tttLRM: Test-Time Training for Long Context and Autoregressive 3D Reconstruction

Chen Wang[1*]   Hao Tan[2]   Wang Yifan[2]   Zhiqin Chen[2]
Yuheng Liu[3*]   Kalyan Sunkavalli[2]   Sai Bi[2]   Lingjie Liu[1†]   Yiwei Hu[2†]
[1]University of Pennsylvania   [2]Adobe Research   [3]UCI
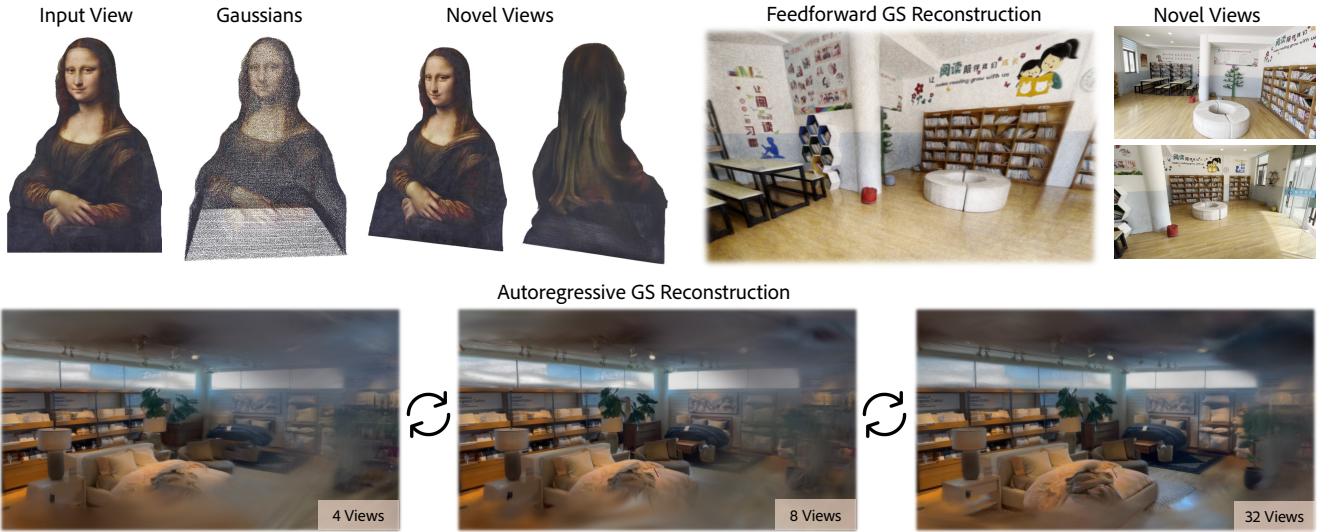https://cwchenwang.github.io/tttLRM

Figure 1. We propose tttLRM, a Large Reconstruction Model based on Test-Time Training, enabling high-resolution, long-context, autoregressive 3D reconstruction. Our model achieves 1) high-resolution (1024px) single-image-to-3D reconstruction via a multi-view generator 2) long-context (64 input views) and feedforward 3DGS reconstruction, and supports 3) autoregressive streaming reconstruction.

Abstract

We propose tttLRM, a novel large 3D reconstruction model that leverages a Test-Time Training (TTT) layer to enable long-context, autoregressive 3D reconstruction with linear computational complexity, further scaling the model's capability. Our framework efficiently compresses multiple image observations into the fast weights of the TTT layer, forming an implicit 3D representation in the latent space that can be decoded into various explicit formats, such as Gaussian Splats (GS) for downstream applications. The online learning variant of our model supports progressive 3D reconstruction and refinement from streaming observations. We demonstrate that pretraining on novel view synthesis tasks effectively transfers to explicit 3D modeling, resulting in improved reconstruction quality and faster convergence. Extensive experiments show that our method achieves superior performance in feedforward 3D Gaussian

reconstruction compared to state-of-the-art approaches on both objects and scenes.

1. Introduction

Reconstructing explicit 3D representations for photo-realistic rendering from streaming visual input is a central goal of 3D reconstruction. This process is similar to how humans perceive the physical world: we observe a continuous visual stream, build an abstract internal representation of the world, and decode this abstraction into explicit 3D only when needed for fine-grained tasks or to recall detailed 3D structure. In light of this human-like process, we aim to enable long-context, autoregressive reconstruction of explicit 3D from streaming visual input.

However, existing 3D reconstruction methods are not designed for long-context scenarios with a memory mechanism. Traditional approaches to generating 3D representations and synthesizing novel views, including Neural Radiance Fields (NeRF) [32, 61] and 3D Gaussian Splat-

---

ting (3DGS) [21] have achieved substantial progress for high-quality rendering, but they either require slow scene-specific optimization or rely on feedforward reconstruction models with limited input-view scalability.

For example, Large Reconstruction Models (LRMs) have been proposed to rapidly reconstruct various 3D representations such as NeRFs [16], meshes [52], and 3DGS [64] from input images. However, these models are typically restricted to only a few input views (e.g., four), which limits their ability to reconstruct large-scale scenes. While Long-LRM [67] extends the number of input views to 32, its use of bidirectional attention layers hinders further scalability and prevents efficient processing of inputs with longer and streamed context, limiting its applicability in real-world scenarios. On the other hand, recent research [12, 18, 35, 66] on implicit latent-space 3D representations has demonstrated superior novel view synthesis quality using purely neural networks. However, despite being feedforward models for reconstruction, their rendering speed is significantly slower than that of explicit representations such as 3DGS due to repetitive network inference, and they lack controllability and interpretability, making them less suitable for many downstream applications.

In this paper, we propose a novel reconstruction model that leverages neural architectures and the knowledge distilled from pretrained implicit latent-space 3D models, decoding them into explicit 3D representations. This design ensures high-quality novel view synthesis with long-context and autoregressive modeling, while maintaining real-time rendering capability through explicit 3D outputs.

Our model builds upon the Test-Time Training (TTT) framework [44, 66] and introduces an architecture composed of LaCT [66] blocks that has only linear computational complexity. We interpret the fast weights of TTT models, which are updated according to inputs during inference, as implicit latent-space 3D representations that can be decoded into various explicit formats such as 3DGS or NeRFs. We demonstrate that, with minimal architectural modification, our framework effectively leverages the pretrained knowledge of large novel view synthesis models [18, 66] for explicit 3D reconstruction. Specifically, our model is trained to *query* the fast weights for different 3D representations, such as a set of virtual view planes for 3DGS, or a triplane feature grid for NeRF-based reconstruction. This design unlocks greater flexibility in the final 3D representation. Furthermore, the fast-weight update schedule can be altered to enable autoregressive 3D reconstruction and refinement with streaming inputs.

We validate our model on both object- and scene-level datasets. Across both datasets, our model achieves superior reconstruction quality compared to baseline methods, while also being highly efficient. We also show that our model supports autoregressive reconstruction, enabling practical real-world applications. The contributions of our paper can be summarized as following:

- We propose ttLRM, a large reconstruction model that enables Test-Time Training for both feedforward long-context and autoregressive modeling.
- We design a unified 3D modeling framework that interprets TTT fast weights into observable and controllable explicit 3D representations.
- We achieve state-of-the-art results on both object- and scene-level datasets, delivering superior quality and efficiency in 3D reconstruction and novel view synthesis.

2. Related Work

Multi-view 3D Reconstruction 3D reconstruction from images has been extensively studied in computer vision. Traditional methods such as structure-from-motion [37] or multi-view stereo (MVS) [14] focus on recovering 3D geometry. Deep learning has enabled feed-forward 3D reconstruction [57, 58], which builds cost volumes using plane sweep for per-view depth estimation. Recently, learning-based MVS approaches [7, 24, 25, 49, 51, 55] directly estimate point clouds from input images and have been applied to camera pose estimation. Test3R [63] optimizes the network at test time in a self-supervised manner to improve 3D reconstruction. Concurrent work TTT3R [7] defines a gradient to update states for point cloud reconstruction. However, none of these methods can produce photo-realistic novel view synthesis.

Neural representations then have emerged as a promising way for both geometry reconstruction and NVS. NeRF [32] represents the scene as a continuous field and leverages a coordinate-based MLP to predict per-point color and density, enabling differential volumetric rendering with rendering-based supervision. Original NeRF takes hours to optimize a single scene and following works improved its training and rendering efficiency using advanced representations, including voxels [27, 42], points [54], hash grids [33], and triplanes [4, 6, 13]. Recently, 3D Gaussian Splatting [17, 21] has become the state-of-the-art neural scene representation. It uses volume rendering and rendering loss for optimization similar to NeRF but represents the scene with simple Gaussian primitives which enables real-time rendering and large-scale scene reconstruction [22, 28]. However, 3DGS still requires optimizing 3D Gaussians from scratch, taking several minutes per scene, whereas our model performs 3D reconstruction within seconds in a feed-forward way.

Learning-based Feedforward 3D Reconstruction The development on learning-based methods enables 3D reconstruction and novel view synthesis by training neural networks on large-scale datasets to directly infer 3D structures without per-scene optimization. Early work utilizes Convolutional Neural Networks (CNN) to predict multi-plane

images [11, 31], points [2, 60] or voxels [41]. Large Reconstruction Models (LRM) [16] propose a transformer-based architecture without 3D inductive bias for 3D object reconstruction from multi-view images, with triplane as the 3D representation. GS-LRM [64] further extends LRM to predict pixel-aligned 3DGS, but the model can only take very few images as input due to the quadratic complexity of attention layers. Similarly, the subsequence approach [5, 8, 45, 53] also apply a feedforward framework with different neural architectures and 3D inductive bias for Gaussian prediction. Mamba-based models [38, 59] has attempted to reduce the complexity of attention layers, but are still limited to very few input views. Long-LRM [67] represents the state of the art in long-sequence Gaussian reconstruction, but it remains limited to 32 input views and relies on additional attention layers. By leveraging TTT, our model achieves longer-context and autoregressive reconstruction with improved NVS quality.

Linear Attention and State Space Models To circumvent the quadratic complexity of attention [46], recent research has explored efficient alternatives that retain contextual expressivity while reducing computational cost. Linear attention models [20, 36, 39] approximate the softmax kernel with linearized feature maps to achieve linear complexity, but uniform compression of past key–value pairs often degrades the upper bound of long sequence modeling.

State Space Models (SSMs) introduce a state variable to represent historical information, similar to classical Recurrent Neural Networks (RNNs). Recent works [9, 15, 29, 43] incorporate attenuation factors into the state updates, allowing the model to retain more recent information while gradually forgetting the distant past. Among them, Mamba [9, 15, 29] proposes "date-dependent decay" to model sequences as continuous-time dynamical systems governed by state transition, but it still cannot compete with transformers in long-context reasoning [48]. Jamba [1] implements a hybrid mamba attention model to improve the performance. Test Time Training (TTT), on the other hand, [3, 44, 66] transforms the problem into an online learning problem and applies modern optimizers to learn the states. DeltaNet [36, 56] and MesaNet [47] share the same idea but use different update rules when updating. Inspired by its success, we introduce Test-Time Training into 3D reconstruction tasks for high-quality long-context novel view synthesis, but with only linear complexity.

3. Method

3.1. Preliminary: TTT and LaCT Layer

We first briefly introduce the fundamentals of TTT and Large Chunk Test-Time Training (LaCT) layer, which form the core building blocks of our model. In sequence modeling, the input is typically represented as a sequence of tokens of length L, denoted by $[\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_L]$, where each token has dimension d: $\mathbf{x}_i \in \mathbb{R}^d$. In standard attention, each input token will be projected into query, key and value vectors, denoted as q_i, k_i, v_i. Each token attends to all others via a dot-product operation, leading to quadratic complexity in sequence length.

TTT [44] learns a set of *fast weights* W that are updated at inference time according to the input to capture the relationship between input tokens. Specifically, it treats the key-value pairs (k_i, v_i) of input tokens as training data to update fast weights using mean-square error: $W \leftarrow W - \eta \nabla \mathcal{L}_{\mathrm{MSE}}(f_W(k), v)$, which can be further applied to queries to obtain the final input $o = f_W(q)$. In this way, the fast weights effectively encode the key–value (KV) cache of the input sequence into a fixed-size neural memory.

Originally, the TTT model [44] updates the fast weights using only a small minibatch (*e.g.* 16 tokens), which results in very low GPU FLOP utilization and difficulty in handling long sequences. Large Chunk Test-Time Training (LaCT) [66] instead updates fast weights with large chunk size (up to 1M tokens). Its chunk-wise update computes the gradient of the summed loss over all keys and values within the chunk. More details can be found in [66].

3.2. Model Architecture

We illustrate our model architecture in Figure 2, using 3DGS reconstruction as an example, though the same framework can be applied to other 3D representations as well. Given a set of posed images, denoted as $\{\mathbf{I}_i \in \mathbb{R}^{H \times W \times 3} | i = 1, 2, .., N\}$, we concatenate them channel-wise with their ray embeddings $\{\mathbf{R}_i \in \mathbb{R}^{H \times W \times 9} | i = 1, 2, .., N\}$ as the positional embedding. After dividing each image into non-overlapping patches of size $p \times p$, we tokenize these image patches using a lightweight linear layer into a sequence of tokens $\mathbf{T}$:

$$\{\mathbf{T}_{i,j}\}_{i=1}^{N} {}_{j=1}^{HW/p^2} = \mathrm{Tokenize}\big(\mathrm{Patchify}([\{\mathbf{I}_i\}_{i=1}^{N}, \{\mathbf{R}\}_{i=1}^{N}])\big),$$

These visual tokens then iteratively update the fast weights W of a set of LaCT blocks using Muon [19] optimizer:

$$\mathbf{T}_i = \mathbf{T}_i + \mathrm{WinAttn}(\mathbf{T}_i), \tag{1}$$

$$W = \mathrm{Update}(\{\mathbf{T}_i\}_{i=1}^{N}), \tag{2}$$

$$\mathbf{T}_i = \mathrm{Apply}(W, \mathbf{T}_i) \tag{3}$$

Each LaCT layer includes a window attention module that captures local relationships within each view. We omit the feedforward layers in the block in the equation for simplicity. The update and apply operations are in linear complexity with respect to the sequence length.

To retrieve information from the fast weights, we introduce a set of virtual tokens that serve as queries to our model. In 3DGS reconstruction, these virtual tokens are virtual views $\{\mathbf{I}_i^{\mathrm{v}} \in \mathbb{R}^{H \times W \times 3} | i = 1, 2, .., M\}$ for GS
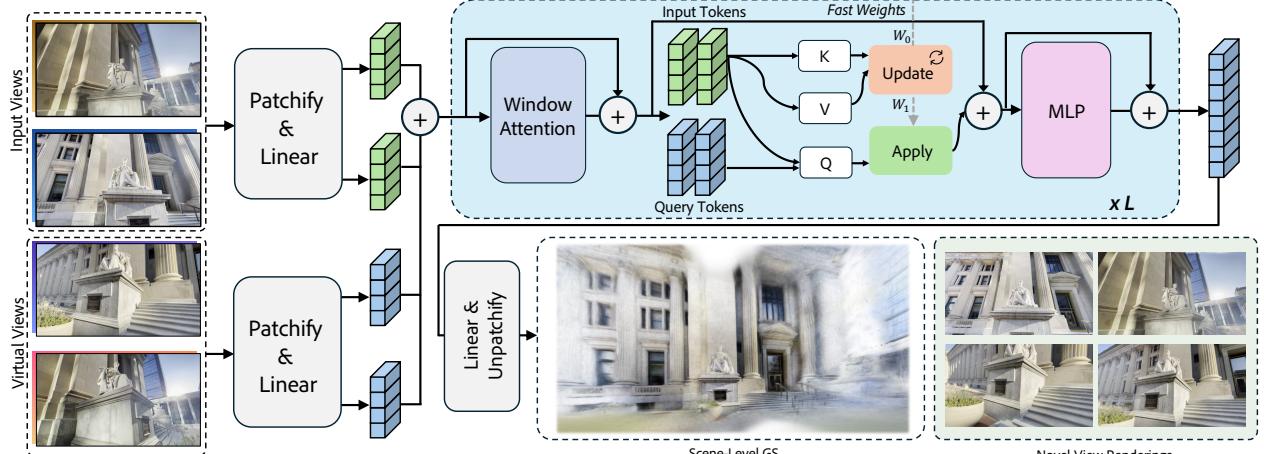
Figure 2. Given a set of posed input images, tttLRM encodes them into tokens (green boxes) after patchifying. The input tokens are fed into the LaCT block (shown in the blue frame) where fast weights are updated accordingly. Another set of virtual tokens (blue boxes) are used to query the updated fast weights, and decoded into 3D representations like 3DGS for high-quality novel view synthesis.

prediction, which will also be patchified and tokenized to $\{\mathbf{T}_{i,j}^{v}\}_{i=1\,j=1}^{N\quad HW/p^2}$. In other 3D representations, such as NeRFs, these query tokens might be derived from triplane features. The virtual tokens are only used in the apply operation without updating the fast weights:

$$\mathbf{T}_i^{v} = \text{Apply}(W, \mathbf{T}_i^{v}) \qquad (4)$$

Given the updated query tokens $\mathbf{T}_i^{v}$, a linear token decoder transforms them into explicit 3D representations, such as per-patch Gaussian parameters in 3DGS reconstruction. The RGB color, scale, rotation, and opacity of each Gaussian are predicted directly. For Gaussian positions, we first decode the depth of each pixel and use a range function (object-centric for object data and linear for scene data) to convert it to real depth. After that, we convert depth to a Gaussian position with known ray locations and directions.

3.3. Autoregressive Reconstruction

Algorithm 1 Autoregressive 3DGS Reconstruction

Input: Reconstructor $\mathcal{F}$ with initial fast weights W_0; input/query view batches $\{(\mathcal{I}_{(b)}, \mathcal{I}_{(b)}^{v})\}_{b=1}^{B}$
Output: Reconstructed GS G
 1: $W \leftarrow W_0$
 2: **for** $b = 1$ to B **do**
 3:     $\_, W \leftarrow \mathcal{F}(W, \mathcal{I}_{(b)})$
 4:     $G_{(b)}, \_ \leftarrow \mathcal{F}(W, \mathcal{I}_{(b)}^{v})$
 5: **end for**
 6: **return** $G_{(B)}$

An important feature of our architecture is its support for autoregressive modeling with streamed input images. To enable this, we modify the update and apply steps to incorporate causal dependencies among tokens. Unlike the standard setting where all input views are jointly processed to

update the fast weights before decoding, the streaming variant performs incremental updates in a causal manner. As illustrated in Algorithm 1, providing our model $\mathcal{F}$, for each incoming mini-batch of views $\mathcal{I}_{(b)}$ (*e.g.*, four images at a time), the model updates the fast weights and immediately predicts the corresponding 3D Gaussian parameters for the new query views $\mathcal{I}_{(b)}^{v}$, returning the current reconstructed Gaussian splat results $G_{(b)}$. This design effectively transforms the model into an RNN-like inference process, where the internal state (fast weights) evolves as new observations arrive, enabling online 3D Gaussian reconstruction.

3.4. Distributed Feedforward Reconstruction

A large number of input views and high-resolution images introduce a substantial number of tokens, leading to a significant increase in both computation and memory cost. A key limitation of prior works lies in their inability to handle long input sequences efficiently, largely due to the lack of parallelism at the sequence level and most methods process all input views within a single device.

To address this limitation, we introduce sequence parallelism for training feedforward reconstruction models, exemplified by 3DGS reconstruction as shown in Figure 3. Specifically, we partition the tokenized input views along the sequence dimension and assign each shard to a separate device. During training:

- Since Gaussians can be predicted independently for each virtual view once the fast weights are synchronized, each GPU predicts pixel-aligned Gaussian primitives for its assigned views (first row).
- The predicted Gaussians from all devices are gathered to form the complete scene representation (second row).
- Each GPU subsequently renders its own set of novel views and computes photometric reconstruction losses against the ground truth, and gradients are all reduced to
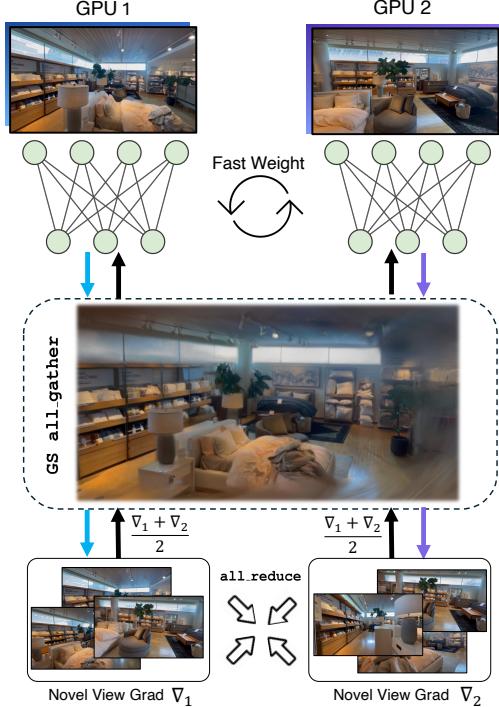
Figure 3. Illustration of distributed feedforward reconstruction training. First, image tokens are sharded across GPUs, and each GPU predicts Gaussians for its assigned virtual views after the fast weights are synchronized. The predicted Gaussians are then gathered to construct the full scene, after which each GPU renders a subset of novel views and computes its respective losses. Gradients are finally all reduced and backpropagated across all devices.

enable sequence-level backpropagation (third row).

Thanks to the linearity of our LaCT fast-weight updates, gradients of the fast weights across devices can be easily synchronized through PyTorch Distributed Data Parallel (DDP), ensuring consistent global optimization. During inference, the distributed reconstruction also allows us to accelerate the reconstruction with more GPUs.

3.5. Training Objective

Our training does not require explicit 3D supervision. We render the reconstructed GS on the target views for supervised training, and minimize the rendering loss that is a combination of Mean Squared Error (MSE) and perceptual loss based on VGG-19 features [40]:

$$\mathcal{L}_{\text{RGB}} = \text{MSE}(\mathbf{I}_{\text{pred}}, \mathbf{I}_{\text{gt}}) + \lambda \, \text{Perceptual}(\mathbf{I}_{\text{pred}}, \mathbf{I}_{\text{gt}}) \quad (5)$$

For non-autoregressive training, we randomly sample unordered input–target image pairs from the dataset. For the autoregressive model, we instead sample ordered input sequences to better simulate streaming use case.

Apart from rendering loss, for scene-level data, we use depth regularization with the scale-invariant depth loss [67] by aligning the Gaussian position along the depth direction

(z axis) with ground truth depth for that Gaussian. We opt for using the monocular depth estimator [50] for pseudo ground truth since we found that feedforward MVS methods like VGGT [49] provide less detailed depth prediction, albeit being multi-view consistent. Similar to Long-LRM [67], we also use opacity regularization to reduce the number of Gaussians. Our final loss function can be written as follows:

$$\mathcal{L} = \mathcal{L}_{\text{RGB}} + \lambda_{\text{depth}}\mathcal{L}_{\text{depth}} + \lambda_{\text{opacity}}\mathcal{L}_{\text{opacity}} \quad (6)$$

4. Experiments

4.1. Model and Training

Model Details Our model consists of 24 LaCT blocks with the hidden dimension of 768. The window attention layers have 64-dimension for each head with QK-normalization for stability. For the feedforward layer, we use a two-layer MLP with 4 intermediate expansion ratios for the intermediate dimension. We use a patch size of 8×8 for the image tokenizer. Our architecture shares the same parameterization as TTT-LVSM [66] except for the decoding module, allowing us to effectively leverage its pretrained weights as a strong initialization for our model.

Training Strategy We adopt a curriculum training strategy that progresses from low to high resolution, motivated by two main reasons. First, fast low-res pretraining enables the model to train with a large batch size with faster iteration time. Second, we found that even with pretrained TTT-LVSM [66] checkpoints at high-resolution (*i.e.* 960×640), the model cannot predict reasonable Gaussians at the beginning iterations, leading to excessive GPU memory usage due to the rendering of a large number of Gaussians.

Therefore, for scene-level training, we train our model with three resolution stages, from 256×144, 512×288 and 960×540 respectively. For each stage, we resize the images to the target resolution, which all have the same aspect ratio as the original dataset. For all stages, we first determine a continuous range based on the start and end frames from the entire video sequence from the dataset. The range is randomly sampled from 128 to 512 for each sample to ensure enough coverage of the scene. Then, we randomly sample 124 frames from this range, from which both input and target views will be further sampled. They are ensured to have overlap frames for stable training. We train the model across 16 to 64 input views. For training, we use the input views as the virtual views and found that provides the best results. More details can be found in the supplemental.

4.2. Datasets

Object-level Dataset We train our object-level reconstruction model on the Objaverse dataset [10]. Following prior works [16, 64], each 3D object is centered and normalized
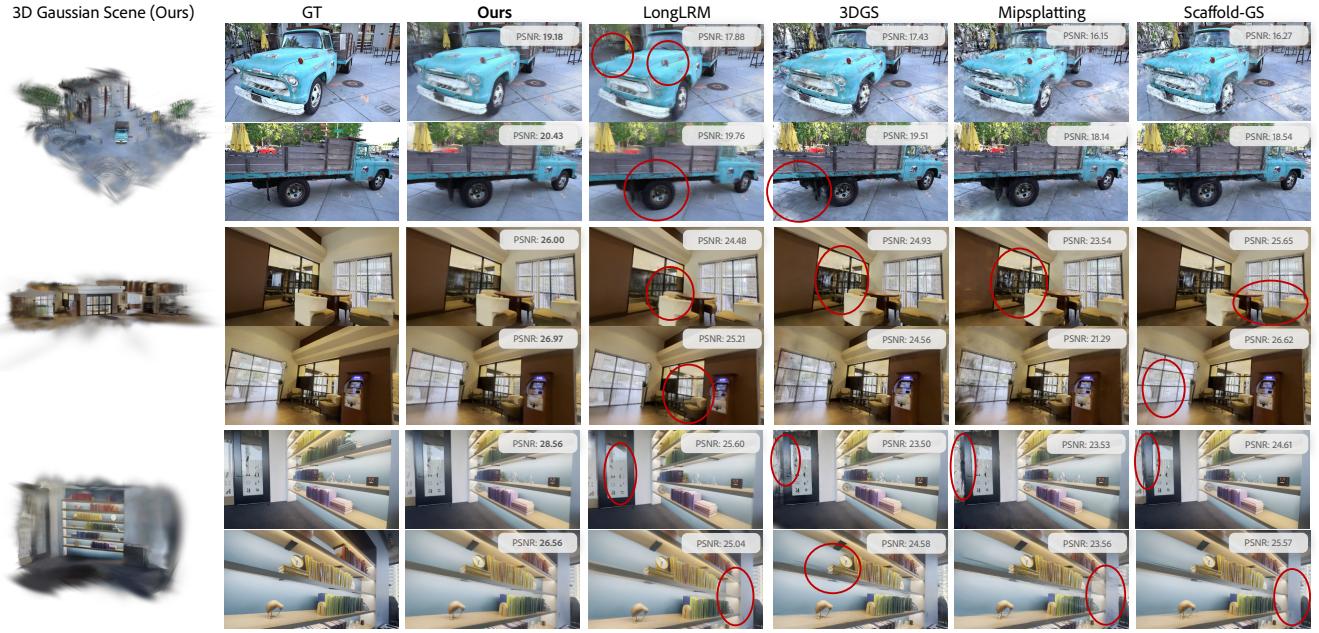
Figure 4. Qualitative comparison between our method and baseline approaches. Our model reconstructs the 3DGS scene with higher fidelity than both optimization-based and feedforward baselines, as also reflected in the PSNR metrics. Please zoom in for a better comparison.

Figure 5. We demonstrate that our high-resolution 1024×1024 3DGS tttLRM can be effectively used for image-to-3D generation when combined with a multi-view generator. Our model enables the reconstruction of fine-grained, photorealistic details *e.g.*, hair, fur, and text, from the input images. Video results are provided in the supplemental material.

to fit within a bounding box of $[-1, 1]$. We render 32 views per object, where cameras are randomly distributed around the object at distances uniformly sampled from $[1.5, 2.8]$. All images are rendered at a resolution of 512×512 under uniform lighting conditions. In total, we use 730K objects for training. We evaluate our model on 100 objects sampled from the Google Scanned Objects (GSO) dataset. For evaluation, we select a few views as input and the same random 8 views for testing.

Scene-level Dataset We train our model on the challenging DL3DV-10K [26] dataset, which consists of 10,510 high-resolution videos, each containing up to 500 keyframes with camera pose annotation obtained from COLMAP [34]. The testing set of DL3DV-140 contains 140 test scenes. We use the same input and target split from that provided by Long-LRM [67]: the testing views are evenly selected from every 8 views (around 40 images each scene) and input views are selected based on K-means clustering based on camera po-

sitions and view directions. We also tested our model on Tanks&Temples [23] dataset.

4.3. Baselines and Metrics

Object-level We compare our method with GS-LRM [64], an attention-based method. We train the model under 8 input views setting with the same iterations of our method.

Scene-level Previous feedforward reconstruction methods like GS-LRM [64] cannot be directly extended to long sequence due to the high complexity of attention. Long-LRM [67] is the only available feedforward method that can handle more than 16 input views. We also include three optimization-based methods: 3DGS [21], Mip-Splatting [62] and Scaffold-GS [30].

Metrics For all baselines, in addition to visual comparisons, we report three metrics to evaluate novel view synthesis quality: PSNR, SSIM, and LPIPS [65].
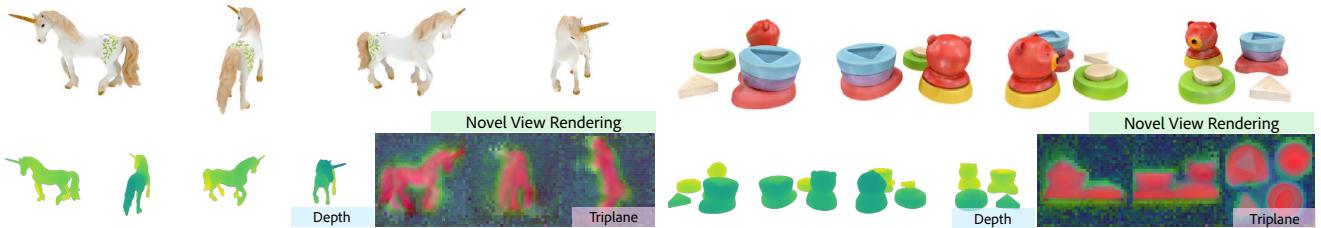
Figure 6. We show that tttLRM, as a general framework, can also interpret the latent 3D memory into formats besides 3DGS. In this experiment, we use a set of triplane tokens to query the fast weights and then fine-tune the model for triplane-based NeRF reconstruction. We visualize the resulting triplanes and present the corresponding renderings and depth maps for 4 views at a resolution of 512×512.

4.4. Results

Object-level We present quantitative comparison results under varying resolutions and numbers of input views in Table 1. Across all settings, our method consistently outperforms the baselines. At lower resolutions and shorter sequences, our inference speed is comparable to full attention models, as it is primarily determined by MLP operations. Thanks to the linear complexity of our architecture with respect to sequence length, at a resolution of 512×512, our model runs twice as fast as attention-based models while achieving over a 1 dB PSNR improvement. Our model also demonstrates strong generalization ability—when trained with 8 input views, it can be directly applied to 16 or 24 views (last two rows in the table). With longer sequences, inference becomes substantially faster, and rendering quality further improves through test-time training.

Moreover, our model scales seamlessly to 1024×1024 resolution, whereas GS-LRM encounters out-of-memory issues under high-resolution training. Results in Figure 5 show that our model can achieve high-quality 3D reconstruction of humans, animals, and texts from a single image when combined with a multi-view diffusion model.

Table 1. Comparison between our method and GS-LRM [64] on the GSO dataset under different resolutions and numbers of input views. Our method consistently outperforms GS-LRM in both inference speed and reconstruction quality, and also shows strong generalization ability. V. denotes the number of virtual views used to query the fast weight, which equals input views unless noted.

Method	Resolution	Views	Time (s) ↓	PSNR↑	SSIM↑	LPIPS↓
GS-LRM [64]	256×256	8	0.1	31.55	0.964	0.028
Ours		8	**0.1**	**33.14**	**0.024**	**0.972**
GS-LRM [64]		8	0.7	32.83	0.969	0.029
Ours		8	**0.3**	**34.02**	**0.974**	**0.025**
GS-LRM [64]	512×512	16	2.5	33.55	0.976	0.023
Ours		16 (10 V.)	**0.8**	**34.67**	**0.978**	**0.022**
GS-LRM [64]		24	5.5	33.26	0.976	**0.022**
Ours		24 (10 V.)	**1.1**	**34.80**	**0.979**	**0.022**

Scene-level We further evaluate our model on scene reconstruction, as shown in Table 2. Compared with optimization-based methods that tend to overfit to input views, our method achieves better results on 16 and 32 input views. With more input views, it remains competitive

Table 2. Quantitative comparison on both DL3DV-140 and Tanks&Temples datasets under different numbers of input views. Our method surpasses previous feedforward methods and is comparable with optimization-based methods. **Note that Long-LRM trains a separate model for each input view, while we are a single model across all input views.** Our model can be **linearly accelerated with multiple GPUs**, here we report time on 1 A100.

Views	Method	Time↓	DL3DV-140			Tanks&Temples		
			PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
16	3D GS$_{30k}$	13m	21.20	0.708	0.264	16.76	0.598	0.334
	Mip-Splatting$_{30k}$	13m	20.88	0.712	0.274	16.82	0.616	0.332
	Scaffold-GS$_{30k}$	16m	**22.13**	**0.738**	**0.250**	17.02	**0.634**	**0.321**
	Long-LRM (16v model)	**0.4s**	22.66	0.740	0.292	17.51	0.555	0.408
	Ours (single model)	3.6s	**23.60**	**0.784**	**0.255**	**18.15**	**0.613**	**0.360**
32	3D GS$_{30k}$	13m	23.60	0.779	0.213	18.10	0.688	0.269
	Mip-Splatting$_{30k}$	13m	23.32	0.784	0.217	18.39	**0.700**	**0.262**
	Scaffold-GS$_{30k}$	16m	**24.77**	**0.805**	**0.205**	**18.41**	0.691	0.290
	Long-LRM (32v model)	**1s**	24.10	0.783	0.254	18.38	0.601	0.363
	Long-LRM (32v model w/ optim)	12s	24.99	0.809	0.243	18.69	0.623	0.360
	Ours (single model)	7.2s	**25.07**	**0.822**	**0.215**	**19.22**	**0.662**	**0.305**
64	3D GS$_{30k}$	13m	26.55	0.852	**0.164**	20.78	**0.778**	**0.205**
	Mip-Splatting$_{30k}$	13m	26.29	0.850	0.166	20.08	0.759	0.220
	Scaffold-GS$_{30k}$	16m	**27.07**	**0.857**	0.173	**20.96**	0.768	0.240
	Long-LRM (64v model)	**3.7s**	24.77	0.804	0.322	18.89	0.614	0.356
	Ours (single model)	14.8s	**25.95**	**0.844**	**0.195**	**20.31**	**0.700**	**0.274**

in reconstruction quality, while being hundreds times faster. Moreover, one single tttLRM model can be applied to different sequence lengths and effectively generalizes to new datasets like Tanks & Temples.

Compared to the feedforward baseline Long-LRM [67], tttLRM achieves substantially better performance—approximately 1 dB PSNR improvement—across different numbers of input views. On the other hand, we show our model constantly outperforms Long-LRM even when it's combined with additional post-optimization. Furthermore, our method can be linearly accelerated by distributing the input across multiple GPUs, as described in Section 3.4.

Figure 4 shows visual comparisons between our method and baselines. tttLRM achieves better visual quality with fewer artifacts than optimization-based methods, thanks to the learned priors across diverse scenes. Our model also outperforms Long-LRM by reconstructing sharper and more detailed geometry (as shown in the red boxes).

Autoregressive Reconstruction We demonstrate the autoregressive reconstruction capability of our model in the

second row of Figure 1. With only 4 initial input views, the model already produces reasonable 3D Gaussian reconstructions; as additional views arrive (8 and 32 views), both the rendering quality and scene coverage progressively improve. Additional examples of autoregressive reconstruction are provided in the supplemental materials.

Decoding into Other 3D Formats Beyond using 3DGS as the output representation, our architecture can also decode the latent 3D representation into other formats, such as triplane-based NeRFs. As described in Section 3.2, replacing the virtual tokens with triplane tokens enables the fast weights to be queried as a triplane representation for NeRF reconstruction. We finetune the model with a rendering loss to enable this capability. We show the NeRF renderings and the corresponding queried triplanes in Figure 6. This demonstrates that our architecture is flexible and can generalize to different 3D output formats.

4.5. Ablation Study

We conduct ablation studies to analyze our design choices in LVSM pretraining and the autoregressive reconstruction strategy.

Pretraining from TTT-LVSM We investigate the effectiveness of leveraging pretrained knowledge for both Gaussian Splatting and triplane training at a resolution of 256×256. The GS reconstruction has 8 input views with patch size 8×8, while the triplane version has 4 input views with patch size 16×16. As shown in Figure 7, Using GS model as an example, initialization with pretrained checkpoints substantially accelerates convergence, especially in the early training stage, where models quickly reach a high PSNR compared to the one trained from scratch.

Moreover, as reported in Table 3, pretrained initialization not only improves convergence speed but also leads to higher final quality after full training. The gains persist even when trying to adapt the pretrained weights to different 3D representations. The results suggest that pretrained knowledge of novel view synthesis serves as an effective inductive bias for 3D reconstruction, improving both training efficiency and final rendering fidelity.

Table 3. Leveraging pretrained knowledge from novel view synthesis tasks improves the final 3D reconstruction quality across different 3D representations.

3D Rep.	Type	PSNR↑	SSIM↑	LPIPS↓
GS	w/o Pretrain	32.77	0.026	0.969
	w Pretrain	**33.14**	**0.024**	**0.972**
Triplane	w/o Pretrain	26.40	0.903	0.093
	w Pretrain	**27.87**	**0.925**	**0.075**

Autoregressive strategy In Section 3.3, We introduce an autoregressive reconstruction strategy. Here we consider a more efficient way called "Predict & Merge": instead of
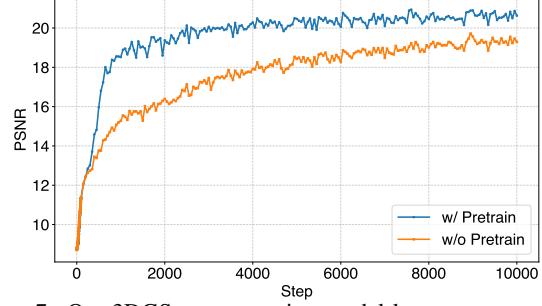

Figure 7. Our 3DGS reconstruction model leverages pretraining with LVSM on novel view synthesis tasks, which significantly accelerates learning and leads to better performance, compared to training from scratch.

generating a new 3DGS $G_{(b)} \leftarrow \mathcal{F}(W, \mathcal{I}^v(b))$ for each step, we reuse the previously predicted Gaussians $G_{(b-1)}$ and merge them with the newly predicted subset $G_{(b')} \leftarrow \mathcal{F}(W, \mathcal{I}^v_{(b')})$, forming $G_{(b)} = G_{(b-1)} \cup G_{(b')}$. Here, $\mathcal{I}^v_{(b')}$ is a subset of $\mathcal{I}^v_{(b)}$ containing only new virtual views not covered in $\mathcal{I}^v_{(b-1)}$. However, we found that though this approach is computationally more efficient, it cannot correct the accumulated errors in $G_{(b-1)}$, leading to worse results than our proposed full reconstruction method, as shown in Table 4.

Table 4. Although progressive GS prediction with merging provides more efficient computation, the reconstruction quality is degraded due to accumulated errors (compared on 32 views under 1K iterations finetuning).

	PSNR↑	SSIM↑	LPIPS↓
Predict & Merge	21.50	0.891	0.318
Ours	**23.63**	**0.904**	**0.259**

Optimizer and Losses Table 5 shows that use Muon as opitmizer can bring better results even on low resolution setting. It will bring better results on longer sequence (*e.g.* high resolution and more input views). Also, incorporating depth and opacity as regularization can help reduce opaque Gaussians.

Table 5. Ablation on 32 view 256×144 input with the same iterations across settings.

Muon	Opacity+Depth	PSNR↑	SSIM↑	LPIPS↓	Opacity> 0.001
✗	✗	20.44	0.649	0.295	96%
✓	✗	20.68	0.661	0.290	97%
✓	✓	**20.76**	**0.666**	**0.285**	**47%**

4.6. Discussions and Limitations

First, the fast-weight memory has a fixed size, which may limit its ability to handle highly complex scenarios with extremely large numbers of input views. Second, we observe that, compared with the pretrained LVSM model from which we fine-tune, our quality slightly degraded but we has much faster rendering speed and explicit 3D representations for flexible downstream tasks. Future works might design a better memory mechanism, further improve the quality,

and speed up the inference to enable real-time high-quality reconstruction for streaming inputs.

5. Conclusion

In this paper, we present tttLRM, a large reconstruction model that supports both feedforward long-context and autoregressive 3D modeling. Under the Test-Time Training framework, it produces implicit fast-weight representations and converts them into explicit 3D representations such as Gaussian splats and triplanes for efficient, high-quality novel view synthesis. Experiments on object- and scene-level datasets show that tttLRM outperforms prior feedforward methods in quality and scalability while approaching the speed of explicit representations. Our framework helps close the gap between neural network rendering and real-time explicit 3D systems.

Acknowledgment

References

[1] Joshua Ainslie, Santiago Ontañón, Yi Tay, et al. Jamba: Hybrid transformer–mamba models for efficient long-context processing. arXiv preprint arXiv:2403.19887, 2024. 3

[2] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. In *European conference on computer vision*, pages 696–712. Springer, 2020. 3

[3] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024. 3

[4] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16123–16133, 2022. 2

[5] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19457–19467, 2024. 3

[6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European conference on computer vision*, pages 333–350. Springer, 2022. 2

[7] Xingyu Chen, Yue Chen, Yuliang Xiu, Andreas Geiger, and Anpei Chen. Ttt3r: 3d reconstruction as test-time training. *arXiv preprint arXiv:2509.26645*, 2025. 2

[8] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *European Conference on Computer Vision*, pages 370–386. Springer, 2024. 3

[9] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024. 3

[10] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13142–13153, 2023. 5

[11] John Flynn, Michael Broxton, Paul Debevec, Matthew Duvall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2367–2376, 2019. 3

[12] John Flynn, Michael Broxton, Lukas Murmann, Lucy Chai, Matthew DuVall, Clément Godard, Kathryn Heal, Srinivas Kaza, Stephen Lombardi, Xuan Luo, Supreeth Achar, Kira Prabhu, Tiancheng Sun, Lynn Tsai, and Ryan Overbeck. Quark: Real-time, high-resolution, and general neural view synthesis. *ACM Trans. Graph.*, 43(6), 2024. 2

[13] Quankai Gao, Qiangeng Xu, Hao Su, Ulrich Neumann, and Zexiang Xu. Strivec: Sparse tri-vector radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17569–17579, 2023. 2

[14] Michael Goesele, Brian Curless, and Steven M Seitz. Multi-view stereo revisited. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, pages 2402–2409. IEEE, 2006. 2

[15] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021. 3

[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. 2, 3, 5

[17] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 conference papers*, pages 1–11, 2024. 2

[18] Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah Snavely, and Zexiang Xu. Lvsm: A large view synthesis model with minimal 3d inductive bias. *arXiv preprint arXiv:2410.17242*, 2024. 2

[19] Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cecista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. *URL https://kellerjordan.github.io/posts/muon*, 6. 3

[20] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020. 3

[21] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time

radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 2, 6

[22] Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *ACM Transactions on Graphics (TOG)*, 43(4):1–15, 2024. 2

[23] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36 (4):1–13, 2017. 6

[24] Yushi Lan, Yihang Luo, Fangzhou Hong, Shangchen Zhou, Honghua Chen, Zhaoyang Lyu, Shuai Yang, Bo Dai, Chen Change Loy, and Xingang Pan. Stream3r: Scalable sequential 3d reconstruction with causal transformer. *arXiv preprint arXiv:2508.10893*, 2025. 2

[25] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European Conference on Computer Vision*, pages 71–91. Springer, 2024. 2

[26] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22160–22169, 2024. 6

[27] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020. 2

[28] Yang Liu, Chuanchen Luo, Lue Fan, Naiyan Wang, Junran Peng, and Zhaoxiang Zhang. Citygaussian: Real-time high-quality large-scale scene rendering with gaussians. In *European Conference on Computer Vision*, pages 265–282. Springer, 2024. 2

[29] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. Vmamba: Visual state space model. *Advances in neural information processing systems*, 37:103031–103063, 2024. 3

[30] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20654–20664, 2024. 6

[31] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (ToG)*, 38(4):1–14, 2019. 3

[32] 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. 1, 2

[33] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022. 2

[34] Linfei Pan, Dániel Baráth, Marc Pollefeys, and Johannes L Schönberger. Global structure-from-motion revisited. In *European Conference on Computer Vision*, pages 58–77. Springer, 2024. 6

[35] Mehdi SM Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lučić, Daniel Duckworth, Alexey Dosovitskiy, et al. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6229–6238, 2022. 2

[36] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pages 9355–9366. PMLR, 2021. 3

[37] 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. 2

[38] Qiuhong Shen, Zike Wu, Xuanyu Yi, Pan Zhou, Hanwang Zhang, Shuicheng Yan, and Xinchao Wang. Gamba: Marry gaussian splatting with mamba for single-view 3d reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025. 3

[39] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3531–3539, 2021. 3

[40] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5

[41] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2437–2446, 2019. 3

[42] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5459–5469, 2022. 2

[43] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023. 3

[44] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024. 2, 3

[45] 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. In *European Conference on Computer Vision*, pages 1–18. Springer, 2024. 3

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia

Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 3

[47] Johannes von Oswald, Nino Scherrer, Seijin Kobayashi, Luca Versari, Songlin Yang, Maximilian Schlegel, Kaitlin Maile, Yanick Schimpf, Oliver Sieberling, Alexander Meulemans, et al. Mesanet: Sequence modeling by locally optimal test-time training. *arXiv preprint arXiv:2506.05233*, 2025. 3

[48] Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, et al. An empirical study of mamba-based language models. *arXiv preprint arXiv:2406.07887*, 2024. 3

[49] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025. 2, 5

[50] Ruicheng Wang, Sicheng Xu, Cassie Dai, Jianfeng Xiang, Yu Deng, Xin Tong, and Jiaolong Yang. Moge: Unlocking accurate monocular geometry estimation for open-domain images with optimal training supervision. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5261–5271, 2025. 5

[51] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20697–20709, 2024. 2

[52] 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 meshes. *arXiv preprint arXiv:2404.12385*, 2024. 2

[53] Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. Depthsplat: Connecting gaussian splatting and depth. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 16453–16463, 2025. 3

[54] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Pointnerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5438–5448, 2022. 2

[55] Jianing Yang, Alexander Sax, Kevin J Liang, Mikael Henaff, Hao Tang, Ang Cao, Joyce Chai, Franziska Meier, and Matt Feiszli. Fast3r: Towards 3d reconstruction of 1000+ images in one forward pass. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21924–21935, 2025. 2

[56] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems*, 37:115491–115522, 2024. 3

[57] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European conference on computer vision (ECCV)*, pages 767–783, 2018. 2

[58] Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. Recurrent mvsnet for high-resolution multi-view stereo depth inference. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5525–5534, 2019. 2

[59] Xuanyu Yi, Zike Wu, Qiuhong Shen, Qingshan Xu, Pan Zhou, Joo-Hwee Lim, Shuicheng Yan, Xinchao Wang, and Hanwang Zhang. Mvgamba: Unify 3d content generation as state space sequence modeling. *Advances in Neural Information Processing Systems*, 37:7580–7607, 2024. 3

[60] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions On Graphics (TOG)*, 38(6):1–14, 2019. 3

[61] 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. 1

[62] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19447–19456, 2024. 6

[63] Yuheng Yuan, Qiuhong Shen, Shizun Wang, Xingyi Yang, and Xinchao Wang. Test3r: Learning to reconstruct 3d at test time. *arXiv preprint arXiv:2506.13750*, 5, 2025. 2

[64] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. In *European Conference on Computer Vision*, pages 1–19. Springer, 2024. 2, 3, 5, 6, 7

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

[66] Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T Freeman, and Hao Tan. Test-time training done right. *arXiv preprint arXiv:2505.23884*, 2025. 2, 3, 5

[67] Chen Ziwen, Hao Tan, Kai Zhang, Sai Bi, Fujun Luan, Yicong Hong, Li Fuxin, and Zexiang Xu. Long-lrm: Long-sequence large reconstruction model for wide-coverage gaussian splats. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4349–4359, 2025. 2, 3, 5, 6, 7