MeshFormer: High-Quality Mesh Generation with 3D-Guided Reconstruction Model

Minghua Liu[*1,2†] **Chong Zeng**[*3‡] **Xinyue Wei**[1,2†] **Ruoxi Shi**[1,2†]
Linghao Chen[2,3†] **Chao Xu**[2,4†] **Mengqi Zhang**[2] **Zhaoning Wang**[5]
Xiaoshuai Zhang[1,2†] **Isabella Liu**[1] **Hongzhi Wu**[3] **Hao Su**[1,2]

[1] UC San Diego [2] Hillbot Inc. [3] Zhejiang University [4] UCLA [5] University of Central Florida

Project Website: `https://meshformer3d.github.io/`

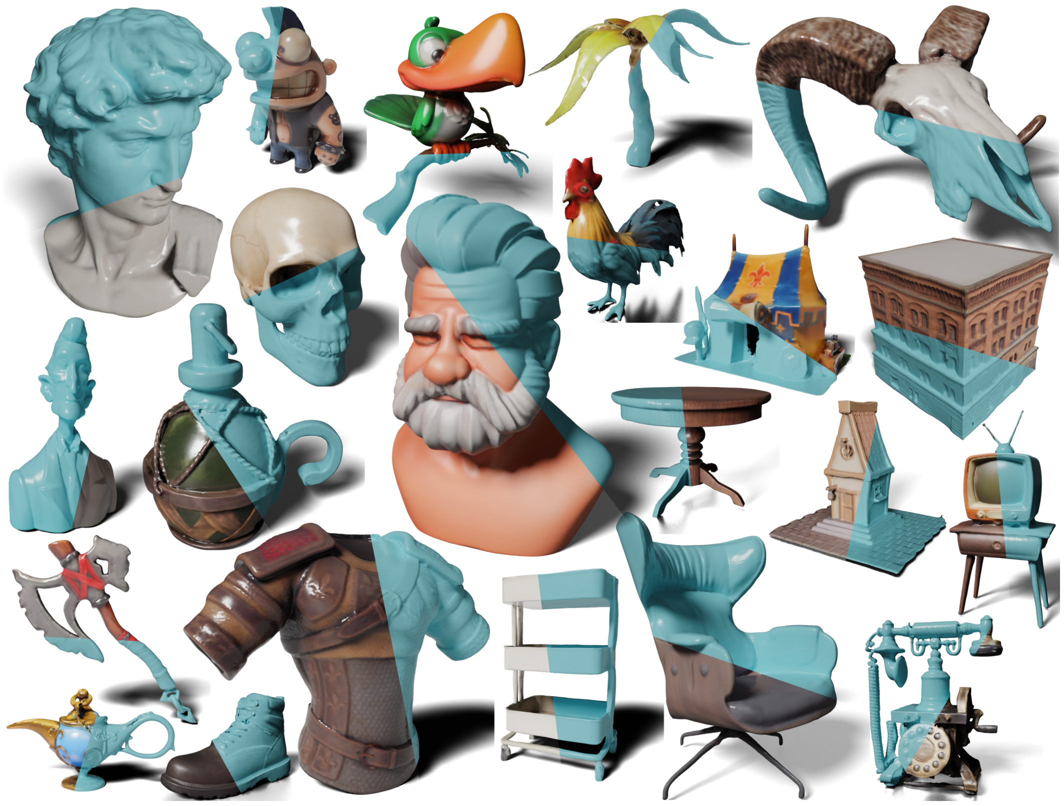


Figure 1: Given a sparse set (e.g., 6) of multi-view RGB images and their normal maps as input, MeshFormer reconstructs high-quality 3D textured meshes with fine-grained, sharp geometric details in a feed-forward pass of just a few seconds. Here, ground truth multi-view RGB and normal images are used as input.

Abstract

Open-world 3D reconstruction models have recently garnered significant attention. However, without sufficient 3D inductive bias, existing methods typically entail expensive training costs and struggle to extract high-quality 3D meshes. In this

[*] Equal contribution. [†] Work done during internship at Hillbot Inc. [‡] Work done during internship at UC San Diego.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

work, we introduce MeshFormer, a sparse-view reconstruction model that explicitly leverages 3D native structure, input guidance, and training supervision. Specifically, instead of using a triplane representation, we store features in 3D sparse voxels and combine transformers with 3D convolutions to leverage an explicit 3D structure and projective bias. In addition to sparse-view RGB input, we require the network to take input and generate corresponding normal maps. The input normal maps can be predicted by 2D diffusion models, significantly aiding in the guidance and refinement of the geometry's learning. Moreover, by combining Signed Distance Function (SDF) supervision with surface rendering, we directly learn to generate high-quality meshes without the need for complex multi-stage training processes. By incorporating these explicit 3D biases, MeshFormer can be trained efficiently and deliver high-quality textured meshes with fine-grained geometric details. It can also be integrated with 2D diffusion models to enable fast single-image-to-3D and text-to-3D tasks.

1 Introduction

High-quality 3D meshes are essential for numerous applications, including rendering, simulation, and 3D printing. Traditional photogrammetry systems [57, 61] and recent neural approaches, such as NeRF [43], typically require a dense set of input views of the object and long processing times. Recently, open-world 3D object generation has made significant advancements, aiming to democratize 3D asset creation by reducing input requirements. There are several prevailing paradigms: training a native 3D generative model using only 3D data [13, 95] or performing per-shape optimization with Score Distillation Sampling (SDS) losses [30, 47]. Another promising direction is to first predict a sparse set of multi-view images using 2D diffusion models [33, 59] and then lift these predicted images into a 3D model by training a feed-forward network [31, 32]. This strategy addresses the limited generalizability of models trained solely on 3D data and overcomes the long runtime and 3D inconsistency of per-shape-optimization-based methods.

While many recent works explore utilizing priors from 2D diffusion models, such as generating consistent multi-view images [59, 60] and predicting normal maps from RGB [12, 37, 59], the feed-forward model that converts multi-view images into 3D remains underexplored. One-2-3-45 [32] leverages a generalizable NeRF method for 3D reconstruction but suffers from limited quality and success rates. One-2-3-45++ [31] improves on this by using a two-stage 3D diffusion model, yet it still struggles to generate high-quality textures or fine-grained geometry. Given that sparse-view reconstruction of open-world objects requires extensive priors, another family of works pioneered by the large reconstruction model (LRM) [16] combines large-scale transformer models with the triplane representation and trains the model primarily using rendering loss. Although straightforward, these methods typically require over a hundred GPUs to train. Moreover, due to their reliance on volume rendering, these methods have difficulty extracting high-quality meshes. For instance, some recent follow-up works [79, 85] implement complex multi-stage "NeRF-to-mesh" training strategies, but the results still leave room for improvement.

In this work, we present MeshFormer, an open-world sparse-view reconstruction model that takes a sparse set of posed images of an arbitrary object as input and delivers high-quality 3D textured meshes with a single forward pass in a few seconds. Instead of representing 3D data as "2D planes" and training a "black box" transformer model optimizing only rendering loss, we find that by incorporating various types of 3D-native priors into the model design, including network architecture, supervision signals, and input guidance, our model can significantly improve both mesh quality and training efficiency. Specifically, we propose representing features in explicit 3D voxels and introduce a novel architecture that combines large-scale transformers with 3D (sparse) convolutions. Compared to triplanes and pure transformers models with little 3D-native design, MeshFormer leverages the explicit 3D structure of voxel features and the precise projective correspondence between 3D voxels and 2D multi-view features, enabling faster and more effective learning.

Unlike previous works that rely on NeRF-based representation in their pipeline, we utilize mesh representation throughout the process and train MeshFormer in a unified, single-stage manner. Specifically, we propose combining surface rendering with additional explicit 3D supervision, requiring the model to learn a signed distance function (SDF) field. The network is trained with high-resolution SDF supervision, and efficient differentiable surface rendering is applied to the extracted meshes for rendering losses. Due to the explicit 3D geometry supervision, MeshFormer enables faster training while eliminating the need for expensive volume rendering and learning an initial coarse NeRF. Furthermore, in addition to multi-view posed RGB images, we propose using corresponding normal maps as input, which can be captured through sensors and photometric techniques [4, 82] or directly estimated by recent 2D vision models [12, 37, 59]. These multi-view normal images provide important clues for 3D reconstruction and fine-grained geometric details. We also task the model with learning a normal texture in addition to the RGB texture, which can then be used to enhance the generated geometry through a traditional post-processing algorithm [44].

Thanks to the explicit 3D-native structure, supervision signal, and normal guidance that we have incorporated, MeshFormer can generate high-quality textured meshes with fine-grained geometric details, as shown in Figure 1. Compared to concurrent methods that require over one hundred GPUs or complex multi-stage training, MeshFormer can be trained more efficiently and conveniently with just eight GPUs over two days, achieving on-par or even better performance. It can also seamlessly integrate with various 2D diffusion models to enable numerous tasks, such as single-image-to-3D and text-to-3D. In summary, our key contributions include:

- We introduce MeshFormer, an open-world sparse-view reconstruction model capable of generating high-quality 3D textured meshes with fine-grained geometric details in a few seconds. It can be trained with only 8 GPUs, outperforming baselines that require over one hundred GPUs.
- We propose a novel architecture that combines 3D (sparse) convolution and transformers. By explicitly leveraging 3D structure and projective bias, it facilitates better and faster learning.
- We propose a unified single-stage training strategy for generating high-quality meshes by combining surface rendering and explicit 3D geometric supervision.
- We are the first to introduce multi-view normal images as input to the feed-forward reconstruction network, providing crucial geometric guidance. Additionally, we propose to predict extra 3D normal texture for geometric enhancement.

2 Related Work

Open-world 3D Object Generation Thanks to the emergence of large-scale 3D datasets [8, 9] and the extensive priors learned by 2D models [50, 51, 55, 56], open-world 3D object generation have recently made significant advancements. Exemplified by DreamFusion [47], a line of work [5, 6, 10, 26, 30, 48, 58, 60, 62, 65, 70, 76] uses 2D models as guidance to generate 3D objects through per-shape optimization with SDS-like losses. Although these methods produce increasingly better results, they are still limited by lengthy runtimes and many other issues. Another line of work [16, 20, 40, 45, 84, 96] trains a feed-forward generative model solely on 3D data that consumes text prompts or single-image inputs. While fast during inference, these methods struggle to generalize to unseen object categories due to the scarcity of 3D data. More recently, works such as Zero123 [33] have shown that 2D diffusion models can be fine-tuned with 3D data for novel view synthesis. A line of work [27, 27, 31, 64, 77, 79, 85], pioneered by One-2-3-45 [32], proposes first predicting multi-view images through 2D diffusion models and then lifting them to 3D through a feed-forward network, effectively addressing the speed and generalizability issues. Many recent works have also explored better strategies to fine-tune 2D diffusion models for enhancing the 3D consistency of multi-view images [14, 17, 23, 34, 36, 49, 59, 60, 69, 72, 80, 81, 89, 91]. In addition to the feed-forward models, the generated multi-view images can also be lifted to 3D through optimizations [14, 34, 37].

Sparse-View Feed-Forward Reconstruction Models When a small baseline between input images is assumed, existing generalizable NeRF methods [35, 52, 68, 88] aim to find pixel correspondences and learn generalizable priors across scenes by leveraging cost-volume-based techniques [3, 38, 90] or transformer-based structures [19, 24, 54, 71, 74]. Some of methods have also incorporated a 2D diffusion process into the pipeline [1, 21, 66]. However, these methods often struggle to handle large baseline settings (e.g., only frontal-view reconstruction) or are limited by a small training set and fail to generalize to open-world objects. Recently, many models [27, 64, 73, 77, 79, 85–87, 92, 94] specifically aimed at open-world 3D object generation have been proposed. They typically build large networks and aim to learn extensive reconstruction priors by training on large-scale 3D datasets [9]. For example, the triplane representation and transformer models are often used. By applying volume rendering or Gaussian splatting [64, 86, 92], they train the model with rendering losses. However, these methods typically require extensive GPUs to train and have difficulty extracting high-quality

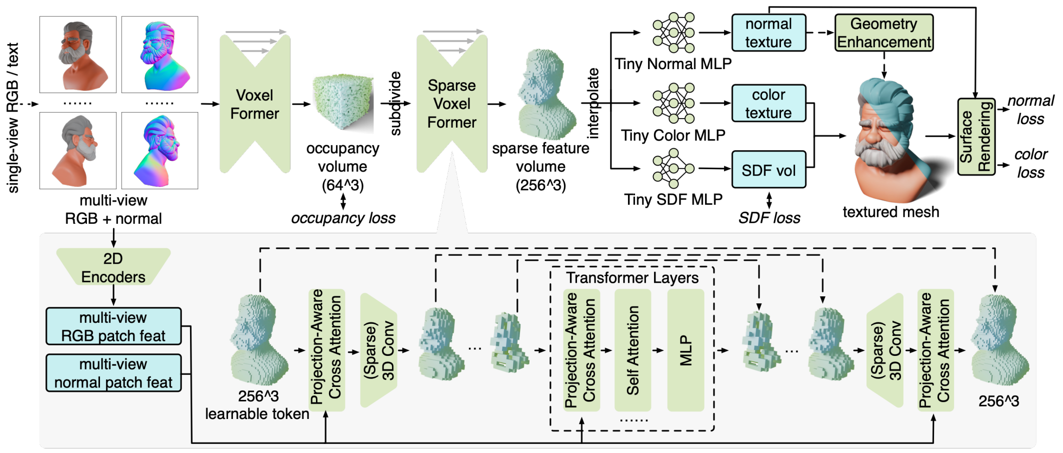


Figure 2: **Pipeline Overview.** MeshFormer takes a sparse set of multi-view RGB and normal images as input, which can be estimated using existing 2D diffusion models. We utilize a 3D feature volume representation, and submodules Voxel Former and Sparse Voxel Former share a similar novel architecture, detailed in the gray region. We train MeshFormer in a unified single stage by combining mesh surface rendering and 512^3 SDF supervision. MeshFormer learns an additional normal texture, which can be used to further enhance the geometry and generate fine-grained sharp geometric details.

meshes. While some recent (concurrent) works [79, 85] utilize multi-stage “NeRF-to-mesh” training strategies to improve the quality, the results still leave room for improvement.

Geometry Guidance for 3D Reconstruction Many recent works have shown that in addition to multi-view RGB images, 2D diffusion models can be fine-tuned to generate other geometric modalities, such as depth maps [75], normal maps [12, 37, 41], or coordinate maps [28, 77]. These additional modalities can provide crucial guidance for 3D generation and reconstruction. While many recent methods utilize these geometric cues as inverse optimization guidance [5, 12, 28, 37, 49, 77], we propose to take normal maps as input in a feed-forward reconstruction model and task the model with generating 3D-consistent normal texture for geometry enhancement of sharp details.

3D Native Representations and Network Architectures in 3D Generation The use of 3D voxel representations and 3D convolutions is common in general 3D generation. However, most recent works focus on 3D-native diffusion [7, 18, 29, 31, 53, 95], one of the key paradigms in 3D generation, which differs from the route taken by MeshFormer. These 3D-diffusion-based methods have some common limitations. For instance, they focus solely on geometry generation and cannot directly predict high-quality textures from the network [7, 18, 29, 31, 53, 95]. Due to the limited availability of 3D data, 3D-native diffusion methods also typically struggle with open-world capabilities and are often constrained to closed-domain datasets (e.g., ShapeNet [2]) in their experiments [7, 29, 95].

In MeshFormer, our goal is to achieve direct high-quality texture generation while handling arbitrary object categories. Therefore, we adopt a different approach: sparse-view feed-forward reconstruction, as opposed to 3D-native diffusion. In this specific task setting, more comparable works are recent LRM-style methods [64, 67, 79, 85]. However, most of these methods rely on a combination of triplane representation and large-scale transformers. In this paper, we demonstrate that 3D-native representations and networks can not only be used in 3D-native diffusion but can also be combined with differentiable rendering to train a feed-forward sparse-view reconstruction model using rendering losses. In open-world sparse-view reconstruction, we are not limited to the triplane representation. Instead, 3D-native structures (e.g., voxels), network architectures, and projective priors can facilitate more efficient training, significantly reducing the required training resources. While scalable networks are necessary to learn extensive priors, scalability is not exclusive to triplane-based transformers. By integrating 3D convolutions with transformer layers, scalability can also be achieved.

3 Method

As shown in Figure 2, MeshFormer takes a sparse set of posed multi-view RGB and normal images as input and generates a high-quality textured mesh in a single feed-forward pass. In the following sections, we will first introduce our choice of 3D representation and a novel model architecture that combines large-scale transformers with 3D convolutions (Sec. 3.1). Then, we will describe our training objectives, which integrate surface rendering and explicit 3D SDF supervision (Sec. 3.2). Last but not least, we will present our normal guidance and geometry enhancement module, which plays a crucial role in generating high-quality meshes with fine-grained geometric details (Sec. 3.3).

3.1 3D Representation and Model Architecture

Triplane vs. 3D Voxels Open-world sparse-view reconstruction requires extensive priors, which can be learned through a large-scale transformer. Prior arts [27, 67, 77, 79, 85] typically utilize the triplane representation, which decomposes a 3D neural field into a set of 2D planes. While straightforward for processing by transformers, the triplane representation lacks explicit 3D spatial structures and makes it hard to enable precise interaction between each 3D location and its corresponding 2D projected pixels from multi-view images. For instance, these methods often simply apply self-attention across all triplane patch tokens and cross-attention between triplane tokens and all multi-view image tokens. This all-to-all attention is not only costly but also makes the methods cumbersome to train. Moreover, the triplane representation often shows results with notable artifacts at the boundaries of patches and may suffer from limited expressiveness for complex structures. Consequently, we choose the 3D voxel representation instead, which explicitly preserves the 3D spatial structures.

Combining Transformer with 3D Convolution To leverage the explicit 3D structure and the powerful expressiveness of a large-scale transformer model while avoiding an explosion of computational costs, we propose VoxelFormer and SparseVoxelFormer, which follow a 3D UNet architecture while integrating a transformer at the bottleneck. The overall idea is that we use local 3D convolution to encode and decode a high-resolution 3D feature volume, while the global transformer layer handles reasoning and memorizing priors for the compressed low-resolution feature volume. Specifically, as shown in Figure 2, a 3D feature volume begins with a learnable token shared by all 3D voxels. With the 3D voxel coordinates, we can leverage the projection matrix to enable each 3D voxel to aggregate 2D local features from multi-view images via a projection-aware cross-attention layer. By iteratively performing projection-aware cross-attention and 3D (sparse) convolution, we can compress the 3D volume to a lower-resolution one. After compression, each 3D voxel feature then serves as a latent token, and a deep transformer model is applied to a sequence of all 3D voxel features (position encoded) to enhance the model's expressiveness. Finally, we use the convolution-based inverse upper branch with skip connection to decode a 3D feature volume with the initial high resolution.

Projection-Aware Cross Attention Regarding 3D-2D interaction, the input multi-view RGB and normal images are initially processed by a 2D feature extractor, such as a trainable DINOv2 [46], to generate multi-view patch features. While previous cost-volume-based methods [3, 38] typically use mean or max pooling to aggregate multi-view 2D features, these simple pooling operations might be suboptimal for addressing occlusion and visibility issues. Instead, we propose a projection-aware cross-attention mechanism to adaptively aggregate the multi-view features for each 3D voxel. Specifically, we project each 3D voxel onto the m views to interpolate m RGB and normal features. We then concatenate these local patch features with the projected RGB and normal values to form m 2D features. In the projection-aware cross-attention module, we use the 3D voxel feature to calculate a query and use both the 3D voxel feature and the m 2D features to calculate $m+1$ keys and values. A cross-attention is then performed for each 3D voxel, enabling precise interaction between each 3D location and its corresponding 2D projected pixels, and allowing adaptive aggregation of 2D features, which can be formulated as:

$$v \leftarrow \mathrm{CrossAttention}(Q=\{v\}, K=\{p_i^v\}_{i=1}^m + \{v\}, V=\{p_i^v\}_{i=1}^m + \{v\}) \tag{1}$$

Where v denotes a 3D voxel feature, and p_i^v denotes its projected 2D pixel feature from view i, which is a concatenation of the RGB feature f_i^v, the normal feature g_i^v, and the RGB and normal values c_i^v and n_i^v, respectively.

Coarse-to-Fine Feature Generation As shown in Fig. 2, to generate a high-resolution 3D feature volume that captures the fine-grained details of 3D shapes, we follow previous work [31, 95] by employing a coarse-to-fine strategy. Specifically, we first use VoxelFormer, which is equipped with full 3D convolution, to predict a low-resolution (e.g., 64^3), coarse 3D occupancy volume. Each voxel in this volume stores a binary value indicating whether it is close to the surface. The predicted occupied voxels are then subdivided to create higher-resolution sparse voxels (e.g., 256^3). Next, we utilize a second module, SparseVoxelFormer, which features 3D sparse convolution [63], to predict features for these sparse voxels. After this, we trilinearly interpolate the 3D feature of any near-surface 3D point, which encodes both geometric and color information, from the high-resolution sparse feature volume. The features are then fed into various MLPs to learn the corresponding fields.

3.2 Unified Single-Stage Training: Surface Rendering with SDF Supervision

Existing works typically use NeRF [42] and volume rendering or 3D Gaussian splatting [22] since they come with a relatively easy and stable learning process. However, extracting high-quality meshes from their results is often non-trivial. For example, directly applying Marching Cubes [39] to density fields of learned NeRFs typically generates meshes with many artifacts. Recent methods [78, 79, 85] have designed complex, multi-stage "NeRF-to-mesh" training with differentiable surface rendering, but the generated meshes still leave room for improvement. On the other hand, skipping a good initialization and directly learning meshes from scratch using purely differentiable surface rendering losses is also infeasible, as it is highly unstable to train and typically results in distorted geometry.

In this work, we propose leveraging explicit 3D supervision in addition to 2D rendering losses. As shown in Figure 2, we task MeshFormer with learning a signed distance function (SDF) field supervised by a high-resolution (e.g., 512^3) ground truth SDF volume. The SDF loss provides explicit guidance for the underlying 3D geometry and facilitates faster learning. It also allows us to use mesh representation and differentiable surface rendering from the beginning without worrying about good geometry initialization or unstable training, as the SDF loss serves as a strong regularization for the underlying geometry. By combining surface rendering with explicit 3D SDF supervision, we train MeshFormer in a unified, single-stage training process. As shown in Figure 2, we employ three tiny MLPs that take as input the 3D feature interpolated from the 3D sparse feature volume to learn an SDF field, a 3D color texture, and a 3D normal texture. We extract meshes from the SDF volume using dual Marching Cubes [39] and employ NVDiffRast [25] for differentiable surface rendering. We render both the multi-view RGB and normal images and compute the rendering losses, which consist of both the MSE and perceptual loss terms. As a result, our training loss can be expressed as:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{MSE}}^{\text{color}} + \lambda_2 \mathcal{L}_{\text{LPIPS}}^{\text{color}} + \lambda_3 \mathcal{L}_{\text{MSE}}^{\text{normal}} + \lambda_4 \mathcal{L}_{\text{LPIPS}}^{\text{normal}} + \lambda_5 \mathcal{L}_{\text{occ}} + \lambda_6 \mathcal{L}_{\text{SDF}} \tag{2}$$

where L_{occ} and L_{SDF} are MSE losses for occupancy and SDF volumes, and λ_i denotes the weight of each loss term. Note that we do not use mesh geometry to derive normal maps; instead, we utilize the learned normal texture from the MLP, which will be detailed later.

3.3 Fine-Grained Geometric Details: Normal Guidance and Geometry Enhancement

Without dense-view correspondences, 3D reconstruction from sparse-view RGB images typically struggles to capture geometric details and suffers from texture ambiguity. While many recent works [27, 79, 85] attempt to employ large-scale models to learn mappings from RGB to geometric details, this typically requires significant computational resources. Additionally, these methods are primarily trained using 3D data, but it's still uncertain whether the scale of 3D datasets is sufficient for learning such extensive priors. On the other hand, unlike RGB images, normal maps explicitly encode geometric information and can provide crucial guidance for 3D reconstruction. Notably, open-world normal map estimation has achieved great advancements. Many recent works [12, 37, 59] demonstrate that 2D diffusion models, trained on billions of natural images, embed extensive priors and can be fine-tuned to predict normal maps. Given the significant disparity in data scale between 2D and 3D datasets, it may be more effective to use 2D models first for generating geometric guidance.

Input Normal Guidance As shown in Figure 2, in addition to multi-view RGB images, MeshFormer also takes multi-view normal maps as input, which can be generated using recent open-world normal estimation models [12, 37, 59]. In our experiments, we utilize Zero123++ v1.2 [59], which trains an additional ControlNet [93] over the multi-view prediction model. The ControlNet takes multi-view RGB images, predicted by Zero123++, as a condition and produces corresponding multi-view normal maps, expressed in the camera coordinate frame. Given these maps, MeshFormer first converts them to a unified world coordinate frame, and then treats them similarly to the multi-view RGB images, using projection-aware cross-attention to guide 3D reconstruction. According to our experiments (Sec. 4.4), the multi-view normal maps enable the networks to better capture geometry details, and thus greatly improve final mesh quality.

Geometry Enhancement While the straightforward approach of deriving normal maps from the learned mesh and using a normal loss to guide geometry learning has been commonly used, we find that this approach makes our mesh learning less stable. Instead, we propose learning a 3D normal texture, similar to a color texture, using a separate MLP. By computing the normal loss for MLP-queried normal maps instead of mesh-derived normal maps, we decouple normal texture learning

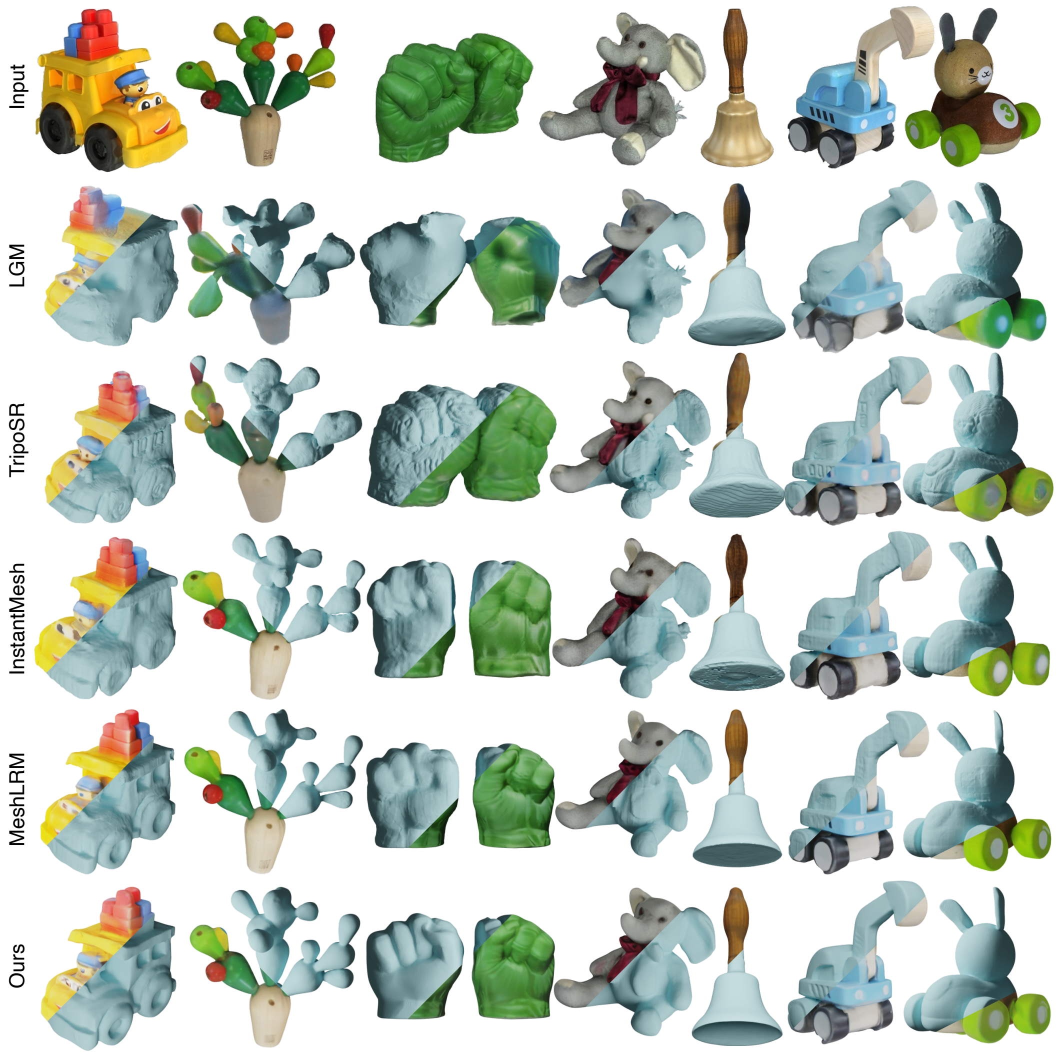


Figure 3: **Qualitative Examples of Single Image to 3D (GSO dataset).** Both the textured and textureless mesh renderings are shown. Please zoom in to examine details and mesh quality, and refer to the supplemental material for results of One-2-3-45++ [31] and CRM [77].

from underlying geometry learning. This makes the training more stable, as it is easier to learn a sharp 3D normal map than to directly learn a sharp mesh geometry. The learned 3D normal texture can be exported with the mesh, similar to the color texture, to support various graphics rendering pipelines. In applications that require precise 3D geometry, such as 3D printing, the learned normal texture can also be used to refine the mesh geometry with traditional algorithms. Specifically, during inference, after extracting a 3D mesh from the SDF volume, we utilize a post-processing algorithm [44] that takes as input the 3D positions of the mesh vertices and the vertex normals estimated from the MLP. The algorithm adjusts the mesh vertices to align with the predicted normals in a few seconds, further enhancing the geometry quality and generating sharp geometric details, as shown in Figure 5.

4 Experiments

4.1 Implementation Details and Evaluation Settings

Implementation Details We trained MeshFormer on the Objaverse [9] dataset. The total number of network parameters is approximately 648 million. We trained the model using 8 H100 GPUs for about one week (350k iterations) with a batch size of 1 per GPU, although we also show that the model can achieve similar results in just two days. Please refer to the supplementary for more details.

Table 1: **Quantitative Results of Single Image to 3D.** Evaluated on the 1,030 and 1,038 3D shapes from the GSO [11] and the OmniObject3D [83] datasets, respectively. One-2-3-45++ [31], InstantMesh [85], MeshLRM [79], and our method all take the same multi-view RGB images predicted by Zero123++ [59] as input. CD denotes Chamfer Distance.

Method	GSO [11]				OmniObject3D [83]			
	F-Score ↑	CD ↓	PSNR ↑	LPIPS ↓	F-Score ↑	CD ↓	PSNR ↑	LPIPS ↓
One-2-3-45++ [31]	0.936	0.039	20.97	0.21	0.871	0.054	17.08	0.31
TripoSR [67]	0.896	0.047	19.85	0.26	0.895	0.048	17.68	0.28
CRM [77]	0.886	0.051	19.99	0.27	0.821	0.065	16.01	0.34
LGM [64]	0.776	0.074	18.52	0.35	0.635	0.114	14.75	0.45
InstantMesh [64]	0.934	0.037	20.90	0.22	0.889	0.049	17.61	0.28
MeshLRM [79]	<u>0.956</u>	<u>0.033</u>	<u>21.31</u>	**0.19**	<u>0.910</u>	<u>0.045</u>	<u>18.10</u>	**0.26**
Ours	**0.963**	**0.031**	**21.47**	<u>0.20</u>	**0.914**	**0.043**	**18.14**	<u>0.27</u>

Evaluation Settings We evaluate the methods on two datasets: GSO [11] and OmniObject3D [83]. Both datasets contain real-scanned 3D objects that were not seen during training. For the GSO dataset, we use all 1,030 3D shapes for evaluation. For the OmniObject3D dataset, we randomly sample up to 5 shapes from each category, resulting in 1,038 shapes for evaluation. We utilize both 2D and 3D metrics. For 3D metrics, we use both the F-score and Chamfer distance (CD), calculated between the predicted meshes and ground truth meshes, following [31, 85]. For 2D metrics, we compute both PSNR and LPIPS for the rendered color images. Since each baseline may use a different coordinate frame for generated results, we carefully align the predicted meshes of all methods to the ground truth meshes before calculating the metrics. Please refer to the supplemental material for more details.

4.2 Comparison with Single/Sparse-View to 3D Methods

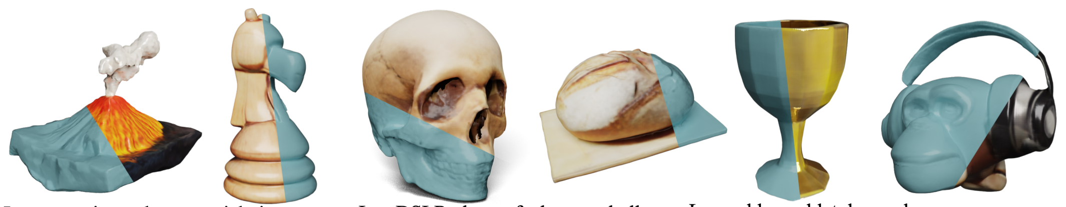
We compare MeshFormer with recent open-world feed-forward single/sparse-view to 3D methods, including One-2-3-45++ [31], TripoSR [67], CRM [77], LGM [64], InstantMesh [85], and MeshLRM [79]. Many of these methods have been released recently and should be considered concurrent methods. For MeshLRM [79], we contacted the authors for the results. For the other methods, we utilized their official implementations. Please refer to the supplementary for details.

Since input settings differ among the baselines, we evaluate all methods in a unified single-view to 3D setting. For the GSO dataset, we utilized the first thumbnail image as the single-view input. For the OmniObject3D dataset, we used a rendered image with a random pose as input. For One-2-3-45++ [31], InstantMesh [85], MeshLRM [79], and our MeshFormer, we first utilized Zero123++ [59] to convert the input single-view image into multi-view images before 3D reconstruction. Other baselines follow their original settings and take a single-view image directly as input. In addition to the RGB images, our MeshFormer also takes additional multi-view normal images as input, which are also predicted by Zero123++ [59]. **Note that when comparing with baseline methods, we never use ground truth normal images to ensure a fair comparison.**

In Fig. 3, we showcase qualitative examples. Our MeshFormer produces the most accurate meshes with fine-grained, sharp geometric details. In contrast, baseline methods produce inferior mesh quality. For example, TripoSR directly extracts meshes from the learned NeRF representation, resulting in significant artifacts. While InstantMesh and MeshLRM use mesh representation in their second stage, notable uneven artifacts are still observable upon a zoom-in inspection. Additionally, all baseline methods incorrectly close the surface of the copper bell. We also provide quantitative results in Tab. 1. Although our baselines include four methods released just one or two months before the time of submission, our MeshFormer significantly outperforms many of them and achieves the best performance on most metrics across two datasets. For the color LPIPS metric, our performance is very similar to MeshLRM's, despite a perceptual loss being their main training loss term. We also highlight that many of the baselines require over one hundred GPUs for training, whereas our model can be efficiently trained with just 8 GPUs. Please refer to Sec. 4.4 for analysis on training efficiency.

4.3 Application: Text to 3D

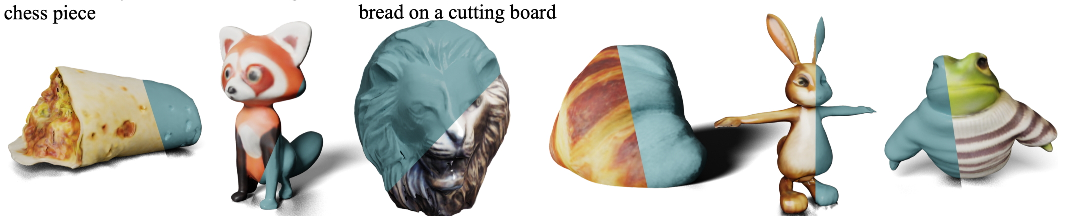
In addition to the single image to 3D, MeshFormer can also be integrated with 2D diffusion models to enable various 3D object generation tasks. For example, we follow the framework proposed by [37] to finetune Stable Diffusion [56] and build a text-to-multi-view model. By integrating this



L: an erupting volcano, aerial view
R: a beautifully carved wooden knight chess piece

L: a DSLR photo of a human skull
R: a freshly baked loaf of sourdough bread on a cutting board

L: a golden goblet, low poly
R: a chimpanzee wearing headphones



L: a pyramid shaped burrito with a slice cut out of it
R: a red panda

L: a metal sculpture of a lion's head, highly detailed
R: a DSLR photo of a delicious croissant

L: a rabbit, animated movie character, high detail 3d model
R: a DSLR photo of a frog wearing a sweater

Figure 4: **Application: Text to 3D.** Given a text prompt, a 2D diffusion model first predicts multi-view RGB and normal images, which are then fed to MeshFormer for 3D reconstruction. Please refer to the supplementary for comparisons with Instant3D [27].

Table 2: We compare methods using limited training resources. Evaluated on the GSO [11] dataset.

Method	Training Resources	F-Score ↑	CD ↓	PSNR-C ↑	LPIPS-C ↓	PSNR-N ↑	LPIPS-N ↓
MeshLRM [79]	8×H100 48h	0.925	0.0397	21.09	0.26	21.69	0.22
Ours	8×H100 48h	0.960	0.0317	21.41	0.20	23.01	0.15

model, along with the normal prediction from Zero123++ [59], with MeshFormer, we can enable the task of text to 3D. Figure 4 shows some interesting results, where we convert a single text prompt into a high-quality 3D mesh in just a few seconds. Please refer to the supplemental materials for a qualitative comparison with one of the state-of-the-art text-to-3D methods, Instant3D [27].

4.4 Analysis and Ablation Study

Explicit 3D structure vs. Triplane In Section 4.2, we demonstrated that MeshFormer outperforms baseline methods that primarily utilize the triplane representation. Here, we highlight two additional advantages of using the explicit 3D voxel structure: training efficiency and the avoidance of "triplane artifacts". Without leveraging explicit 3D structure, existing triplane-based large reconstruction models require extensive computing resources for training. For example, TripoSR requires 176 A100 GPUs for five days of training. InstantMesh relies on OpenLRM [15], which requires 128 A100 GPUs for three days of training. MeshLRM also utilizes similar resources during training. By utilizing explicit 3D structure and projective bias, our MeshFormer can be trained much more efficiently using only 8 GPUs. To better understand the gap, we trained both MeshLRM and our MeshFormer under very limited training resources, and the results are shown in Table 2. When using only 8 GPUs for two days, we found that MeshLRM failed to converge and experienced significant performance degradation compared to the results shown in Table 1, while our MeshFormer had already converged to a decent result, close to the fully-trained version, demonstrating superior training efficiency.

We observe that the triplane typically generates results with axis-aligned artifacts, as shown in Fig.3 (5th row, please zoom in). As demonstrated in the supplementary (Fig. 7), these artifacts also cause difficulties for MeshLRM [79] in capturing the words on objects. These limitations are likely caused by the limited number of triplane tokens (e.g., $32 \times 32 \times 3$), constrained by the global attention, which often leads to artifacts at the boundaries of the triplane patches. In contrast, MeshFormer leverages sparse voxels, supports a higher feature resolution of 256^3, and is free from such artifacts.

Normal Input and SDF supervision As shown in Table 3 (a), the performance significantly drops when multi-view input normal maps are removed, indicating that the geometric guidance and clues provided by normal images are crucial for facilitating network training, particularly for local geometric details. In (f), we replace ground truth normal maps with normal predictions by Zero123++ [59] and

Table 3: **Ablation Study on the GSO [11] dataset.** -C denotes color renderings, and -N denotes normal renderings. CD stands for Chamfer distance. By default, ground truth multi-view images are used to exclude the influence of errors from 2D diffusion models.

	Setting	PSNR-C ↑	LPIPS-C ↓	PSNR-N ↑	LPIPS-N ↓	F-Score ↑	CD ↓
a	w/o normal input	24.82	0.129	24.85	0.107	0.964	0.024
b	w/o SDF supervision	20.72	0.244	20.42	0.257	0.940	0.035
c	w/o transformer layer	26.63	0.101	29.80	0.036	0.992	0.013
d	w/o projection-aware cross-attention	25.48	0.155	29.01	0.045	0.991	0.013
e	w/o geometry enhancement	27.95	0.085	29.10	0.048	0.992	0.012
f	w/ pred normal	26.84	0.096	26.99	0.067	0.987	0.017
g	full	28.15	0.083	29.80	0.036	0.992	0.012

observe a notable performance gap compared to (g). This indicates that although predicted multi-view normal images can be beneficial, existing 2D diffusion models still have room for improvement in generating more accurate results. See supplementary for qualitative examples. As shown in (b), if we remove the SDF loss after the first epoch and train the network using only surface rendering losses, the geometry learning quickly deteriorates, resulting in poor geometry. This explains why existing methods [27, 79] typically employ complex multi-stage training and use volume rendering to learn a coarse NeRF in the initial stage. By leveraging explicit 3D SDF supervision as strong geometric regularization, we enable a unified single-stage training, using mesh as the only representation.

Projection-Aware Cross-Attention and Transformer Layers We propose to utilize projection-aware cross-attention to precisely aggregate multi-view projected 2D features for each 3D voxel. In conventional learning-based multi-view stereo (MVS) methods [3, 38], average or max pooling is typically employed for feature aggregation. In Table 3 (d), we replace the cross-attention with a simple average pooling and we observe a significant performance drop. This verifies that projection-aware cross-attention provides a more effective way for 3D-2D interaction while simple average pooling may fail to handle the occlusion and visibility issues. In the bottleneck of the UNet, we treat all 3D (sparse) voxels as a sequence of tokens and apply transformer layers to them. As shown in row (c), after removing these layers, we observe a performance drop in metrics related to texture quality. This indicates that texture learning requires more extensive priors and benefits more from the transformer layers.

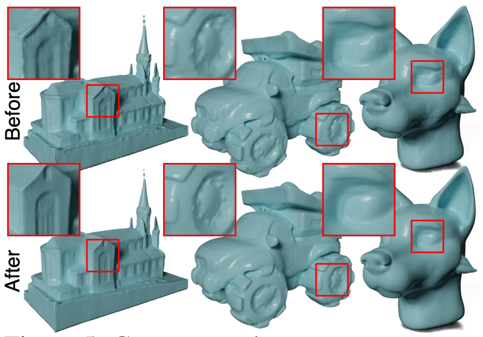


Figure 5: Geometry enhancement generates sharper details. Please zoom in to see the details.

Geometry Enhancement We propose to learn an additional normal map texture and apply a traditional algorithm as post-processing for geometry enhancement during inference. As shown in Figure 5, the geometry enhancement aligns the mesh geometry with the learned normal texture and generates fine-grained sharp details. In some cases (such as the wolf), the meshes output by the network are already good enough, and the difference caused by the enhancement tends to be subtle. Row (e) also quantitatively verifies the effectiveness of the module.

5 Conclusion and Limitations

We present MeshFormer, an open-world sparse-view reconstruction model that leverages explicit 3D native structure, supervision signals, and input guidance. MeshFormer can be conveniently trained in a unified single-stage manner and efficiently with just 8 GPUs. It generates high-quality meshes with fine-grained geometric details and outperforms baselines trained with over one hundred GPUs.

MeshFormer relies on 2D models to generate multi-view RGB and normal images from a single input image or text prompt. However, existing models still have limited capabilities to generate consistent multi-view images, which can cause a performance drop. Strategies to improve model robustness against such imperfect predictions are worth further exploration, and we leave this as future work.

References

[1] Eric R Chan, Koki Nagano, Matthew A Chan, Alexander W Bergman, Jeong Joon Park, Axel Levy, Miika Aittala, Shalini De Mello, Tero Karras, and Gordon Wetzstein. Genvs: Generative novel view synthesis with 3d-aware diffusion models, 2023.

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[3] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14124–14133, 2021.

[4] Guanying Chen, Kai Han, Boxin Shi, Yasuyuki Matsushita, and Kwan-Yee K. Wong. Sdps-net: Self-calibrating deep photometric stereo networks. In *CVPR*, 2019.

[5] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023.

[6] Zilong Chen, Feng Wang, Yikai Wang, and Huaping Liu. Text-to-3d using gaussian splatting, 2024.

[7] Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, Alexander G Schwing, and Liang-Yan Gui. Sdfusion: Multimodal 3d shape completion, reconstruction, and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4456–4465, 2023.

[8] 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, 2023.

[9] 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, 2022.

[10] Congyue Deng, Chiyu Jiang, Charles R Qi, Xinchen Yan, Yin Zhou, Leonidas Guibas, Dragomir Anguelov, et al. Nerdi: Single-view nerf synthesis with language-guided diffusion as general image priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20637–20647, 2023.

[11] 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.

[12] 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. *arXiv preprint arXiv:2403.12013*, 2024.

[13] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. *Advances In Neural Information Processing Systems*, 35:31841–31854, 2022.

[14] Ruiqi Gao*, Aleksander Holynski*, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul P. Srinivasan, Jonathan T. Barron, and Ben Poole*. Cat3d: Create anything in 3d with multi-view diffusion models. *arXiv*, 2024.

[15] Zexin He and Tengfei Wang. Openlrm: Open-source large reconstruction models. `https://github.com/3DTopia/OpenLRM`, 2023.

[16] 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.

[17] Hanzhe Hu, Zhizhuo Zhou, Varun Jampani, and Shubham Tulsiani. Mvd-fusion: Single-view 3d via depth-consistent multi-view generation. In *CVPR*, 2024.

[18] Ka-Hei Hui, Aditya Sanghi, Arianna Rampini, Kamal Rahimi Malekshan, Zhengzhe Liu, Hooman Shayani, and Chi-Wing Fu. Make-a-shape: a ten-million-scale 3d shape model. In *Forty-first International Conference on Machine Learning*, 2024.

[19] Mohammad Mahdi Johari, Yann Lepoittevin, and François Fleuret. Geonerf: Generalizing nerf with geometry priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18365–18375, 2022.

[20] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions, 2023.

[21] Animesh Karnewar, Andrea Vedaldi, David Novotny, and Niloy J Mitra. Holodiffusion: Training a 3d diffusion model using 2d images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18423–18433, 2023.

[22] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023.

[23] Xin Kong, Shikun Liu, Xiaoyang Lyu, Marwan Taher, Xiaojuan Qi, and Andrew J Davison. Eschernet: A generative model for scalable view synthesis. *arXiv preprint arXiv:2402.03908*, 2024.

[24] Jonáš Kulhánek, Erik Derner, Torsten Sattler, and Robert Babuška. Viewformer: Nerf-free neural rendering from few images using transformers. In *European Conference on Computer Vision*, pages 198–216. Springer, 2022.

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

[26] Han-Hung Lee and Angel X. Chang. Understanding pure clip guidance for voxel grid nerf models, 2022.

[27] 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.

[28] Weiyu Li, Rui Chen, Xuelin Chen, and Ping Tan. Sweetdreamer: Aligning geometric priors in 2d diffusion for consistent text-to-3d. *arxiv:2310.02596*, 2023.

[29] Yuhan Li, Yishun Dou, Xuanhong Chen, Bingbing Ni, Yilin Sun, Yutian Liu, and Fuzhen Wang. Generalized deep 3d shape prior via part-discretized diffusion process. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16784–16794, 2023.

[30] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[31] Minghua Liu, Ruoxi Shi, Linghao Chen, Zhuoyang Zhang, Chao Xu, Xinyue Wei, Hansheng Chen, Chong Zeng, Jiayuan Gu, and Hao Su. One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion. *arXiv preprint arXiv:2311.07885*, 2023.

[32] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *Advances in Neural Information Processing Systems*, 36, 2024.

[33] 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.

[34] 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. *arXiv preprint arXiv:2309.03453*, 2023.

[35] Yuan Liu, Sida Peng, Lingjie Liu, Qianqian Wang, Peng Wang, Christian Theobalt, Xiaowei Zhou, and Wenping Wang. Neural rays for occlusion-aware image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7824–7833, 2022.

[36] Yuxin Liu, Minshan Xie, Hanyuan Liu, and Tien-Tsin Wong. Text-guided texturing by synchronized multi-view diffusion. *arXiv preprint arXiv:2311.12891*, 2023.

[37] 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. *arXiv preprint arXiv:2310.15008*, 2023.

[38] Xiaoxiao Long, Cheng Lin, Peng Wang, Taku Komura, and Wenping Wang. Sparseneus: Fast generalizable neural surface reconstruction from sparse views. In *European Conference on Computer Vision*, pages 210–227. Springer, 2022.

[39] William E. Lorensen and Harvey E. Cline. *Marching cubes: a high resolution 3D surface construction algorithm*, page 347–353. Association for Computing Machinery, New York, NY, USA, 1998.

[40] J. Lorraine, K. Xie, X. Zeng, C. Lin, T. Takikawa, N. Sharp, T. Lin, M. Liu, S. Fidler, and J. Lucas. Att3d: Amortized text-to-3d object synthesis. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17900–17910, Los Alamitos, CA, USA, oct 2023. IEEE Computer Society.

[41] 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, 2024.

[42] 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. In *ECCV*, 2020.

[43] 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.

[44] Diego Nehab, Szymon Rusinkiewicz, James Davis, and Ravi Ramamoorthi. Efficiently combining positions and normals for precise 3d geometry. *ACM Trans. Graph.*, 24(3):536–543, jul 2005.

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

[46] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023.

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

[48] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, and Bernard Ghanem. Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[49] Lingteng Qiu, Guanying Chen, Xiaodong Gu, Qi Zuo, Mutian Xu, Yushuang Wu, Weihao Yuan, Zilong Dong, Liefeng Bo, and Xiaoguang Han. Richdreamer: A generalizable normal-depth diffusion model for detail richness in text-to-3d, 2023.

[50] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[51] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021.

[52] Konstantinos Rematas, Ricardo Martin-Brualla, and Vittorio Ferrari. Sharf: Shape-conditioned radiance fields from a single view. *arXiv preprint arXiv:2102.08860*, 2021.

[53] Xuanchi Ren, Jiahui Huang, Xiaohui Zeng, Ken Museth, Sanja Fidler, and Francis Williams. Xcube: Large-scale 3d generative modeling using sparse voxel hierarchies. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4209–4219, 2024.

[54] Yufan Ren, Tong Zhang, Marc Pollefeys, Sabine Süsstrunk, and Fangjinhua Wang. Volrecon: Volume rendering of signed ray distance functions for generalizable multi-view reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16685–16695, 2023.

[55] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685, Los Alamitos, CA, USA, jun 2022. IEEE Computer Society.

[56] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022.

[57] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[58] Junyoung Seo, Wooseok Jang, Min-Seop Kwak, Jaehoon Ko, Hyeonsu Kim, Junho Kim, Jin-Hwa Kim, Jiyoung Lee, and Seungryong Kim. Let 2d diffusion model know 3d-consistency for robust text-to-3d generation. *arXiv preprint arXiv:2303.07937*, 2023.

[59] 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.

[60] Yichun Shi, Peng Wang, Jianglong Ye, Long Mai, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv:2308.16512*, 2023.

[61] Robust Multiview Stereopsis. Accurate, dense, and robust multiview stereopsis. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 32(8), 2010.

[62] Jingxiang Sun, Bo Zhang, Ruizhi Shao, Lizhen Wang, Wen Liu, Zhenda Xie, and Yebin Liu. Dreamcraft3d: Hierarchical 3d generation with bootstrapped diffusion prior, 2023.

[63] Haotian Tang, Shang Yang, Zhijian Liu, Ke Hong, Zhongming Yu, Xiuyu Li, Guohao Dai, Yu Wang, and Song Han. Torchsparse++: Efficient training and inference framework for sparse convolution on gpus. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2023.

[64] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. *arXiv preprint arXiv:2402.05054*, 2024.

[65] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023.

[66] Ayush Tewari, Tianwei Yin, George Cazenavette, Semon Rezchikov, Joshua B Tenenbaum, Frédo Durand, William T Freeman, and Vincent Sitzmann. Diffusion with forward models: Solving stochastic inverse problems without direct supervision. *arXiv preprint arXiv:2306.11719*, 2023.

[67] 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, 2024.

[68] Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d representation and rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15182–15192, 2021.

[69] Vikram Voleti, Chun-Han Yao, Mark Boss, Adam Letts, David Pankratz, Dmitrii Tochilkin, Christian Laforte, Robin Rombach, and Varun Jampani. SV3D: Novel multi-view synthesis and 3D generation from a single image using latent video diffusion. *arXiv*, 2024.

[70] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. *arXiv preprint arXiv:2212.00774*, 2022.

[71] Peihao Wang, Xuxi Chen, Tianlong Chen, Subhashini Venugopalan, Zhangyang Wang, et al. Is attention all nerf needs? *arXiv preprint arXiv:2207.13298*, 2022.

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

[73] 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.

[74] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2021.

[75] Zhen Wang, Qiangeng Xu, Feitong Tan, Menglei Chai, Shichen Liu, Rohit Pandey, Sean Fanello, Achuta Kadambi, and Yinda Zhang. Mvdd: Multi-view depth diffusion models, 2023.

[76] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *arXiv preprint arXiv:2305.16213*, 2023.

[77] 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. *arXiv preprint arXiv:2403.05034*, 2024.

[78] Xinyue Wei, Fanbo Xiang, Sai Bi, Anpei Chen, Kalyan Sunkavalli, Zexiang Xu, and Hao Su. NeuManifold: Neural Watertight Manifold Reconstruction with Efficient and High-Quality Rendering Support. *arXiv preprint*, 2023.

[79] 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.

[80] Haohan Weng, Tianyu Yang, Jianan Wang, Yu Li, Tong Zhang, C. L. Philip Chen, and Lei Zhang. Consistent123: Improve consistency for one image to 3d object synthesis, 2023.

[81] Sangmin Woo, Byeongjun Park, Hyojun Go, Jin-Young Kim, and Changick Kim. Harmonyview: Harmonizing consistency and diversity in one-image-to-3d, 2023.

[82] Robert J. Woodham. *Photometric method for determining surface orientation from multiple images*, page 513–531. MIT Press, Cambridge, MA, USA, 1989.

[83] Tong Wu, Jiarui Zhang, Xiao Fu, Yuxin Wang, Jiawei Ren, Liang Pan, Wayne Wu, Lei Yang, Jiaqi Wang, Chen Qian, et al. Omniobject3d: Large-vocabulary 3d object dataset for realistic perception, reconstruction and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 803–814, 2023.

[84] Kevin Xie, Jonathan Lorraine, Tianshi Cao, Jun Gao, James Lucas, Antonio Torralba, Sanja Fidler, and Xiaohui Zeng. Latte3d: Large-scale amortized text-to-enhanced3d synthesis. *arXiv preprint arXiv:2403.15385*, 2024.

[85] 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.

[86] Yinghao Xu, Zifan Shi, Wang Yifan, Sida Peng, Ceyuan Yang, Yujun Shen, and Wetzstein Gordon. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation. *arxiv: 2403.14621*, 2024.

[87] Yinghao Xu, Hao Tan, Fujun Luan, Sai Bi, Peng Wang, Jiahao Li, Zifan Shi, Kalyan Sunkavalli, Gordon Wetzstein, Zexiang Xu, et al. Dmv3d: Denoising multi-view diffusion using 3d large reconstruction model. *arXiv preprint arXiv:2311.09217*, 2023.

[88] Hao Yang, Lanqing Hong, Aoxue Li, Tianyang Hu, Zhenguo Li, Gim Hee Lee, and Liwei Wang. Contranerf: Generalizable neural radiance fields for synthetic-to-real novel view synthesis via contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16508–16517, 2023.

[89] Jianglong Ye, Peng Wang, Kejie Li, Yichun Shi, and Heng Wang. Consistent-1-to-3: Consistent image to 3d view synthesis via geometry-aware diffusion models. *arXiv preprint arXiv:2310.03020*, 2023.

[90] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021.

[91] Wangbo Yu, Li Yuan, Yan-Pei Cao, Xiangjun Gao, Xiaoyu Li, Wenbo Hu, Long Quan, Ying Shan, and Yonghong Tian. Hifi-123: Towards high-fidelity one image to 3d content generation, 2024.

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

[93] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.

[94] Xin-Yang Zheng, Hao Pan, Yu-Xiao Guo, Xin Tong, and Yang Liu. Mvd^2: Efficient multiview 3d reconstruction for multiview diffusion. In *SIGGRAPH*, 2024.

[95] Xin-Yang Zheng, Hao Pan, Peng-Shuai Wang, Xin Tong, Yang Liu, and Heung-Yeung Shum. Locally attentional sdf diffusion for controllable 3d shape generation. *arXiv preprint arXiv:2305.04461*, 2023.

[96] Zi-Xin Zou, Zhipeng Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Yan-Pei Cao, and Song-Hai Zhang. Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers, 2023.

A Appendix

A.1 Comparison with Instant3D

In Figure 6, we showcase the comparison with Instant3D [27] on the text-to-3D task. The results are obtained from the paper authors. While Instant3D [27] also generates 3D shapes that match the input text prompt, our method generates results with superior mesh quality and fine-grained, sharp geometric details.

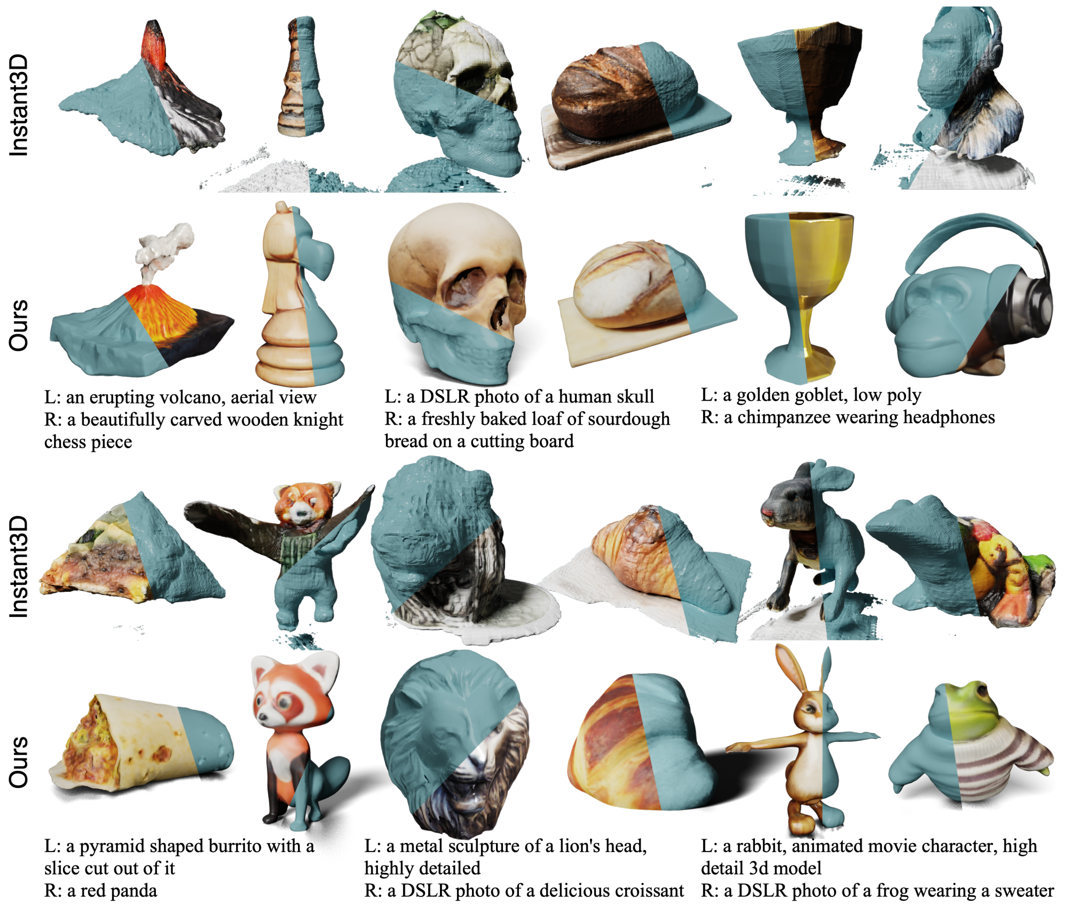

Instant3D
Ours
L: an erupting volcano, aerial view
R: a beautifully carved wooden knight chess piece
L: a DSLR photo of a human skull
R: a freshly baked loaf of sourdough bread on a cutting board
L: a golden goblet, low poly
R: a chimpanzee wearing headphones
Instant3D
Ours
L: a pyramid shaped burrito with a slice cut out of it
R: a red panda
L: a metal sculpture of a lion's head, highly detailed
R: a DSLR photo of a delicious croissant
L: a rabbit, animated movie character, high detail 3d model
R: a DSLR photo of a frog wearing a sweater

Figure 6: **Application: Text-to-3D**. Comparison with Instant3D [27].

A.2 Triplane Artifacts

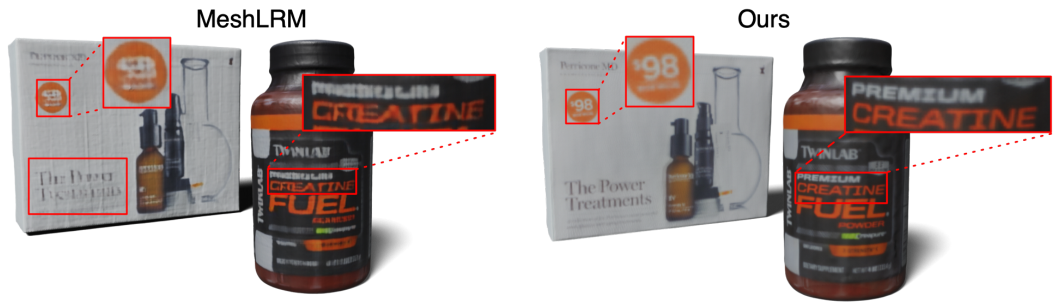


Figure 7: The triplane-based method MeshLRM [79] has difficulty capturing words on objects, even when ground truth multi-view RGB images are used as input.

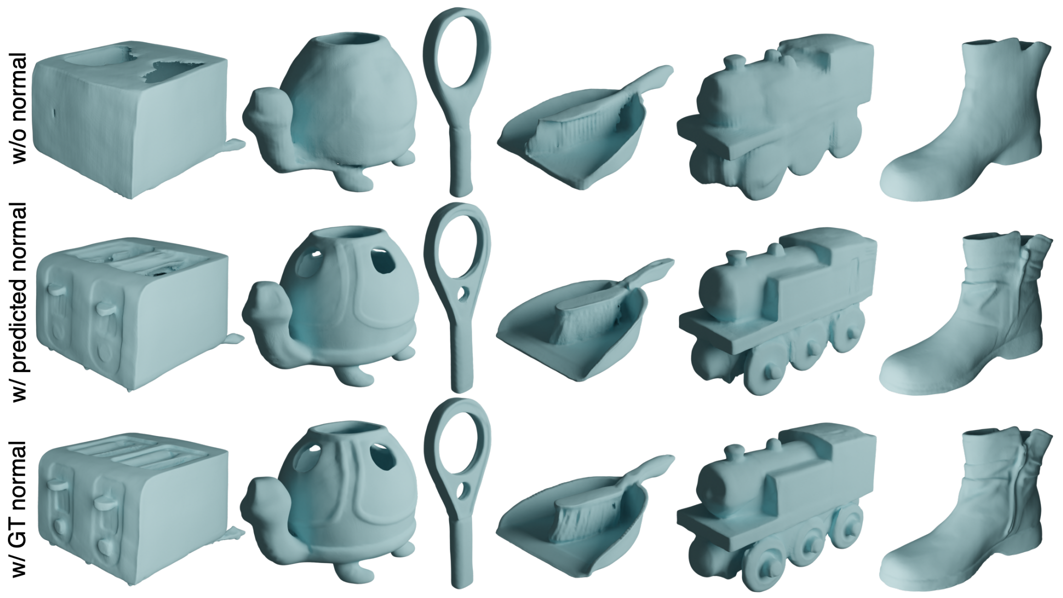


Figure 8: Ablation study on input normal maps. Evaluated on the GSO dataset [11]. "w/o normal" indicates that the model is trained with multi-view RGB images only. "w/ predicted normal" indicates that the model is trained with ground truth normal maps but evaluated with predicted normals by Zero123++ [59]. "w/ GT normal" indicates that the model is trained and tested with ground truth normals.

Table 4: Normal consistency (angle error) between the mesh geometry (mesh vertex normals) and the predicted normal maps, both before and after the geometry enhancement post-processing. The ratio of mesh vertices below a specific error threshold is shown. Evaluated on the GSO dataset.

angle error threshold	before	after
$< 1°$	8.83%	16.27%
$< 2°$	26.39%	40.83%
$< 5°$	60.55%	73.19%
$< 10°$	78.79%	86.43%
$< 15°$	86.46%	91.29%

As shown in Fig.7, MeshLRM [79] has difficulty capturing words on objects, even when ground truth multi-view RGB images are used as input. We speculate that this is due to the limited number of triplane patches (e.g., $32 \times 32 \times 3$) restricted by global attention. In contrast, our method leverages sparse voxels and supports a much higher feature resolution of 256^3, making it free from such issues.

A.3 Ablation Study: Input Normal Maps

In Figure 8, we qualitatively demonstrate the effect of input normal maps. When the model is trained without multi-view normal maps, we find that the generated model can only capture the global 3D shape but fails to generate fine-grained geometric details. However, when the model is given predicted normal maps, the performance is significantly better, although there are still some small gaps when compared to the results of ground truth normals (see the bread hole of the toaster and the wheel of the tram). This indicates errors or inconsistencies from the 2D normal prediction models.

A.4 Ablation Study: Geometry Enhancement

We propose asking the network to predict an additional normal texture, which can be used for further geometric enhancement by applying a traditional algorithm as post-processing. The geometric enhancement aims to align the mesh geometry with the predicted normal map by adjusting the vertex locations. However, the traditional algorithm we used cannot guarantee that the mesh normals will

Table 5: **Analysis of our mesh generation quality over training time.** Evaluated on the GSO [11] dataset.

Training Time	PSNR-C ↑	LPIPS-C ↓	PSNR-N ↑	LPIPS-N ↓	CD ↓	F-Score ↑
8×H100 12h	21.28	0.2135	22.89	0.1536	0.0330	0.960
8×H100 24h	21.32	0.2076	22.96	0.1516	0.0320	0.960
8×H100 48h	21.41	0.2033	23.01	0.1484	0.0317	0.960
8×H100 120h	21.44	0.2029	23.04	0.1480	0.0314	0.961
8×H100 168h	21.47	0.2010	23.09	0.1466	0.0313	0.963

be fully aligned with the predicted normal maps after processing. This limitation arises because the algorithm operates in local space and avoids large vertex displacements. Moreover, the predicted normal maps may contain errors or inconsistencies, such as conflicting neighboring normals. The adopted algorithm is an iterative numerical optimization method and does not compute an analytic solution.

However, we have quantitatively verified that the post-processing module can significantly improve normal consistency with the predicted normal map. For example, before post-processing, only 26.4% of mesh vertices had a normal angle error of less than 2 degrees. After post-processing, this number increased to 40.8%. For a 10-degree threshold, the ratio increases from 78.8% to 86.4%. For more details, please refer to Table 4.

A.5 Ablation Study: Training time

Our MeshFormer can be trained efficiently using only 8 GPUs, typically converging in approximately two days. Table 5 presents a quantitative analysis of our mesh generation quality over the training period. We observe that performance improves rapidly and nearly converges, with only marginal changes occurring after the two-day training period.

A.6 Training Details and Evaluation Metrics

Training Details: We trained the model using a subset of 395k 3D shapes filtered from the Objaverse [9] dataset. These objects have a distributable Creative Commons license and were obtained by the Objaverse team using Sketchfab's public API. For each filtered 3D shape, we randomly rotated the mesh and generated 10 data samples. For each data sample, we compute a 512^3 ground truth SDF volume using a CUDA-based program and render multi-view RGB and normal images using BlenderProc. In our experiments, the resolutions of the occupancy volume and sparse feature volume are 64 and 256, respectively. The resolution of the predicted and ground truth SDF volumes is 512. The model is trained with the Adam optimizer and a cosine learning rate scheduler. The loss weights $\lambda_1, \cdots, \lambda_6$ are set to 80, 2, 16, 2, 8, and 8, respectively.

All data preparation, including image rendering and SDF computation, is performed using an internal cluster. This process can be completed using 4000 CPU cores in roughly one week. The generated data takes up approximately 30TB. All model training tasks are conducted in public cloud clusters. Our main model is trained using 8 H100 GPUs for one week. All experiments listed in the paper can be completed in 15 days using 32 H100 GPUs (running multiple parallel experiments), excluding the preliminary exploration experiments.

Architecture Details: For VoxelFormer, the UNet consists of four levels with resolutions of 64^3, 32^3, 16^3 and 16^3. Each level includes a ResNet module, a projection-aware cross-attention module, and a downsampling module, with channel sizes of 64, 128, 256, and 512. We added 6 transformer layers at the bottleneck of the UNet, with each 3D voxel treated as a token, and token channels set to 512.

For SparseVoxelFormer, the sparse UNet consists of six levels with resolutions of 256^3, 128^3, 64^3, 32^3, 16^3, and 16^3. Each level includes a sparse ResNet module, a projection-aware cross-attention module, and a downsampling module, with channel sizes of 16, 32, 64, 128, 512, and 2,048. We added 16 transformer layers at the bottleneck of the UNet, with each 3D sparse voxel treated as a token, and token channels set to 1,024. The feature dimension of the output sparse feature volume (before the MLP) is 32.

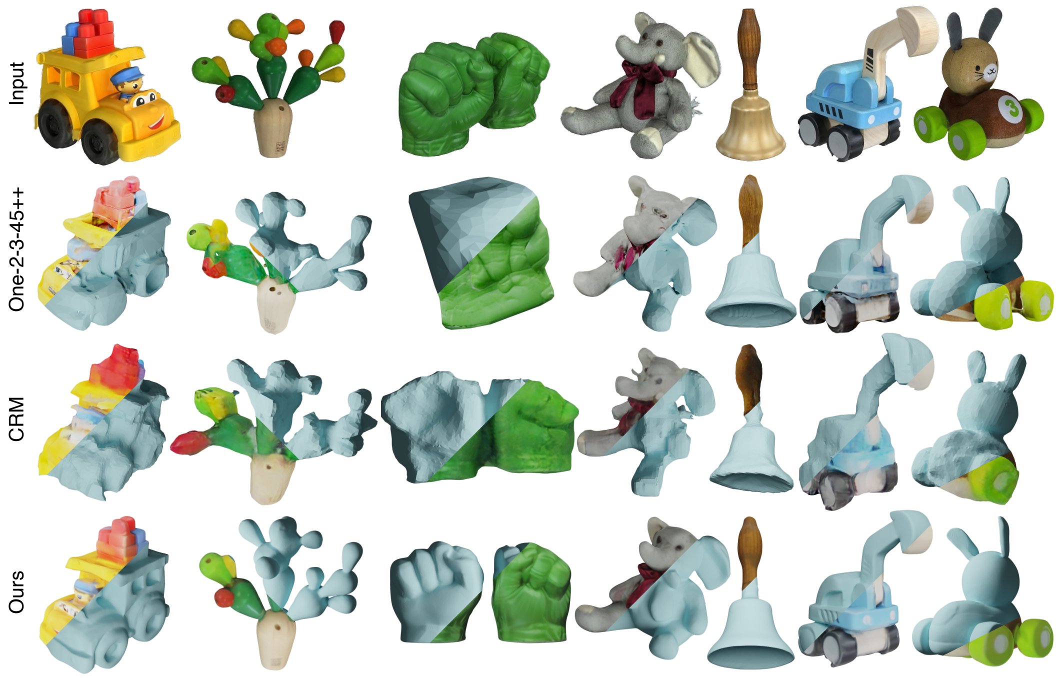


Figure 9: **Qualitative Results of One-2-3-45++ [31] and CRM [77] on Single Image to 3D.** Both the textured and textureless mesh renderings are shown.

For both of them, a skip connection is added to the UNet.

Evaluation Metrics: To account for the scale and pose ambiguity of the generated mesh from different baselines, we align the predicted mesh with the ground truth mesh prior to the evaluation metric calculation. This alignment process involves uniformly sampling rotations and scales for initialization and subsequently refining the alignment using the Iterative Closest Point (ICP) algorithm. We select the alignment that yields the highest inlier ratio. Both the ground truth and predicted meshes are then scaled to fit within a unit bounding box.

For 3D metrics, we sample 100,000 points on both the ground truth mesh and the predicted mesh and compute the F-score and Chamfer distance, setting the F-score threshold at 0.05. To evaluate texture quality, we compute the PSNR and LPIPS between images rendered from the reconstructed mesh and those of the ground truth. Following InstantMesh [85], we sample 24 camera poses, encompassing a full 360-degree view around the object, and utilize BlenderProc for rendering RGB and normal images with a resolution of 320×320. Since we use the VGG model for LPIPS loss calculation during training, we employ the Alex model for LPIPS loss calculation during evaluation.

A.7 Training Details of MeshLRM

All results of MeshLRM, except those in Table 2, were reproduced by the MeshLRM authors at Hillbot following the original settings as described in the paper. For the results in Table 2, we trained the model using the same training data as our method on $8\times$H100 GPUs for 48 hours. We maintained the same batch size as reported in the paper and proportionally scaled down the original training time for each stage of MeshLRM based on a total training time of 48 hours. This included 5.8 seconds per iteration for 20,000 iterations in the 256-resolution pre-training, 12 seconds per iteration for 4,000 iterations in the 512-resolution fine-tuning, and 4.7 seconds per iteration for 4,000 iterations in mesh refinement.

A.8 Qualitative Examples of One-2-3-45++ and CRM

Figure 9 shows qualitative results of One-2-3-45++ [31] and CRM [77] on single image to 3D and our method produces better results.

A.9 Broader Impact

We introduce an efficient approach for training open-world sparse-view reconstruction models, which has the potential to significantly reduce energy consumption and carbon emissions, as baseline models typically require much more computing resources for training. Previously, the creation of 3D assets was reserved for specialized artists who spent hours or even days producing a single 3D model. Our proposed technique allows even novice individuals without specialized 3D modeling knowledge to create high-quality 3D assets in seconds. This democratization of 3D modeling has unleashed unprecedented creative potential and operational efficiency across various sectors.

However, like other generative AI models, it also carries the risk of misuse, such as spreading misinformation and creating pornography models. Therefore, it is crucial to implement strict ethical guidelines to mitigate these risks.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We believe so. We provide an overview of our contributions at the end of the introduction (see section 1).

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We explicitly discuss limitations of our work in section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: We do not present theoretical results in this paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide extensive details about the implementation of our method, baselines, and evaluation in subsection 4.1, subsection A.6, and subsection A.7 to ensure reproducibility.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The code and model release have not yet passed our internal inspection, and the model also needs a safety evaluation.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We have included all details in subsection 4.1 and subsection A.6.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Error bars are not reported because it would be too computationally expensive to train the models multiple times. Additionally, the computation of evaluation metrics does not involve significant randomness or variation.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We include details about computing resource in subsection 4.1, subsection A.6 and subsection A.7.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors have read the NeurIPS Code of Ethics and ensured that our research conforms to it.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the broader impacts of our method in subsection A.9

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [No]

    Justification: As discussed in subsection A.9, our model carries the risk of misuse, including the spread of misinformation and the creation of pornographic content. We will conduct an internal safety inspection and implement necessary safeguards before releasing the model.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited all the original papers that produced the code and dataset used in our paper, and have properly respected all licenses and terms of use.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release any new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.