfaces. *ACM transactions on graphics (TOG)*, 21(3):312–321, 2002. [3](#)
- [7] Mark Boss, Zixuan Huang, Aaryaman Vasishta, and Varun Jampani. Sf3d: Stable fast 3d mesh reconstruction with uv-unwrapping and illumination disentanglement. *arXiv preprint*, 2024. [2](#)
- [8] Andrew Brock, Theodore Lim, James Millar Ritchie, and Nicholas J Weston. Generative and discriminative voxel modeling with convolutional neural networks. In *Neural Information Processing Conference: 3D Deep Learning*, 2016. [1](#)
- [9] Bindita Chaudhuri, Nikolaos Sarafianos, Linda Shapiro, and Tony Tung. Semi-supervised synthesis of high-resolution editable textures for 3d humans. In *CVPR*, 2021. [3](#)
- [10] Jun-Kun Chen, Jipeng Lyu, and Yu-Xiong Wang. NeuralEditor: Editing neural radiance fields via manipulating point clouds. In *CVPR*, 2023. [3](#)
- [11] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International conference on machine learning*, pages 1691–1703. PMLR, 2020. [3](#)
- [12] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting, 2023. [3](#)
- [13] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5939–5948, 2019. [1](#)
- [14] Chong Bao and Bangbang Yang, Zeng Junyi, Bao Hujun, Zhang Yinda, Cui Zhaopeng, and Zhang Guofeng. Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In *European Conference on Computer Vision (ECCV)*, 2022. [3](#)
- [15] Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, et al. Abo: Dataset and benchmarks for real-world 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21126–21136, 2022. [5](#)
- [16] Sabine Coquillart. Extended free-form deformation: A sculpturing tool for 3d geometric modeling. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 187–196, 1990. [3](#)
- [17] Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: Diffusion-based semantic image editing with mask guidance. *arXiv preprint arXiv:2210.11427*, 2022. [3](#), [5](#)
- [18] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. [2](#), [5](#)
- [19] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36, 2024. [2](#)
- [20] Wenqi Dong, Bangbang Yang, Lin Ma, Xiao Liu, Liyuan Cui, Hujun Bao, Yuewen Ma, and Zhaopeng Cui. Coin3d: Controllable and interactive 3d assets generation with proxy-guided conditioning. In *SIGGRAPH*, 2024. [3](#)
- [21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020. [3](#), [4](#)
- [22] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2553–2560. IEEE, 2022. [5](#)
- [23] Anna Frühstück, Nikolaos Sarafianos, Yuanlu Xu, Peter Wonka, and Tony Tung. VIVE3D: Viewpoint-independent video editing using 3D-aware GANs. In *CVPR*, 2023. [3](#)

- [24] Ran Gal, Olga Sorkine, Niloy J Mitra, and Daniel Cohen-Or. iwiress: An analyze-and-edit approach to shape manipulation. In *ACM SIGGRAPH 2009 papers*, pages 1–10, 2009. 3
- [25] William Gao, Noam Aigerman, Groueix Thibault, Vladimir Kim, and Rana Hanocka. Textdeformer: Geometry manipulation using text guidance. In *ACM Transactions on Graphics (SIGGRAPH)*, 2023. 2, 3, 7, 8
- [26] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, the Netherlands, October 11–14, 2016, Proceedings, Part VI 14*, pages 484–499. Springer, 2016. 1
- [27] Xiao Han, Yukang Cao, Kai Han, Xiatian Zhu, Jiankang Deng, Yi-Zhe Song, Tao Xiang, and Kwan-Yee K. Wong. Head-sculpt: Crafting 3d head avatars with text. *arXiv preprint arXiv:2306.03038*, 2023. 3
- [28] Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *CVPR*, 2023. 3
- [29] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022. 3, 4
- [30] Zexin He and Tengfei Wang. Openlrm: Open-source large reconstruction models, 2023. 2
- [31] Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. Spaghetti: Editing implicit shapes through part aware generation. *ACM Transactions on Graphics (TOG)*, 41(4):1–20, 2022. 3
- [32] 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. *arXiv preprint arXiv:2311.04400*, 2023. 2, 3
- [33] Muhammad Zubair Irshad, Sergey Zakharov, Vitor Guizilini, Adrien Gaidon, Zsolt Kira, and Rares Ambrus. Nerf-mae: Masked autoencoders for self-supervised 3d representation learning for neural radiance fields. In *European Conference on Computer Vision*, pages 434–453. Springer, 2025. 3
- [34] Clément Jambon, Bernhard Kerbl, Georgios Kopanas, Stavros Diolatzis, Thomas Leimkühler, and George” Drettakis. Nerf-shop: Interactive editing of neural radiance fields”. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 6(1), 2023. 3
- [35] Jincen Jiang, Xuequan Lu, Lizhi Zhao, Richard Dazaley, and Meili Wang. Masked autoencoders in 3d point cloud representation learning. *IEEE Transactions on Multimedia*, 2023. 3
- [36] Hyunyoung Jung, Seonghyeon Nam, Nikolaos Sarafianos, Sungjoo Yoo, Alexander Sorkine-Hornung, and Rakesh Ranjan. Geometry transfer for stylizing radiance fields. In *CVPR*, 2024. 3
- [37] Takashi Kanai, Hiromasa Suzuki, Jun Mitani, and Fumihiko Kimura. Interactive mesh fusion based on local 3d metamorphosis. In *Graphics interface*, pages 148–156, 1999. 3
- [38] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 2
- [39] Hyun Woo Kim, Itai Lang, Thibault Groueix, Noam Aigerman, Vladimir G. Kim, and Rana Hanocka. Meshup: Multi-target mesh deformation via blended score distillation. In *arXiv preprint*, 2024. 3
- [40] Dmytro Kotovenko, Olga Grebenkova, Nikolaos Sarafianos, Avinash Palwal, Pingchuan Ma, Omid Pourzaeed, Sreyas Mohan, Yuchen Fan, Yilei Li, Rakesh Ranjan, et al. Wast-3d: Wasserstein-2 distance for scene-to-scene stylization on 3d gaussians. In *ECCV*, 2024. 3
- [41] Eran Levin and Ohad Fried. Differential diffusion: Giving each pixel its strength, 2023. 3, 5, 7
- [42] Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model. *arXiv preprint arXiv:2311.06214*, 2023. 2, 3
- [43] Muheng Li, Yueqi Duan, Jie Zhou, and Jiwen Lu. Diffusion-sdf: Text-to-shape via voxelized diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12642–12651, 2023. 2
- [44] 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, 2023. 3
- [45] Yaqian Liang, Shanshan Zhao, Baosheng Yu, Jing Zhang, and Fazhi He. Meshmae: Masked autoencoders for 3d mesh data analysis. In *European Conference on Computer Vision*, pages 37–54. Springer, 2022. 3
- [46] Yaron Lipman, Olga Sorkine, Daniel Cohen-Or, David Levin, Christian Rossi, and Hans-Peter Seidel. Differential coordinates for interactive mesh editing. In *Proceedings Shape Modeling Applications, 2004.*, pages 181–190, 2004. 3
- [47] Feng-Lin Liu, Hongbo Fu, Yu-Kun Lai, and Lin Gao. Sketch-dream: Sketch-based text-to-3d generation and editing. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2024)*, 43(4), 2024. 3
- [48] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9298–9309, 2023. 2, 14
- [49] Zhen Liu, Yao Feng, Michael J Black, Derek Nowrouzezahrai, Liam Paull, and Weiyang Liu. Meshdiffusion: Score-based generative 3d mesh modeling. *arXiv preprint arXiv:2303.08133*, 2023. 2
- [50] Hsien-Yu Meng, Lin Gao, Yu-Kun Lai, and Dinesh Manocha. Vv-net: Voxel vae net with group convolutions for point cloud segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8500–8508, 2019. 1
- [51] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019. 1
- [52] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization

- for meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13492–13502, 2022. 2
- [53] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.
- [54] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. Clip-mesh: Generating textured meshes from text using pretrained image-text models. In *SIGGRAPH Asia 2022 conference papers*, pages 1–8, 2022. 2
- [55] Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *International conference on machine learning*, pages 7220–7229. PMLR, 2020. 2, 3
- [56] Andrew Nealen, Olga Sorkine, Marc Alexa, and Daniel Cohen-Or. A sketch-based interface for detail-preserving mesh editing. In *ACM SIGGRAPH 2005 Papers*, pages 1142–1147, 2005. 3
- [57] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022. 2, 3
- [58] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *European conference on computer vision*, pages 604–621. Springer, 2022. 3
- [59] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022. 3, 7
- [60] Rolandos Alexandros Potamias, Michail Tarasiou Stylianous Ploumpis, and Stefanos Zafeiriou. Shapefusion: A 3d diffusion model for localized shape editing. *arXiv preprint arXiv:2403.19773*, 2024. 3
- [61] Zhangyang Qi, Yunhan Yang, Mengchen Zhang, Long Xing, Xiaoyang Wu, Tong Wu, Dahua Lin, Xihui Liu, Jiaqi Wang, and Hengshuang Zhao. Tailor3d: Customized 3d assets editing and generation with dual-side images, 2024. 14
- [62] Mervi Ranta, Masatomo Inui, Fumihiko Kimura, and Martti Mäntylä. Cut and paste based modeling with boundary features. In *Proceedings on the second ACM symposium on Solid modeling and applications*, pages 303–312, 1993. 3
- [63] Abdurakhmanov Renat and Kerimbek Imangali. Learning latent representations for 3d voxel grid generation using variational autoencoders. In *2024 IEEE AITU: Digital Generation*, pages 169–173. IEEE, 2024. 1
- [64] Nikolaos Sarafianos, Tuur Stuyck, Xiaoyu Xiang, Yilei Li, Jo-van Popovic, and Rakesh Ranjan. Garment3dgen: 3d garment stylization and texture generation. In *3DV*, 2025. 3
- [65] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022. 2
- [66] Thomas W Sederberg and Scott R Parry. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 151–160, 1986. 3
- [67] Etaı Sella, Gal Fiebelman, Peter Hedman, and Hadar Averbuch-Elor. Vox-e: Text-guided voxel editing of 3d objects. In *ICCV*, 2023.
- [68] 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, 2023. 2, 5, 7, 8
- [69] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, page 175–184. Association for Computing Machinery, 2004. 3
- [70] Yongbin Sun, Yue Wang, Ziwei Liu, Joshua Siegel, and Sanjay Sarma. Pointgrow: Autoregressively learned point cloud generation with self-attention. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 61–70, 2020. 2, 3
- [71] Kenshi Takayama, Daniele Panozzo, Alexander Sorkine-Hornung, and Olga Sorkine-Hornung. Sketch-based generation and editing of quad meshes. *ACM Trans. Graph.*, 32(4):97–1, 2013. 3
- [72] Jiaxiang Tang, Zhaoshuo Li, Zekun Hao, Xian Liu, Gang Zeng, Ming-Yu Liu, and Qinsheng Zhang. Edgerunner: Autoregressive auto-encoder for artistic mesh generation. *arXiv preprint arXiv:2409.18114*, 2024. 2, 3
- [73] Shitao Tang, Jiacheng Chen, Dilin Wang, Chengzhou Tang, Fuyang Zhang, Yuchen Fan, Vikas Chandra, Yasutaka Furukawa, and Rakesh Ranjan. Mvdiffusion++: A dense high-resolution multi-view diffusion model for single or sparse-view 3d object reconstruction. *arXiv preprint arXiv:2402.12712*, 2024. 8
- [74] Dmitry Tochilkin, David Pankratz, Zexiang Liu, Zixuan Huang, Adam Letts, Yangguang Li, Ding Liang, Christian Laforte, Varun Jampani, and Yan-Pei Cao. Triposr: Fast 3d object reconstruction from a single image. *arXiv preprint arXiv:2403.02151*, 2024. 2
- [75] Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, Karsten Kreis, et al. Lion: Latent point diffusion models for 3d shape generation. *Advances in Neural Information Processing Systems*, 35:10021–10039, 2022. 2
- [76] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008. 3
- [77] Peihao Wang, Zhiwen Fan, Dejia Xu, Dilin Wang, Sreyas Mohan, Forrest Iandola, Rakesh Ranjan, Yilei Li, Qiang Liu, Zhangyang Wang, et al. Steindreamer: Variance reduction for text-to-3d score distillation via stein identity. *arXiv preprint arXiv:2401.00604*, 2023. 2
- [78] Peng Wang, Hao Tan, Sai Bi, Yinghao Xu, Fujun Luan, Kalyan Sunkavalli, Wenping Wang, Zexiang Xu, and Kai Zhang. Pf-lrm: Pose-free large reconstruction model for joint pose and shape prediction. *arXiv preprint arXiv:2311.12024*, 2023. 2

- [79] Yunhai Wang, Shmulik Asafi, Oliver Van Kaick, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Active co-analysis of a set of shapes. *ACM Transactions on Graphics (TOG)*, 31(6):1–10, 2012. 8
- [80] Yuxuan Wang, Xuanyu Yi, Zike Wu, Na Zhao, Long Chen, and Hanwang Zhang. View-consistent 3d editing with gaussian splatting. In *ECCV*, 2024. 3
- [81] Ethan Weber, Aleksander Holynski, Varun Jampani, Saurabh Saxena, Noah Snavely, Abhishek Kar, and Angjoo Kanazawa. Nerfiller: Completing scenes via generative 3d inpainting. In *CVPR*, 2024. 3, 14
- [82] 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. 2, 3, 5, 7, 14
- [83] 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, 3, 5, 7, 8, 14
- [84] Yinghao Xu, Hao Tan, Fujun Luan, Sai Bi, Peng Wang, Jiahao Li, Zifan Shi, Kalyan Sunkavalli, Gordon Wetzstein, Zexiang Xu, and Kai Zhang. Dmv3d: Denoising multi-view diffusion using 3d large reconstruction model, 2023. 2
- [85] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34:4805–4815, 2021. 4
- [86] Xianggang Yu, Mutian Xu, Yidan Zhang, Haolin Liu, Chongjie Ye, Yushuang Wu, Zizheng Yan, Chenming Zhu, Zhangyang Xiong, Tianyou Liang, et al. Mvimgnet: A large-scale dataset of multi-view images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9150–9161, 2023. 2
- [87] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. Nerf-editing: geometry editing of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18353–18364, 2022. 3
- [88] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. In *European Conference on Computer Vision*, pages 1–19. Springer, 2025. 2, 3
- [89] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 5
- [90] Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li. Point-m2ae: multi-scale masked autoencoders for hierarchical point cloud pre-training. *Advances in neural information processing systems*, 35:27061–27074, 2022. 3
- [91] Xin-Yang Zheng, Yang Liu, Peng-Shuai Wang, and Xin Tong. Sdf-stylegan: Implicit sdf-based stylegan for 3d shape generation. In *Comput. Graph. Forum (SGP)*, 2022. 1
- [92] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5826–5835, 2021. 2

3D Mesh Editing using Masked LRM

Supplementary Material

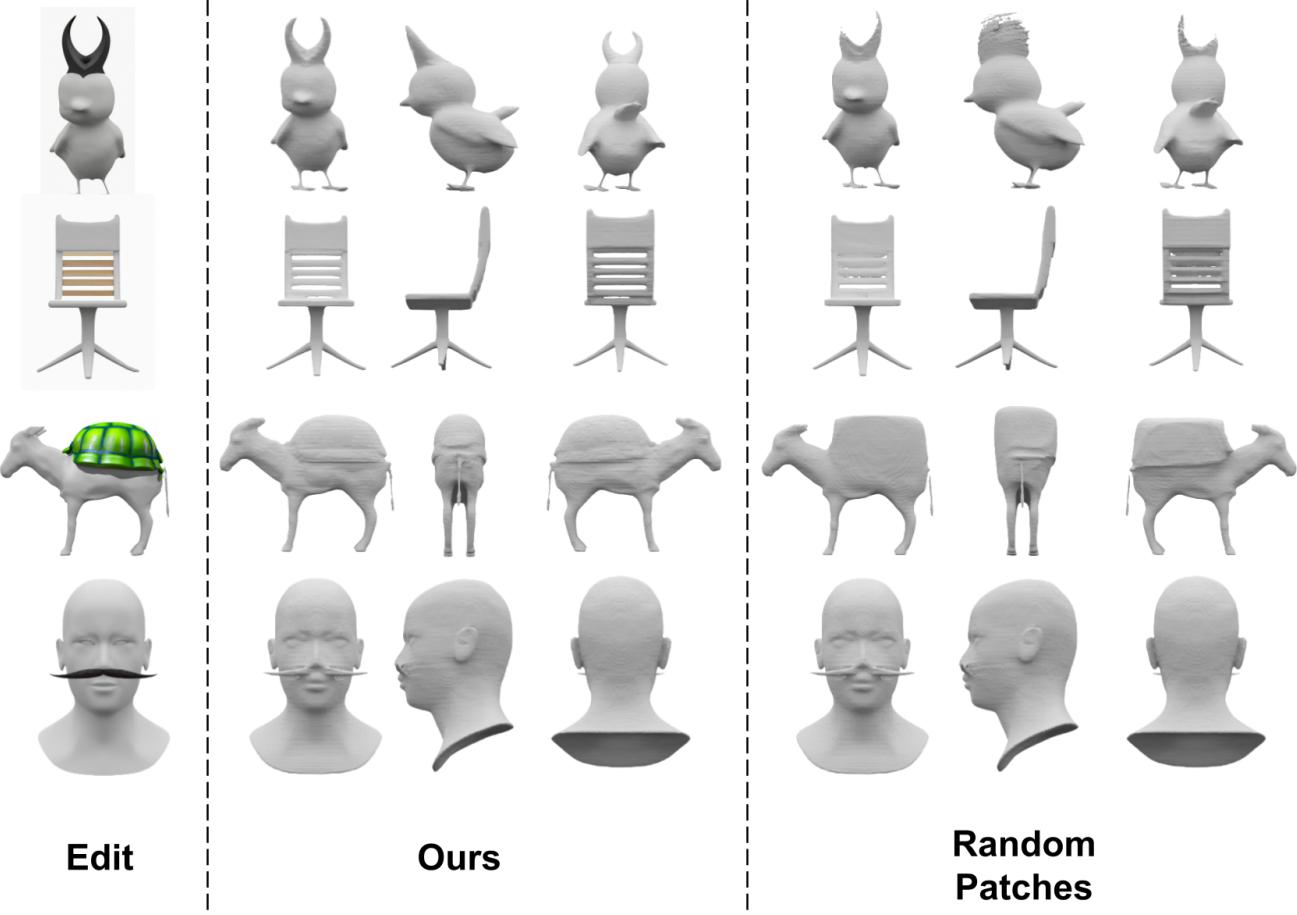


Figure 8. **Impact of Random Masking:** We test our choice of masking strategy by comparing it to masking 25% of patches uniformly at random. The left column shows the conditional image, the middle section shows our results, and the right section shows the results using uniformly random masking. While the model is still capable of generating correct geometry, there is a train-test gap in the masked patches since we define a contiguous 3D region to mask during inference. Thus, the model produces artifacts such as lack of sharp features (in the bird horns and chair slats) in rows 1 and 2, and overall incorrect shape in row 3 (square turtle shell).

6. Introduction

We refer the interested reader to the supplementary video where we provide a plethora of qualitative results of our method. In the following sections we: i) conduct an ablation study that showcases the impact of our masking strategy, ii) showcase qualitative results of 2 recent methods (Ner-filler and Tailor3D) and explain some of their shortcomings, iii) provide implementation details of our method and iv) provide several figures with qualitative results.

7. Masking Ablation

In order to justify our masking strategy, we train our model masking patches uniformly randomly instead of using 3D occlusions. Figure 8 compares meshes extracted from a

model trained using our 3D masking versus masking 25% of patches uniformly at random. We observe that uniformly random patch masking can still generate “roughly correct” shapes, especially adding a moustache to the face in the last row. This is because we add camera pose embeddings after masking, so the model can differentiate masked and non-masked tokens, regardless of their distribution within the image. Furthermore, the reconstruction outside of the masked region is still accurate. However, there still exists a train-test gap between random patches and contiguous patches created by selecting an editing region, which causes significant artifacts in the other three examples. In the first and second rows, we observe a blurring artifact, where the model cannot generate sharp features in the horns on the bird and between the slats of the chair. In the third row, using

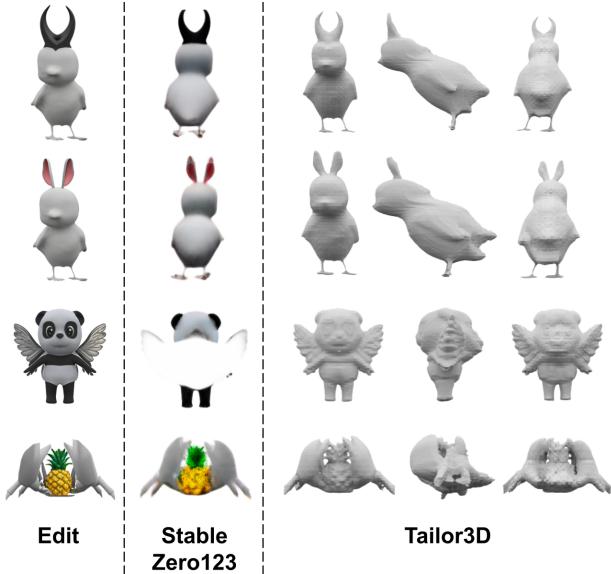


Figure 9. **Tailor3D Meshes:** Tailor3D results with some of the conditions we used for our method throughout the paper. The left column shows the source image. The middle column shows a back view generated by Stable Zero-123 [48]. The right section shows the Tailor3D geometry rendered from 3 viewpoints. In the first two rows, Tailor3D suffers from ambiguity since it only sees the front and the back views and reconstructs an incorrectly elongated body. In the third row and fourth rows, Stable Zero-123 fails to generate a high-quality back view, failing completely for the wings on the panda. We observe the Janus effect in the generated panda and a lack of sharp features in the generated crab, especially viewed from the side.

random patches causes the shape of the turtle shell to be malformed. In comparison, using our masking method produces accurate and sharp geometry in all examples.

8. Comparison to Tailor3D

Tailor3D [61] is a recent work in image-to-3D generation. Similar to InstantMesh [83], Tailor3D relies on a multiview diffusion model, namely Stable Zero-123 [48], to generate inputs that are then lifted into 3D. Tailor3D differs in that it only requires frontal and back views, using a novel transformer design to generate 3D assets from these sparse views. However, Tailor3D cannot replicate our method’s mesh editing results due to two sources of error. First, as with other models that rely on multi-view synthesis, inaccuracies in generating the back view propagate into the 3D model. Second, despite its unique architecture, Tailor3D still suffers from ambiguity artifacts due to the sparse input. Figure 9 demonstrates some of these artifacts. In the first two rows, we observe that Tailor3D fails to recover the geometry of the body of the bird, due to the lack of information in the front and back views, creating an incorrectly elongated shape. In the third row, we see that Stable Zero-123 completely fails

to generate the back view of the wings, leading to a mirroring artifact in the final 3D shape. The fourth row suffers similar issues as the previous three, with the generated view being not only low-quality but also a mirror of the front view instead of a true back view.

9. Comparison to Nerfiller

Nerfiller [81] is a recent work that uses pre-trained image generation models for guidance in order to inpaint masked regions in NeRFs. Nerfiller begins by training a NeRF on unoccluded pixels, and then slowly updates the training set over time via generative inpainting. They adapt their method to image-conditional completion by simply prompting the generative process using a single inpainted image as reference. This is exactly analogous to the input image edits in our method. Figure 10 shows some of the images Nerfiller produces using our image edits as reference. We observe that, although the inpainted images are generally semantically correct, details are inconsistent such as the color of the hats in the first row. Some frames are even missing the hat or rabbit ears entirely. While training a NeRF may tolerate some noise within the training set, this causes blurriness artifacts in the resultant 3D asset and is not suitable for explicit geometry extraction. Furthermore, since Nerfiller repeats this process of training a NeRF and then updating the dataset several times, it is significantly more expensive to run than our method, taking over an hour on an A100 GPU.

10. Additional Implementation Details

Our model implementation details are based mostly on [82]. Our model tokenizes 16×16 sized patches. The token embedding size and transformer width are 1024. The transformer depth is 24 layers. Each attention and cross attention module use multi-head attention with 16 heads. Our model uses LayerNorm and GeLU activations with a Pre-LN architecture.

We trained our models using 64 H100 GPUs with 80GB of RAM each. We use an AdamW optimizer with $(\beta_1, \beta_2) = (0.9, 0.95)$ and a weight decay of 0.01. During stage 1 of training, we train for 30 epochs. For each batch consisting of 12 shapes, we randomly sample the number of input views for the batch uniformly at random between 6 and 8, not including the 1 view for the conditional view. We use another 4 views for supervision. Over the first 1500 iterations, we linearly warm up to a peak learning rate of $4e - 4$ and then use cosine learning rate decay. During stage 2, to account for increased rendering costs, we reduce the batch size to 8 shapes. We train for 20 epochs, with a peak learning rate of $5e - 6$.

11. Additional Qualitative Examples

We present some additional qualitative examples of our model in Figures 11 and 12. We show the network inputs

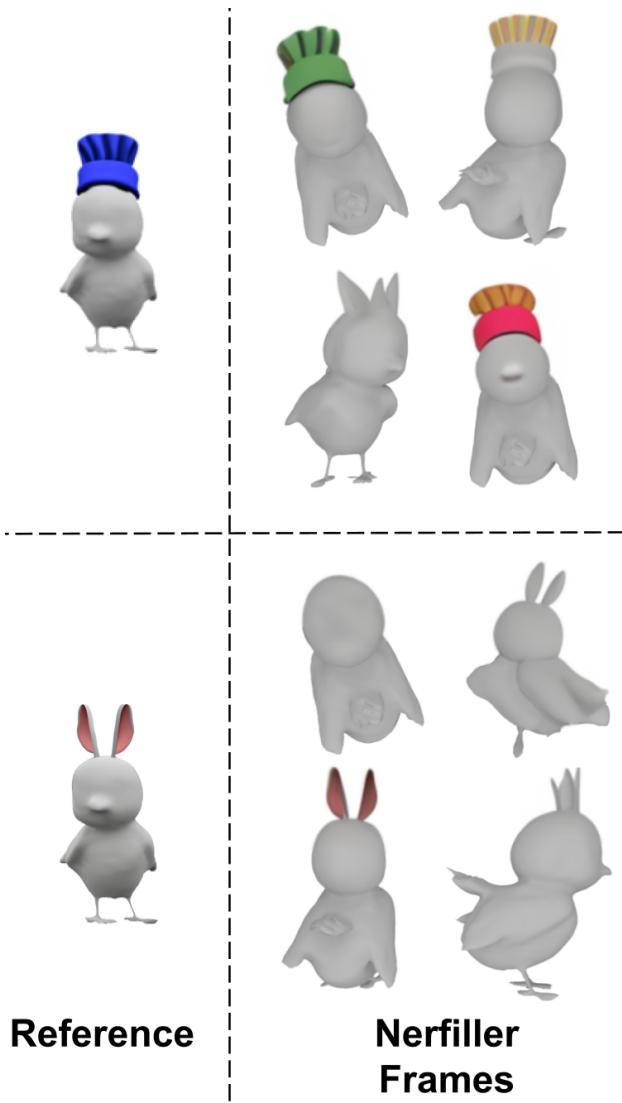


Figure 10. **Nerfiller Images:** Nerfiller results with a couple of bird edits we used for our method. We use their adapted method for reference-image based inpainting. The left column shows the reference image and the right column shows a collection of Nerfiller generated images. We observe that their inpainting method based on pre-trained diffusion models creates noisy output images. The semantics may be correct, but the details can be incorrect *e.g.* incorrect hat colors or completely missing *e.g.* missing hat and ears in a couple of the examples. NeRF training may be tolerant to somewhat noisy input data, but these viewpoints are not suitable for precise geometry reconstruction.

i.e. the masked views and the edited image on the left, and the network outputs *i.e.* the resulting geometry with RGB volumetric renders inset on the right.

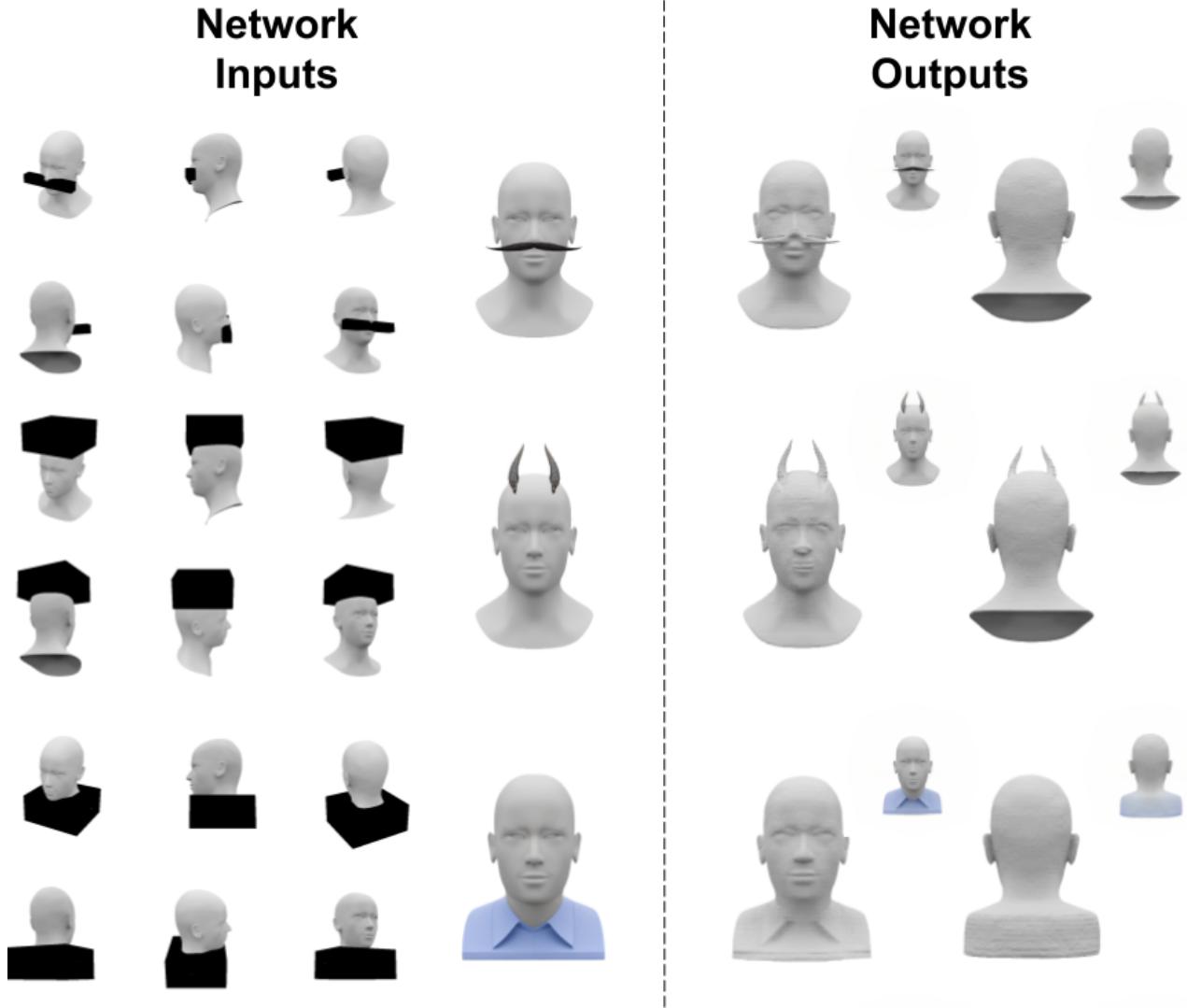


Figure 11. **Additional Qualitative Examples:** Additional qualitative examples editing a person's head. The left section shows the masked views and the edited conditional image. The right section shows the mesh extracted from the network output with the volumetric renders of the SDF inset.

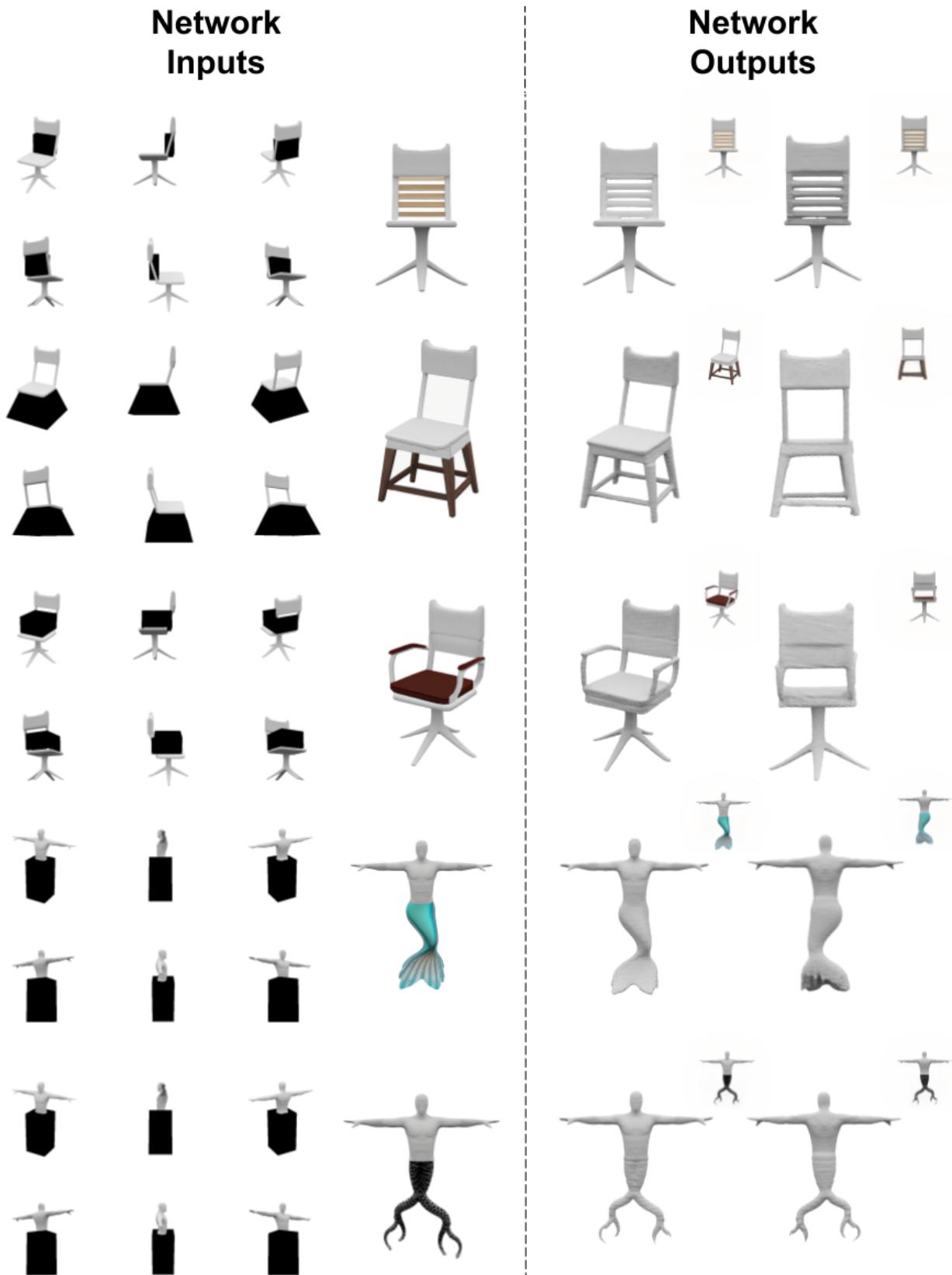


Figure 12. **Additional Qualitative Examples:** Additional qualitative examples editing a chair and a full human. The left section shows the masked views and the edited conditional image. The right section shows the mesh extracted from the network output with the volumetric renders of the SDF inset.