MeshLRM: Large Reconstruction Model for High-Quality Meshes

Xinyue Wei[1*], Kai Zhang[2*], Sai Bi[2], Hao Tan[2], Fujun Luan[2], Valentin Deschaintre[2], Kalyan Sunkavalli[2], Hao Su[1], Zexiang Xu[2]

[1] UC San Diego
[2] Adobe Research
https://sarahweiii.github.io/meshlrm/

Fig. 1: MeshLRM is an LRM-based content creation framework designed to produce high-quality 3D assets. The intricate 3D meshes and textures featured in this scene are all reconstructed from our method within 1 second per asset, which incorporates existing image/text-to-3D generation and our end-to-end sparse-view mesh reconstruction.

Abstract. We propose MeshLRM, a novel LRM-based approach that can reconstruct a high-quality mesh from merely four input images in less than one second. Different from previous large reconstruction models (LRMs) that focus on NeRF-based reconstruction, MeshLRM incorporates differentiable mesh extraction and rendering within the LRM framework. This allows for end-to-end mesh reconstruction by fine-tuning a pre-trained NeRF LRM with mesh rendering. Moreover, we improve the LRM architecture by simplifying several complex designs in previous LRMs. MeshLRM's NeRF initialization is sequentially trained with low- and high-resolution images; this new LRM training strategy enables significantly faster convergence and thereby leads to better quality

* Equal contribution.
Research done when X. Wei was an intern at Adobe Research.

with less compute. Our approach achieves state-of-the-art mesh reconstruction from sparse-view inputs and also allows for many downstream applications, including text-to-3D and single-image-to-3D generation.

Keywords: Sparse-view reconstruction · High-quality mesh · Large Reconstruction Models

1 Introduction

High-quality 3D mesh models are the core of 3D vision and graphics applications and 3D editing, rendering, and simulation tools are specifically optimized for them. 3D mesh assets are usually created manually by expert artists or alternatively reconstructed from multi-view 2D images. Traditionally this has been done with complex photogrammetry systems [19, 49, 50, 55], though recent neural approaches, such as NeRF [41], offer a simpler end-to-end pipeline through per-scene optimization. However, these neural methods typically produce volumetric representations [6, 33, 41, 42, 63, 78]; converting them into meshes requires additional optimization in post [47, 59, 70, 77]. Furthermore, both traditional and neural approaches require a large number of input images and long processing time (up to hours), limiting their applicability in time-sensitive design workflows.

Our goal is efficient and accurate 3D asset creation via few-shot mesh reconstruction with direct feed-forward network inference and no per-scene optimization. We base our approach on recent large reconstruction models (LRMs) [23, 30, 64, 73] for 3D reconstruction and generation. Existing LRMs use triplane NeRFs [5] as the 3D representation for high rendering quality. While these NeRFs can be converted into meshes via a Marching Cube (MC) [39] post-processing, this leads to a significant drop in rendering quality and geometric accuracy.

We address this with MeshLRM, a novel transformer-based large reconstruction model, designed to directly output high-fidelity 3D meshes from sparse-view inputs. Specifically, MeshLRM incorporates differentiable surface extraction and rendering into a NeRF-based LRM. We apply a recent Differentiable Marching Cube (DiffMC) [70] technique to extract an iso-surface from the triplane NeRF's density field and render the extracted mesh using a differentiable rasterizer [29]. This enables end-to-end training of MeshLRM using mesh rendering losses, to optimize it to produce high-quality meshes that render realistic images.

We train MeshLRM by initializing the model using volumetric NeRF rendering; since our meshing components do not introduce any new parameters they can start from these pre-trained weights. However, we find that training a mesh-based LRM with differentiable MC is still highly challenging. The primary issue lies in the (spatially) sparse gradients from the DiffMC operation, that only affect surface voxels and leave the vast empty space untouched. This leads to poor local minima for model optimization and manifests as floaters in the reconstructions. We address this with a novel ray opacity loss that ensures that empty space along all pixel rays maintains near-zero density, effectively stabilizing the training and guiding the model to learn accurate floater-free surface

geometry. Our end-to-end training approach reconstructs high-quality meshes with rendering quality that surpasses NeRF volumetric rendering (see Tab. 5).

Our DiffMC-based meshing technique is general and can potentially be applied to any NeRF-based LRMs for mesh reconstruction. In this work, we present a simple and efficient LRM architecture comprising of a large transformer model that directly processes concatenated multi-view image tokens and triplane tokens with purely self-attention layers to regress final triplane features for NeRF and mesh reconstruction. In particular, we simplify many complex design choices used in previous LRMs [30, 64, 73], including the removal of pre-trained DINO modules in image tokenization and replacing the large triplane decoder MLP with small two-layer ones. We train MeshLRM on the Objaverse dataset [14] with a novel low-resolution pre-training and high-resolution fine-tuning strategy. These design choices lead to a state-of-the-art LRM model with faster training and inference and higher reconstruction quality.

In summary, our key contributions are:

- A novel LRM-based framework that integrates differentiable mesh extraction and rendering for end-to-end few-shot mesh reconstruction.
- A novel ray opacity loss for stabilizing DiffMC-based generalizable training.
- An efficient LRM architecture and training strategies to enable fast and high-quality reconstruction.

We benchmark MeshLRM for 3D reconstruction (on synthetic and real datasets [17,32]) and 3D generation (in combination with other multi-view generation methods). **Fig. 1 showcases high-quality mesh outputs from MeshLRM all reconstructed in less than one second.**

2 Related Work

Mesh Reconstruction. Despite the existence of various 3D representations, meshes remain the most widely used format in industrial 3D engines. Reconstructing high-quality meshes from multi-view images is a long-standing challenge in computer vision and graphics. Classically, this is addressed via a complex multi-stage photogrammetry pipeline, integrating techniques such as structure from motion (SfM) [1, 49, 55], multi-view stereo (MVS) [19, 50], and mesh surface extraction [27, 39]. Recently, deep learning-based methods have also been proposed to address these problems for higher efficiency and accuracy [11,24,25,58,62,75]. However, the photogrammety pipeline is not only costly, requiring dense input images and long processing time, but also often suffers from low-quality mesh rendering. Our approach enables sparse-view mesh reconstruction through direct feed-forward network inference in an end-to-end manner. We leverage neural volumetric representations and differentiable rendering, training our model to output high-quality meshes that can render realistic images.

Neural Reconstruction. Neural rendering techniques have recently gained significant attention for their ability to produce high-quality 3D reconstructions for realistic rendering [28, 37, 41, 80]. Most recent methods are based on NeRF [41] and reconstruct scenes as various volumetric radiance fields [6, 18, 41, 42, 72] with

per-scene rendering optimization. Though radiance fields can be converted into meshes with Marching Cubes [39], the quality of the resulting mesh is not guaranteed. Previous methods have aimed to transform radiance fields into implicit surface representations (SDFs) [43, 63, 76], enhancing mesh quality from Marching Cubes. In addition, some methods [47, 59, 70, 77] attempt to directly extract meshes from radiance fields using differentiable surface rendering. However, these methods all require dense input images and long per-scene optimization time, which our approach avoids.

On the other hand, generalizable neural reconstruction has been explored, often leveraging MVS-based cost volumes [7,26,38] or directly aggregating multi-view features along pixels' projective epipolar lines [57, 65, 79]. While enabling fast and few-shot reconstruction, these methods can only handle small baselines, still requiring dense images with local reconstruction in overlapping windows to model a complete object. More recently, transformer-based large reconstruction models (LRMs) [23, 30, 64, 73] have been proposed and achieved 3D NeRF reconstruction from highly sparse views. In this work, we propose a novel efficient LRM, incorporating DiffMC techniques [70], to enable high-quality sparse-view mesh reconstruction via direct feed-forward inference.

3D Generation. Generative models have seen rapid progress with GANs [21] and, more recently, Diffusion Models [56] for image and video generation. In the context of 3D generation, many approaches utilize 3D or multi-view data to train 3D generative models with GANs [5, 20, 22, 71] or diffusion models [2, 8, 36, 67, 73], which are promising but limited by the scale and diversity of 3D data. DreamFusion [45] proposed to leverage the gradient of a pre-trained 2D diffusion model to optimize a NeRF for text-to-3D generation with a score distillation sampling (SDS) loss, achieving diverse generation results beyond common 3D data distributions. This approach inspired many follow-up approaches, to make the optimization faster or improve the results quality [10,31,40,60,68]. However, these SDS-based methods rely on NeRF-like per-scene optimization, which is highly time-consuming, often taking hours. More recently, a few attempts have been made to achieve fast 3D generation by utilizing pre-trained 2D diffusion models to generate multi-view images and then perform 3D reconstruction [30, 34, 35]. In particular, Instant3D [30] leverages an LRM to reconstruct a triplane NeRF from four sparse generated images. In this work, we focus on the task of sparse-view reconstruction and propose to improve and re-target a NeRF LRM to directly generate meshes with higher quality. Our model can be naturally coupled with a multi-view generator, such as the ones in [30, 53], to enable high-quality text-to-3D and single-image-to-3D generation (see Fig. 5 and Fig. 6).

3 Method

In this section, we present our MeshLRM that reconstructs high-quality meshes within 1 second. We start with describing our backbone transformer architecture (Sec. 3.1), which simplifies and improves previous LRMs. We then introduce our two-stage framework for training the model: we first (Sec. 3.2) train the model

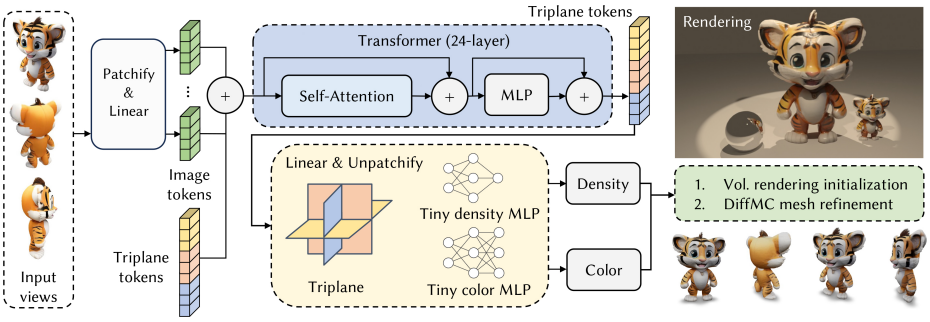


Fig. 2: The model architecture of MeshLRM. The images are first patchified to tokens. The transformer takes the concatenated image and triplane tokens as input. The output triplane tokens are upsampled with the unpatchifying operator while the output image tokens are dropped (not drawn in Fig.). With two tiny MLPs for density and color decoding, this model supports both the volumetric rendering (Sec. 3.2) and the DiffMC fine-tuning (Sec. 3.3). We render our final mesh output on the rightmost of the figure.

to predict NeRF from sparse input images by supervising volume renderings at novel views, followed by refining mesh surface extractions (Sec. 3.3) by performing differentiable marching cubes on the predicted density field and minimizing a surface rendering loss with differentiable rasterization.

3.1 Model Architecture

As shown in Fig. 2, we propose a simple transformer-based architecture for MeshLRM, mainly consisting of a sequence of self-attention-based transformer blocks over concatenated image tokens and triplane tokens, similar to PF-LRM [64]. In contrast to PF-LRM and other LRMs [23,30,73], we simplify the designs for both image tokenization and triplane NeRF decoding, leading to fast training and inference. We next describe the model in detail.

Input posed image tokenization. MeshLRM adopts a simple tokenizer for posed images, inspired by ViT [16]. We convert the camera parameters for each image into Plücker ray coordinates [44] and concatenate them with the RGB pixels (3-channel) to form a 9-channel feature map. Then we split the feature map into non-overlapping patches, and linearly transform them as transformer's input. With this process, the model does not need additional positional embedding as in ViT, since Plücker coordinates contain spatial information.

Note that our image tokenizer is much simpler than previous LRMs that use a pre-trained DINO ViT [4] for image encoding. We empirically find that the pre-trained DINO ViT is unnecessary, possibly because DINO is mainly trained for intra-view semantic reasoning, while 3D reconstruction mainly requires inter-view low-level correspondences. By dropping this per-view DINO encoding, the model also enables a better flow between raw pixels and 3D-related information.

Transformer. We concatenate multi-view image tokens and learnable triplane (positional) embeddings, and feed them into a sequence of transformer blocks [61],

where each block is comprised of self-attention and MLP layers. We add layer normalization [3] before both layers (i.e., Pre-LN architecture), and use residual connections. This deep transformer network enables comprehensive information exchange among all the tokens, and effectively models intra-view, inter-view, and cross-modal relationships. The output triplane tokens, contextualized by all input views, are then decoded into the renderable triplane NeRF. The output image tokens are dropped. Lastly, each triplane token is unprojected with a linear layer and further unpatchified to 8×8 triplane features via reshaping. [3] All the predicted triplane features are then assembled into the final triplane NeRF.

Tiny density and color MLPs. Previous LRMs [23, 30, 64, 73] all use a heavy shared MLP (e.g., 9 hidden layers with a hidden dimension of 64) to decode densities and colors from triplane features. This design leads to slow rendering in training. This is a bottleneck for MeshLRM since we need to compute a dense density grid, using the MLP and triplane features, for extracting the mesh with DiffMC. Therefore, we opt to use tiny MLPs with a narrower hidden dimension of 32 and fewer layers. In particular, we use an MLP with one hidden layer for density decoding and another MLP with two hidden layers for color decoding. Compared to a large MLP, our tiny MLPs can lead to 50% speed-up in training without compromising quality (see Tab. 4). We use separate MLPs since the density MLP and color MLP are used separately in the Marching Cubes and surface rendering processes, respectively. We empirically find this MLP separation is necessary for DiffMC fine-tuning (described later in Sec. 3.3) as it largely improves the optimization stability which is critical to large-scale training.

While being largely simplified, our transformer model can effectively transform input posed images into a triplane NeRF for density and color decoding. The density and color are then used to achieve both radiance field rendering for 1st-stage volume initialization (Sec. 3.2) and surface extraction+rendering for 2nd-stage mesh reconstruction (Sec. 3.3).

3.2 Stage 1: Efficient Training for Volume Rendering

We train our model with ray marching-based radiance field rendering (as in NeRF [41]) to bootstrap our model, providing good initialized weights for mesh reconstruction. Instead of training directly using high-res (512×512) input images like prior LRM works [23, 30], we develop an efficient training scheme, inspired by ViT [16]. Specifically, we first pretrain our model with 256×256 images until convergence and then finetune it for fewer iterations with 512×512 images. This training scheme cuts the computing cost significantly, hence obtaining much better quality than training at 512-res from scratch given the same amount of compute (see Tab. 1).

256-res pretraining. The pre-training uses 256-resolution images for both model input and output. We use a batch size of 8 objects per GPU and sample

[3] This operator is logically the same to a shared 2D-deconvolution with stride 8 and kernel 8 on each plane.

128 points per ray during ray marching. In contrast to training with 512-res images from scratch (like previous LRMs), our training efficiency benefits from the low-res pre-training from two factors: shorter sequence length for computing self-attention and fewer samples per ray for volume rendering (compared to our high-res fine-tuning).

512-res finetuning. For high-res fine-tuning, we use 512-resolution images for the model input and output. We use a batch size of 2 per GPU and densely sample 512 points per ray, which is 4× the number in low-res pretraining. Here, we compensate for the increased computational costs (from longer token sequences and denser ray samples) by reducing the batch size 4 times, achieving a similar training speed (per iteration) to the low-res pretraining. It's also worth noting that our resolution-varying pretraining and finetuning scheme benefits from the use of Plücker coordinates for camera conditioning. It avoids the positional embedding interpolation [16, 46] and has inherent spatial smoothness.

Loss. We use an L2 regression loss $L_{v,r}$ and a perceptual loss $L_{v,p}$ (proposed in [9]) to supervise the renderings from both phases. Since rendering full-res images is not affordable for volume rendering, we randomly sample a 128×128 patch from each target 256- or 512-res image for supervision with both losses. We also randomly sample 4096 pixels per target image for additional L2 supervision, allowing the model to capture global information beyond a single patch. The loss for volume rendering training is expressed by:

$$L_v = L_{v,r} + w_{v,p} * L_{v,p} \tag{1}$$

where we use $w_{v,p} = 0.5$ for both 256- and 512-res training.

3.3 Stage 2: Stable Training for Surface Rendering

Once trained with volume rendering, our model already achieves high-fidelity sparse-view NeRF reconstruction, which can be rendered with ray marching to create realistic images. However, directly extracting a mesh from the NeRF's density field results in significant artifacts as reflected by Tab. 2. Therefore, we propose to fine-tune the network using differentiable marching cubes and differentiable rasterization, enabling high-quality feed-forward mesh reconstruction.

Mesh extraction and rendering. We compute a 256^3 density grid by decoding the triplane features and adopt a recent differentiable marching cubes (DiffMC) technique [70] to extract mesh surface from the grid. The DiffMC module is based on a highly optimized CUDA implementation, much faster than existing alternatives [51,52], enabling fast training and inference for mesh reconstruction.

To compute the rendering loss we need to render our generated mesh. We do so using a differentiable rasterizer Nvdiffrast [29] and utilize the triplane features to (neurally) render novel images from our extracted meshes. This full rendering process is akin to deferred shading where we first obtain per-pixel XYZ locations via differentiable rasterization before querying the corresponding triplane

features and regressing per-pixel colors using the color MLP. We supervise novel-view renderings with ground-truth images, optimizing the model for high-quality end-to-end mesh reconstruction.

However, using the rendering loss alone leads to high instability during training and severe floaters in the meshes (see Fig. 3). This is mainly caused by the sparsity of density gradients during mesh rendering: unlike volume rendering which samples points throughout the entire view frustum, mesh (rasterization) rendering is restricted to surface points, lacking gradients in the empty scene space beyond the surface.

Ray opacity loss. To stabilize the training and prevent the formation of floaters, we propose to use a ray opacity loss. This loss is applied to each rendered pixel ray, expressed by:

$$L_\alpha = ||\alpha_{\mathbf{q}}||_1, \quad \alpha_{\mathbf{q}} = 1 - \exp(-\sigma_{\mathbf{q}}||\mathbf{p} - \mathbf{q}||) \tag{2}$$

where $\mathbf{p}$ represents the ground truth surface point along the pixel ray, $\mathbf{q}$ is randomly sampled along the ray between $\mathbf{p}$ and camera origin, and $\sigma_{\mathbf{q}}$ is the volume density at $\mathbf{q}$; when no surface exists for a pixel, we sample $\mathbf{q}$ inside the object bounding box along the ray and use the far ray-box intersection as $\mathbf{p}$.

In essence, this loss is designed to enforce the empty space in each view frustum to contain near-zero density. Here we minimize the opacity value $\alpha_{\mathbf{q}}$, computed using the ray distance from the sampled point to the surface. This density-to-opacity conversion functions as an effective mechanism for weighting the density supervision along the ray with lower loss values for points sampled closer to the surface. An alternative approach would be to directly regularize the density $\sigma_{\mathbf{q}}$ using an L1 loss, but we found this to lead to holes in surfaces as all points contribute the same, regardless of their distance to the surface.

Combined losses. To measure the visual difference between our renderings and ground-truth (GT) images, we use an L2 loss $L_{m,r}$ and a perceptual loss $L_{m,p}$ proposed in [9] similar to stage 1. To compute the ray opacity loss L_α, we obtain surface points using the GT depth maps. In addition, to further improve our geometry accuracy and smoothness, we apply an L2 normal loss L_n to supervise the face normals of our extracted mesh with GT normal maps in foreground regions. Our final loss for mesh reconstruction is

$$L_m = L_{m,r} + w_{m,p} * L_{m,p} + w_\alpha * L_\alpha + w_n * L_n \tag{3}$$

where we use $w_{m,p} = 2$, $w_\alpha = 0.5$ and $w_n = 1$ in our experiments. Since mesh rasterization is significantly cheaper than volume ray marching, we render the images at full resolution (e.g., 512×512 in our experiment) for supervision, instead of the random patches+rays used in Stage 1 volume training.

4 Experiments

4.1 Dataset and Evaluation Protocols

Our model is trained on the Objaverse dataset [14] (730K objects) for the 1st-stage volume rendering training and is subsequently fine-tuned on the Objaverse-LVIS subset (46K objects) for the 2nd-stage surface rendering finetuning. Empirically, the Objaverse-LVIS subset has higher quality and previous work [13,48,69] shows that fine-tuning favors quality more than quantity. We thus follow this common practice. We evaluate the reconstruction quality of MeshLRM alongside other existing methods on the GSO [17], NeRF-Synthetic [41], and OpenIllumination [32] datasets, employing PSNR, SSIM, and LPIPS as metrics for rendering quality and bi-directional Chamfer distance (CD) as the metric for mesh geometry quality. For the CD metric, since we cannot expect the model to precisely reconstruct unseen parts, we cast rays from all test views and sample 100K points at the ray-surface intersections for each object. The Chamfer distance is computed using the points sampled from both the ground-truth and reconstructed mesh within a $[-1, 1]^3$ bounding box, measured in 10^{-3} unit. All results are generated using four sparse-view input images.

4.2 Analysis and Ablation Study

Volume rendering (Stage 1) training strategies. To verify the effectiveness of our training strategy that uses 256-res pretraining and 512-res fine-tuning (details in Sec. 3.2), we compare with a model having the same architecture but trained with high-resolution only (i.e., 512-res from scratch). Tab. 1 shows the quantitative results on the GSO dataset with detailed training settings and timings used for the two training strategies. As seen in the table, with the same total compute budget of 64 hours and 128 GPUs, our low-to-high-res training strategy achieves significantly better performance, with a 2.6dB increase in PSNR. The key to enabling this is our fast low-res pretraining, which takes only 2.6 seconds per iteration, allowing for many more iterations to be trained within a shorter period and enabling much faster convergence. Benefiting from effective pretraining, our high-res finetuning can be trained with a smaller batch size that reduces the iteration time. In general, our 1st-stage training strategy significantly accelerates the LRM training, leading to high-quality NeRF reconstruction. This naturally improves the mesh reconstruction quality in the second stage.

Effectiveness of surface fine-tuning (Stage 2). We justify the effectiveness and significance of our 2nd-stage surface fine-tuning by comparing our final meshes with those directly extracted from 1st-stage model using marching cubes on GSO dataset. The quantitative results are shown in Tab. 2, and the qualitative results are presented in the Appendix. Note that, while our 1st-stage NeRF reconstruction, i.e. MeshLRM (NeRF), can achieve high volume rendering quality, directly extracting meshes from it with Marching Cubes (MC), i.e. 'MeshLRM (NeRF)+MC', leads to a significant drop of rendering quality on all

Table 1: Comparison between the model trained from scratch using 512 resolution images and the model trained by our pretraining + finetuning strategy. Note the two models used the same compute budget — 128 A100 (40G VRAM) GPUs for 64 hours.

	PSNR↑	SSIM↑	LPIPS↓	#Pts/Ray	BatchSize	T_{iter}	#Iter	T_{total}
512-res from scratch	25.53	0.892	0.123	512	1024	7.2s	32k	64hrs
256-res pretrain	**28.13**	**0.923**	**0.093**	128	1024	2.6s	60k	44hrs
512-res fine-tune				512	256	3.6s	20k	20hrs

metrics. On the other hand, our final MeshLRM model, fine-tuned with DiffMC-based mesh rendering, achieves significantly better mesh rendering quality and geometry quality than the MeshLRM (NeRF)+MC baseline, notably increasing PSNR by 2.5dB and lowering CD by 0.58. This mesh rendering quality is even comparable — with a slight decrease in PSNR but improvements in SSIM and LPIPS — to our 1st-stage volume rendering results of MeshLRM (NeRF).

Table 2: The effectiveness of our surface rendering fine-tuning stage. 'MeshLRM (NeRF)' is our first stage model. 'MC' is Marching Cube. Volume rendering for first row; surface rendering for last two rows.

	PSNR↑	SSIM↑	LPIPS↓	CD↓
MeshLRM (NeRF)	**28.13**	0.923	0.093	-
MeshLRM (NeRF)+MC	25.48	0.903	0.097	3.26
MeshLRM	27.93	**0.925**	**0.081**	**2.68**

Table 3: Ablation study for losses used in surface rendering fine-tuning on GSO dataset.

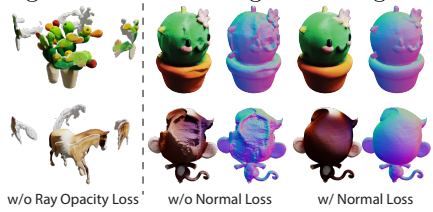
	PSNR↑	SSIM↑	LPIPS↓	CD↓
w/o Ray Opacity Loss	18.46	0.836	0.218	51.98
w/o Normal Loss	**27.93**	**0.925**	**0.080**	2.87
Ours (full model)	**27.93**	**0.925**	0.081	**2.68**

Surface fine-tuning losses. We demonstrate the significance of the different losses applied during the mesh fine-tuning stage (details in Sec. 3.3) with an ablation study in Tab. 3 and Fig. 3. We observe a significant performance drop in the model without our proposed ray opacity loss, leading to severe floater artifacts as shown in Fig. 3. This is because, without supervision on empty space, the model may produce floaters to overfit to the training views rather than learning the correct geometry, as it does not receive a penalty for inaccuracies in empty space. The normal loss particularly helps to generate better geometry, i.e. a lower Chamfer distance. While the normal loss does not lead to quantitative improvements in rendering (metrics are about the same) on the GSO dataset, we still observe qualitative improvements with this loss, especially when using generated images, as shown in Fig. 3 (right). Empirically, the normal loss leads to better robustness in handling inconsistent input views, a common challenge in text-to-3D or image-to-3D scenarios (e.g. the cactus). Moreover, the geometric improvements from the normal loss are also crucial for real applications to use our meshes in 3D engines (like the one in Fig. 1), where accurate shapes are critical for high-fidelity physically-based rendering.

Table 4: Comparison between using big MLP and tiny MLP as triplane decoder. Both are trained for 60k iterations. T_{iter} refers to the training time per iter. Results are volume rendering with Stage-1 256-res pretrained models.

	PSNR↑	SSIM↑	LPIPS↓	T_{iter}
Triplane+Big MLP	27.44	**0.915**	0.081	3.6s
Triplane+Tiny MLP	**27.47**	**0.915**	**0.080**	**2.7s**

Fig. 3: Ablation study on losses used in Stage-2 surface rendering fine-tuning.



Tiny MLPs. To justify our choices of using tiny MLPs for triplane coding, instead of a large one in previous LRMs, we compare with a version that replaces the two tiny MLPs with a 10-layer (i.e., 9 hidden layers) 64-width shared MLP decoder (as used in [23,30]). The quantitative results on the GSO dataset are in Tab. 4. We find that the tiny MLP can achieve similar performance to the large MLP while offering a notable training speed advantage (2.7s / step vs 3.6s / step). Moreover, we observe that the heavy shared MLP hardly converge during the Stage 2 DiffMC training while the tiny separate MLPs could.

4.3 Comparison Against Baselines

Table 5: Comparison with feed-forward approaches on GSO dataset. T_{infer} is the wall-clock time from images to 3D representation (triplane for first two rows and mesh for last two rows), and T_{train} is the training budget; all reported using A100 GPUs. FPS is for rendering 512×512 resolution images from the corresponding representations.

	Render	FPS	T_{infer}	Params	T_{train} (GPU×day)	Quality (GSO) PSNR↑	SSIM↑	LPIPS↓	CD↓
In3D-LRM [30]	128 pts/ray	0.5	0.07s	500M	128×7	26.54	0.893	**0.064**	-
MeshLRM (NeRF)	512 pts/ray	**2**	**0.06s**	300M	**128×2.7**	**28.13**	**0.923**	0.093	-
In3D-LRM [30]+MC	Mesh	>60	1s	500M	128×7	23.70	0.875	0.111	3.40
MeshLRM		>60	**0.8s**	300M	**128×3**	**27.93**	**0.925**	**0.081**	**2.68**

Table 6: Comparison with optimization approaches on Open-Illumination and NeRF-Synthetic datasets. Inference time is wall-clock time with a single A100 GPU.

	Inference Time	OpenIllumination PSNR↑	SSIM↑	LPIPS↓	NeRF-Synthetic PSNR↑	SSIM↑	LPIPS↓	CD↓
FreeNeRF [74]	3hrs	12.21	0.797	0.358	18.81	0.808	0.188	-
ZeroRF [54]	25min	24.42	0.930	0.098	**21.94**	**0.856**	0.139	6.01
MeshLRM	**0.8s**	**26.10**	**0.940**	**0.070**	21.85	0.850	**0.137**	**4.94**

Comparisons with feed-forward methods. To the best of our knowledge, the LRM model proposed in Instant3D [30] is the only previous model that can

also achieve 3D object reconstruction from four sparse views. As reported by Instant3D, previous cost-volume-based feed-forward methods (like SparseNeus [38]) cannot handle this challenging case. Therefore, we focus on comparing with the LRM model in Instant3D (labeled as In3D-LRM) on the GSO dataset, showing quantitative results with detailed training settings and timings in Tab. 5. In the table, we show our results from both stages and compare them with both In3D-LRM and In3D-LRM + MC; for In3D-LRM + MC, Marching Cubes is directly applied to extract meshes from their NeRF reconstruction.

As shown in Tab. 5, our model, across both stages, achieves superior quality compared to Instant3D. As expected, In3D-LRM + MC leads to much worse rendering quality than In3D-LRM, due to the quality drop caused by MC. Thanks to our effective 2nd-Stage DiffMC-based training, MeshLRM can reconstruct significantly better meshes in terms of both rendering quality (with 4.2dB PSNR and 0.05 SSIM improvements) and geometry quality (lowering the CD by 0.72). Our mesh rendering is even better than their volume rendering (In3D-LRM), notably increasing their PSNR by 1.4dB and SSIM by 0.03, while achieving substantially faster rendering speed because of outputting meshes. Our superior quality is also reflected by the qualitative comparisons in Fig. 4, where our mesh renderings reproduce more texture and geometric details than the baselines. From Tab. 5, we can also see that our high reconstruction quality is even achieved with a smaller model size and substantially less compute, less than half of the total GPUxDay compute cost required for Instant3D-LRM. This is enabled by our simplified LRM architecture designs and efficient low-res-to-high-res 1st-stage training strategy, leading to significantly faster training speed and convergence. Besides, our model also leads to a faster inference speed, enabling high-quality mesh reconstruction within 1 second. Overall, our MeshLRM leads to state-of-the-art sparse-view mesh reconstruction quality and achieves high efficiency in parameter utilization, and training, inference, and rendering speed. Meanwhile, our MeshLRM (NeRF) has good performance as well and is a state-of-the-art LRM model for sparse-view NeRF reconstruction.

Comparisons with per-scene optimization methods. We also compare our MeshLRM with recent per-scene optimization methods that are designed for sparse-view NeRF reconstruction, including ZeroRF [54] and FreeNeRF [74]. Because of using per-scene optimization, these methods are much slower, requiring tens of minutes and even several hours to reconstruct one single scene. As a result, it is not practical to evaluate them on the GSO dataset, comprising more than 1000 objects. Therefore, we conduct this experiment on the NeRF-Synthetic and OpenIllumination datasets, following the settings used in ZeroRF [54] with four input views. Tab. 6 shows the quantitative novel view synthesis results, comparing our mesh rendering quality with their volume rendering quality. Since ZeroRF also offers a post-processing step to improve their mesh reconstruction from Marching Cubes, we also compare our mesh geometry with theirs using the CD metric on the NeRF-Synthetic dataset (where GT meshes are accessible).

As shown in Tab. 6, our NeRF-Synthetic results outperform FreeNeRF significantly; we also achieve comparable rendering quality and superior geometry

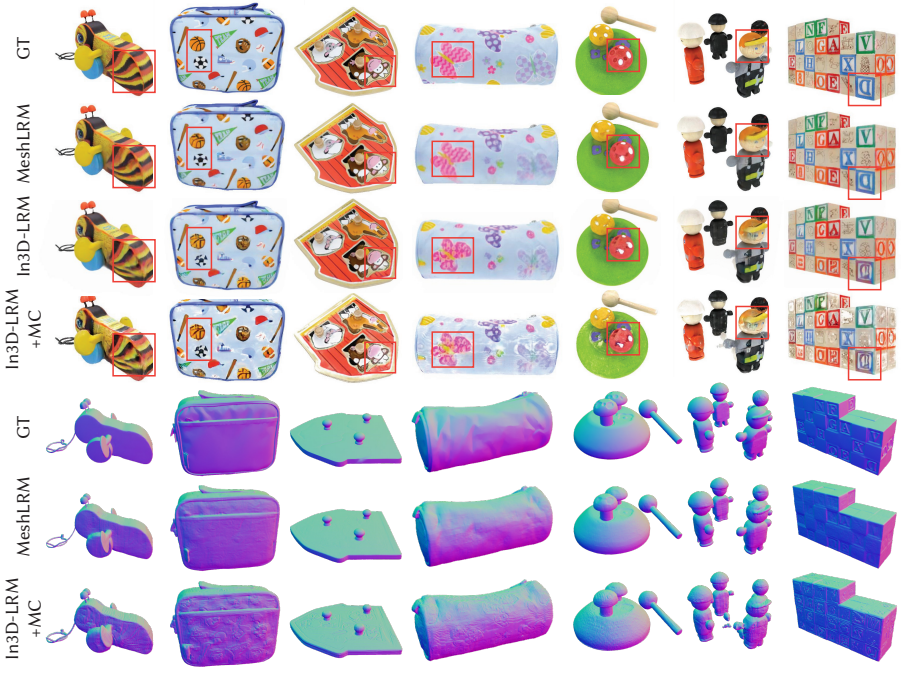


Fig. 4: Qualitative comparison between MeshLRM and other feed-forward methods. 'In3D-LRM' is the Triplane-LRM in Instant3D [30]; 'MC' is Marching Cube. 'In3D-LRM' uses volume rendering and others use surface rendering.

quality to ZeroRF. On the challenging OpenIllumination real dataset, our approach outperforms both FreeNeRF and ZeroRF by a large margin, consistently for all three metrics. The OpenIllumination dataset, consisting of real captured images, demonstrates our model's capability to generalize to real captures for practical 3D reconstruction applications, despite being trained only on rendered images. In addition to our superior (or comparable) quality, our feed-forward mesh reconstruction approach is significantly faster than these optimization-based NeRF methods, in terms of both reconstruction (1000× to 10000× speed up) and rendering.

5 Applications

Our approach achieves efficient and high-quality mesh reconstruction in less than one second, thus naturally enabling many 3D generation applications, when combined with multi-view image generation techniques.

Text-to-3D Generation We apply the multi-view diffusion models from Instant3D [30] to generate 4-view images from text inputs, followed by our MeshLRM to achieve text-to-3D generation. Our results and comparison with the original Instant3D pipeline with In3D-LRM (that is already quantitatively com-

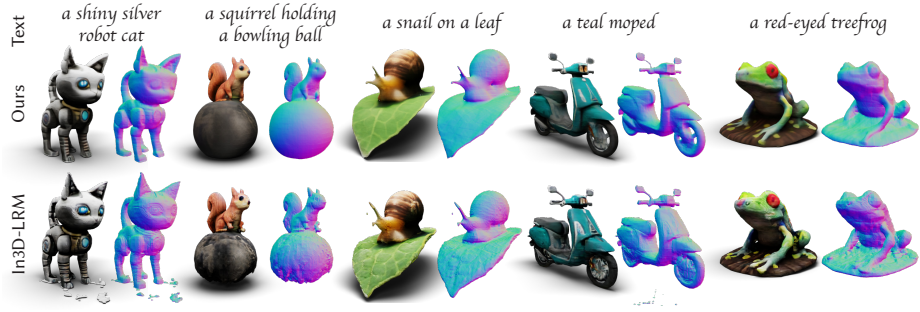


Fig. 5: Text-to-3D results by applying Instant3D's [30] diffusion model to generate 4-view images from a text input. Our method can generate far more accurate and smoother geometry, along with sharp textures.

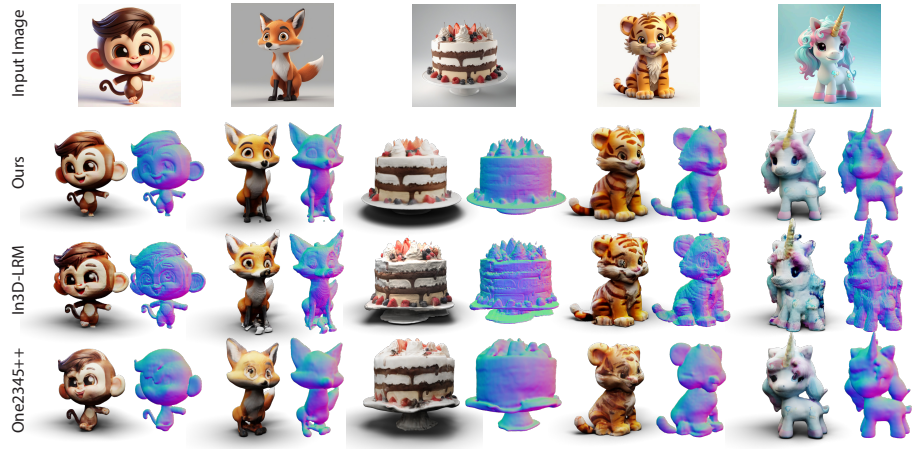


Fig. 6: Image-to-3D results by applying Zero123++ [53] to generate 6 multi-view images from a single image input. Our results outperform other reconstruction methods. Note that our model is trained on 4 views and can zero-shot generalize to 6 views.

pared in Tab. 5 in terms of a reconstruction task) are shown in Fig. 5. Our approach leads to better mesh quality and fewer rendering artifacts.

Image-to-3D Generation We apply Zero123++ [53] v1.2 to generate multi-view images from a single image input, followed by our MeshLRM to achieve image-to-3D generation. Our results and comparison with the One2345++ [34], which uses a diffusion-based reconstruction model, are shown in Fig. 6. Our results are of much higher quality, containing sharper textures and more faithful geometric details than One2345++.

6 Conclusion

In this paper, we have presented MeshLRM, a novel Large Reconstruction Model that can directly output high-quality meshes. We achieved it by applying the Differentiable Marching Cubes (DiffMC) method and differentiable rasterization to fine-tune a pre-trained NeRF-based LRM trained with volume rendering. As DiffMC requires backbone efficiency, we brought multiple improvements (tiny shared MLP and simplified image tokenization) to the LRM architecture, facilitating both NeRF and mesh training. We also found that a low-to-high-res training strategy can significantly accelerate the training of the NeRF-based model. Compared with existing methods, our method has both quality increase and speed improvement and is the only one that can output high-quality meshes. In addition, we showed that our method can be directly applied to applications like text-to-3D and image-to-3D generation. As meshes are the most widely accepted format for 3D assets in the industry, we believe our method takes a step towards automating 3D asset creation and potentially opens up new possibilities with novel 3D workflows.

Acknowledgement

Thanks to Nathan Carr for his support on this project and to Linghao Chen for helping to run the One2345++ baseline.

References

1. Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S.M., Szeliski, R.: Building rome in a day. Communications of the ACM **54**(10), 105–112 (2011)
2. Anciukevičius, T., Xu, Z., Fisher, M., Henderson, P., Bilen, H., Mitra, N.J., Guerrero, P.: Renderdiffusion: Image diffusion for 3d reconstruction, inpainting and generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12608–12618 (2023)
3. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
4. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 9650–9660 (2021)
5. Chan, E.R., Lin, C.Z., Chan, M.A., Nagano, K., Pan, B., De Mello, S., Gallo, O., Guibas, L.J., Tremblay, J., Khamis, S., et al.: Efficient geometry-aware 3d generative adversarial networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16123–16133 (2022)
6. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. In: European Conference on Computer Vision (ECCV) (2022)
7. Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., Su, H.: Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14124–14133 (2021)

8. Chen, H., Gu, J., Chen, A., Tian, W., Tu, Z., Liu, L., Su, H.: Single-stage diffusion nerf: A unified approach to 3d generation and reconstruction. arXiv preprint arXiv:2304.06714 (2023)
9. Chen, Q., Koltun, V.: Photographic image synthesis with cascaded refinement networks. In: Proceedings of the IEEE international conference on computer vision. pp. 1511–1520 (2017)
10. Chen, R., Chen, Y., Jiao, N., Jia, K.: 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)
11. Cheng, S., Xu, Z., Zhu, S., Li, Z., Li, L.E., Ramamoorthi, R., Su, H.: Deep stereo using adaptive thin volume representation with uncertainty awareness. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2524–2534 (2020)
12. Collins, J., Goel, S., Deng, K., Luthra, A., Xu, L., Gundogdu, E., Zhang, X., Vicente, T.F.Y., Dideriksen, T., Arora, H., et al.: Abo: Dataset and benchmarks for real-world 3d object understanding. In: CVPR. pp. 21126–21136 (2022)
13. Dai, X., Hou, J., Ma, C.Y., Tsai, S., Wang, J., Wang, R., Zhang, P., Vandenhende, S., Wang, X., Dubey, A., et al.: Emu: Enhancing image generation models using photogenic needles in a haystack. arXiv preprint arXiv:2309.15807 (2023)
14. Deitke, M., Schwenk, D., Salvador, J., Weihs, L., Michel, O., VanderBilt, E., Schmidt, L., Ehsani, K., Kembhavi, A., Farhadi, A.: Objaverse: A universe of annotated 3d objects. In: CVPR. pp. 13142–13153 (2023)
15. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186 (2019)
16. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2020)
17. Downs, L., Francis, A., Koenig, N., Kinman, B., Hickman, R., Reymann, K., McHugh, T.B., Vanhoucke, V.: Google scanned objects: A high-quality dataset of 3d scanned household items. In: 2022 International Conference on Robotics and Automation (ICRA). pp. 2553–2560. IEEE (2022)
18. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5501–5510 (2022)
19. Furukawa, Y., Ponce, J.: Accurate, dense, and robust multiview stereopsis. IEEE transactions on pattern analysis and machine intelligence **32**(8), 1362–1376 (2009)
20. Gao, J., Shen, T., Wang, Z., Chen, W., Yin, K., Li, D., Litany, O., Gojcic, Z., Fidler, S.: Get3d: A generative model of high quality 3d textured shapes learned from images. In: Advances In Neural Information Processing Systems (2022)
21. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in neural information processing systems **27** (2014)
22. Henzler, P., Mitra, N.J., Ritschel, T.: Escaping plato's cave: 3d shape from adversarial rendering. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (October 2019)

23. Hong, Y., Zhang, K., Gu, J., Bi, S., Zhou, Y., Liu, D., Liu, F., Sunkavalli, K., Bui, T., Tan, H.: Lrm: Large reconstruction model for single image to 3d. In: International Conference on Learning Representations (2024)
24. Huang, P.H., Matzen, K., Kopf, J., Ahuja, N., Huang, J.B.: Deepmvs: Learning multi-view stereopsis. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
25. Im, S., Jeon, H.G., Lin, S., Kweon, I.S.: Dpsnet: End-to-end deep plane sweep stereo. arXiv preprint arXiv:1905.00538 (2019)
26. Johari, M.M., Lepoittevin, Y., Fleuret, F.: Geonerf: Generalizing nerf with geometry priors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18365–18375 (2022)
27. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proceedings of the fourth Eurographics symposium on Geometry processing. vol. 7, p. 0 (2006)
28. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (2023)
29. Laine, S., Hellsten, J., Karras, T., Seol, Y., Lehtinen, J., Aila, T.: Modular primitives for high-performance differentiable rendering. ACM Transactions on Graphics **39**(6) (2020)
30. Li, J., Tan, H., Zhang, K., Xu, Z., Luan, F., Xu, Y., Hong, Y., Sunkavalli, K., Shakhnarovich, G., Bi, S.: Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model. In: International Conference on Learning Representations (2024)
31. Lin, C.H., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M.Y., Lin, T.Y.: Magic3d: High-resolution text-to-3d content creation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2023)
32. Liu, I., Chen, L., Fu, Z., Wu, L., Jin, H., Li, Z., Wong, C.M.R., Xu, Y., Ramamoorthi, R., Xu, Z., et al.: Openillumination: A multi-illumination dataset for inverse rendering evaluation on real objects. Advances in Neural Information Processing Systems **36** (2024)
33. Liu, L., Gu, J., Zaw Lin, K., Chua, T.S., Theobalt, C.: Neural sparse voxel fields. Advances in Neural Information Processing Systems **33**, 15651–15663 (2020)
34. Liu, M., Shi, R., Chen, L., Zhang, Z., Xu, C., Wei, X., Chen, H., Zeng, C., Gu, J., Su, H.: One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion. arXiv preprint arXiv:2311.07885 (2023)
35. Liu, M., Xu, C., Jin, H., Chen, L., Varma T, M., Xu, Z., Su, H.: 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)
36. Liu, Y., Lin, C., Zeng, Z., Long, X., Liu, L., Komura, T., Wang, W.: Syncdreamer: Generating multiview-consistent images from a single-view image. arXiv preprint arXiv:2309.03453 (2023)
37. Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehrmann, A., Sheikh, Y.: Neural volumes: Learning dynamic renderable volumes from images. arXiv preprint arXiv:1906.07751 (2019)
38. Long, X., Lin, C., Wang, P., Komura, T., Wang, W.: Sparseneus: Fast generalizable neural surface reconstruction from sparse views. In: European Conference on Computer Vision. pp. 210–227. Springer (2022)
39. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. In: Seminal graphics: pioneering efforts that shaped the field. pp. 347–353 (1998)

40. Metzer, G., Richardson, E., Patashnik, O., Giryes, R., Cohen-Or, D.: Latent-nerf for shape-guided generation of 3d shapes and textures. arXiv preprint arXiv:2211.07600 (2022)
41. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV (2020)
42. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (ToG) **41**(4), 1–15 (2022)
43. Oechsle, M., Peng, S., Geiger, A.: Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In: International Conference on Computer Vision (ICCV) (2021)
44. Plücker, J.: Xvii. on a new geometry of space. Philosophical Transactions of the Royal Society of London (155), 725–791 (1865)
45. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: Text-to-3d using 2d diffusion. arXiv preprint arXiv:2209.14988 (2022)
46. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021)
47. Rakotosaona, M.J., Manhardt, F., Arroyo, D.M., Niemeyer, M., Kundu, A., Tombari, F.: Nerfmeshing: Distilling neural radiance fields into geometrically-accurate 3d meshes. In: International Conference on 3D Vision (3DV) (2023)
48. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10684–10695 (2022)
49. Schonberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4104–4113 (2016)
50. Schönberger, J.L., Zheng, E., Pollefeys, M., Frahm, J.M.: Pixelwise view selection for unstructured multi-view stereo. In: European Conference on Computer Vision (ECCV) (2016)
51. Shen, T., Gao, J., Yin, K., Liu, M.Y., Fidler, S.: Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In: Advances in Neural Information Processing Systems (NeurIPS) (2021)
52. Shen, T., Munkberg, J., Hasselgren, J., Yin, K., Wang, Z., Chen, W., Gojcic, Z., Fidler, S., Sharp, N., Gao, J.: Flexible isosurface extraction for gradient-based mesh optimization. ACM Trans. Graph. **42**(4) (jul 2023). https://doi.org/10.1145/3592430, https://doi.org/10.1145/3592430
53. Shi, R., Chen, H., Zhang, Z., Liu, M., Xu, C., Wei, X., Chen, L., Zeng, C., Su, H.: Zero123++: a single image to consistent multi-view diffusion base model (2023)
54. Shi, R., Wei, X., Wang, C., Su, H.: Zerorf: Fast sparse view 360 {\deg} reconstruction with zero pretraining. arXiv preprint arXiv:2312.09249 (2023)
55. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: exploring photo collections in 3d. In: ACM siggraph 2006 papers. pp. 835–846 (2006)
56. Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics. In: International conference on machine learning. pp. 2256–2265. PMLR (2015)
57. Suhail, M., Esteves, C., Sigal, L., Makadia, A.: Light field neural rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8269–8279 (2022)

58. Tang, C., Tan, P.: Ba-net: Dense bundle adjustment network. arXiv preprint arXiv:1806.04807 (2018)
59. Tang, J., Zhou, H., Chen, X., Hu, T., Ding, E., Wang, J., Zeng, G.: Delicate textured mesh recovery from nerf via adaptive surface refinement. arXiv preprint arXiv:2303.02091 (2022)
60. Tang, J., Wang, T., Zhang, B., Zhang, T., Yi, R., Ma, L., Chen, D.: Make-it-3d: High-fidelity 3d creation from a single image with diffusion prior. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 22819–22829 (October 2023)
61. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
62. Vijayanarasimhan, S., Ricco, S., Schmid, C., Sukthankar, R., Fragkiadaki, K.: Sfm-net: Learning of structure and motion from video. arXiv preprint arXiv:1704.07804 (2017)
63. Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In: Advances in Neural Information Processing Systems (2021)
64. Wang, P., Tan, H., Bi, S., Xu, Y., Luan, F., Sunkavalli, K., Wang, W., Xu, Z., Zhang, K.: Pf-lrm: Pose-free large reconstruction model for joint pose and shape prediction. arXiv preprint arXiv:2311.12024 (2023)
65. Wang, Q., Wang, Z., Genova, K., Srinivasan, P.P., Zhou, H., Barron, J.T., Martin-Brualla, R., Snavely, N., Funkhouser, T.: Ibrnet: Learning multi-view image-based rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4690–4699 (2021)
66. Wang, S., Leroy, V., Cabon, Y., Chidlovskii, B., Revaud, J.: Dust3r: Geometric 3d vision made easy. arXiv preprint arXiv:2312.14132 (2023)
67. Wang, T., Zhang, B., Zhang, T., Gu, S., Bao, J., Baltrusaitis, T., Shen, J., Chen, D., Wen, F., Chen, Q., et al.: Rodin: A generative model for sculpting 3d digital avatars using diffusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4563–4573 (2023)
68. Wang, Z., Lu, C., Wang, Y., Bao, F., Li, C., Su, H., Zhu, J.: Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. In: Thirty-seventh Conference on Neural Information Processing Systems (2023)
69. Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A.W., Lester, B., Du, N., Dai, A.M., Le, Q.V.: Finetuned language models are zero-shot learners. In: International Conference on Learning Representations (2021)
70. Wei, X., Xiang, F., Bi, S., Chen, A., Sunkavalli, K., Xu, Z., Su, H.: Neumanifold: Neural watertight manifold reconstruction with efficient and high-quality rendering support. arXiv preprint arXiv:2305.17134 (2023)
71. Wu, J., Zhang, C., Xue, T., Freeman, W.T., Tenenbaum, J.B.: Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In: Advances in Neural Information Processing Systems. pp. 82–90 (2016)
72. Xu, Q., Xu, Z., Philip, J., Bi, S., Shu, Z., Sunkavalli, K., Neumann, U.: Point-nerf: Point-based neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5438–5448 (2022)
73. Xu, Y., Tan, H., Luan, F., Bi, S., Wang, P., Li, J., Shi, Z., Sunkavalli, K., Wetzstein, G., Xu, Z., Zhang, K.: Dmv3d: Denoising multi-view diffusion using 3d large reconstruction model (2023)

74. Yang, J., Pavone, M., Wang, Y.: Freenerf: Improving few-shot neural rendering with free frequency regularization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8254–8263 (2023)
75. Yao, Y., Luo, Z., Li, S., Fang, T., Quan, L.: Mvsnet: Depth inference for unstructured multi-view stereo. In: Proceedings of the European conference on computer vision (ECCV). pp. 767–783 (2018)
76. Yariv, L., Gu, J., Kasten, Y., Lipman, Y.: Volume rendering of neural implicit surfaces. In: Thirty-Fifth Conference on Neural Information Processing Systems (2021)
77. Yariv, L., Hedman, P., Reiser, C., Verbin, D., Srinivasan, P.P., Szeliski, R., Barron, J.T., Mildenhall, B.: Bakedsdf: Meshing neural sdfs for real-time view synthesis. arXiv (2023)
78. Yariv, L., Kasten, Y., Moran, D., Galun, M., Atzmon, M., Ronen, B., Lipman, Y.: Multiview neural surface reconstruction by disentangling geometry and appearance. Advances in Neural Information Processing Systems **33**, 2492–2502 (2020)
79. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelnerf: Neural radiance fields from one or few images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4578–4587 (2021)
80. Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo magnification: Learning view synthesis using multiplane images. arXiv preprint arXiv:1805.09817 (2018)

A Effectiveness of surface fine-tuning (Stage 2).

We have discussed the effectiveness of our second-stage fine-tuning in Sec. 4.2 with quantitative results shown in Tab. 2 in the main paper. We now show examples of qualitative results in Fig. 7, comparing our final meshes (MeshLRM) with the meshes (MeshLRM (NeRF) + MC) generated by directly applying Marching Cubes on the 1st-stage model's NeRF results. As shown in the figure, the MeshLRM (NeRF) + MC baseline can lead to severe artifacts with non-smooth surfaces and even holes, whereas our final model with the surface rendering fine-tuning can effectively address these issues and produce high-quality meshes and realistic renderings. These qualitative results demonstrate the large visual improvements achieved by our final model for mesh reconstruction, reflecting the big quantitative improvements shown in the paper.

B ABO and OpenIllumination datasets

We compare our proposed method with In3D-LRM [30] on 1000 randomly sampled examples in the ABO [12] dataset with quantitative results shown in Tab. 7. In particular, ABO is a highly challenging synthetic dataset that contains many objects with complex glossy materials. This is a big challenge for our model, as well as In3D-LRM's model, since both assume Lambertian appearance. In this case, the NeRF models are often better than the mesh models, since NeRF rendering can possibly fake glossy effects by using colors from multiple points along each pixel ray while mesh rendering cannot. Nonetheless, as shown in the table, both our NeRF and final mesh model can outperform In3D-LRM's NeRF and

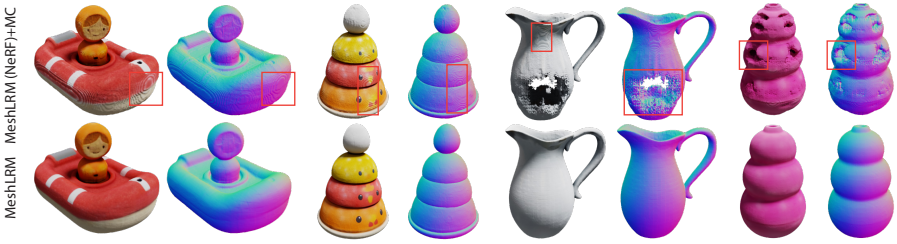


Fig. 7: The model performance drops significantly when applying marching cubes to the volume rendering trained model ('MeshLRM (NeRF) + MC') without our mesh refinement fine-tuning ('MeshLRM').

Table 7: Comparison with feed-forward approaches on ABO [12] and OpenIllumination [32] datasets.

	ABO			OpenIllumination		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
In3D-LRM [30]	27.50	0.896	0.093	8.96	0.568	0.682
MeshLRM (NeRF)	**28.31**	**0.906**	**0.108**	**20.53**	**0.772**	**0.290**
In3D-LRM [30]+MC	22.11	0.850	0.144	8.92	0.507	0.691
MeshLRM	**26.09**	**0.898**	**0.102**	**20.51**	**0.786**	**0.218**

mesh variants respectively. Especially, thanks to our effective MC-based finetuning, our mesh rendering quality surpasses that of In3D-LRM + MC by a large margin (e.g. 3.92dB in PSNR and 0.48 in SSIM).

On the other hand, we also compare with In3D-LRM on the full OpenIllumination [32] dataset in Tab. 7. To clarify, this experiment setting is different from the setting used in Tab. 6 in the main paper, where we follow ZeroRF [54] to test only 8 objects with combined masks in OpenIllumination; here, we evaluate all models on the full dataset with 100 objects with object masks and we also take square crops that tightly bounding the objects from the rendered/GT images to compute rendering metrics, avoiding large background regions. As shown in the table, we observe that In3D-LRM cannot generalize to handle this challenging real dataset, leading to very low PSNRs, despite being trained on the same training data as ours. In contrast, our model can still work well on this out-of-domain dataset with high rendering quality.

C Additional Implementation Details

The transformer size follows the transformer-large config [15]. It has 24 layers and a model width of 1024. Each attention layer has a 16 attention head and each head has a dim of 64. The intermediate dimension of the MLP layer is 4096. We use GeLU activation inside the MLP and use Pre-LN architecture. The layer

normalization is also applied after the input image tokenization, and also before the triplane's unpachifying operator.

For the training of both stages, we use AdamW with $\beta_2 = 0.95$. Other Adam-specific parameters are set as default and we empirically found that the model is not sensitive to them. A weight decay of 0.05 is applied to all parameters excluding the bias terms and layer normalization layers. The learning rate for stage 1 is $4e-4$. We also use cosine learning rate decay and a linear warm-up in the first 2K steps. For stage 2, we use a learning rate of $1e-5$, combined with cosine learning rate decay and a linear warm-up during the first 500 steps. In total, there are 10k fine-tuning steps. The resolution of DiffMC is 256 within a $[-1, 1]^3$ bounding box, which is consistent with the triplane resolution.

D Limitations

Since our model employs surface rendering and assumes only Lambertian appearance without performing any inverse rendering, it is not sufficiently robust when the input images contain complex materials, e.g. some examples in the ABO dataset. The generated mesh may use white color to fit the specular areas. We believe that incorporating inverse rendering into the current pipeline could address this issue. However, this requires sufficient training data with accurate material annotations, which we leave for future exploration. On the other hand, while handling highly sparse input views, our model requires input camera poses. Although poses can be readily obtained from text- or image-to-multi-view models [30,53] for 3D generation tasks, calibrating sparse-view poses for real captures can be a big challenge. In the future, it will be an interesting direction to explore combining our approach with recent pose estimation techniques [64, 66] for joint mesh reconstruction and camera calibration.